KU LEUVEN

ARENBERG DOCTORAL SCHOOL
FACULTY OF ENGINEERING SCIENCE

# Quality maintenance in geographical data and services for spatial networks

**Joris Maervoet**

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

January 2014

# Quality maintenance in geographical data and services for spatial networks

**Joris MAERVOET**

Examination Committee:
Prof. dr. ir. J. Berlamont, chair
Prof. dr. P. De Causmaecker, supervisor
Prof. dr. ir. G. Vanden Berghe, supervisor
Prof. dr. ir. H. Blockeel
Dr. ir. J. Crompvoets
Dr. K. Verbeeck
Dr. P. Brackman
(RouteYou, Belgium)
Prof. dr. N. Van de Weghe
(Ghent University, Belgium)
Prof. dr. C. Claramunt
(Naval Academy Research Institute, France)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

January 2014

D/2014/7515/6
ISBN 978-94-6018-782-7

# Preface

Long years, difficult years... this would have summarized the PhD project I have just completed, if it had not been for the discovery of numerous nice service- and quality oriented people in my network. Beyond doubt, without their expertise and helpful argument, the road to completion would also have been less attractive.

First and foremost, I would like to thank my supervisors, Patrick De Causmaecker and Greet Vanden Berghe. They provided me with a challenging applied research environment for more than 10 years. This environment enabled me to become acquainted with both corporate life and the academic world. Moreover, I got the opportunity to combine this applied research with the study of a M.Sc. in applied computer science. Thank you for the numerous suggestions, feedback and support during my PhD.

I am very grateful to those people from industry who were willing to launch, finance and manage the research projects that contributed to this thesis: in particular to Pascal Brackman and Philiep De Sutter from the company RouteYou, and to Gert Vervaet from the company Tele Atlas (currently TomTom). Besides, much of this research would not have been possible without the support of the IWT (Vlaams agentschap voor Innovatie door Wetenschap en Technologie).

I would like to thank Hendrik Blockeel, Nico Van de Weghe, Katja Verbeeck, Joep Crompvoets and Christophe Claramunt for accepting to be members of the jury and for their useful feedback which helped improving the quality of this text. I want to thank prof. Berlamont for being the chairman of my examination committee.

I highly appreciated the collaboration with, the support of and the atmosphere created by all my former and current colleagues at the Computer Science group of KAHO Sint-Lieven (formerly 'vakgroep IT'): Annemie, Annick, Bart x2, Björn, Bram, Burak, Faysal, Filip, Geert, Geoffrey, Jan x2, Jannes, Joris, Jorn, Katja,

Koen x3, Laurens, Hilde, Mieke, Mihail, Murat, Mustafa, Nele, Peter, Philippe, Phyllis, Pieter x3, Sam, Stijn, Tahir, Tijs, Tim, Tony, Vincent, Werner, Wim, Wouter and Yvan. The same applies to the colleagues at RouteYou: Bénédict, Benjamin, David, Kevin, Niels and Pieter.

Special thanks go out to Erik Van Achter, who dotted the i's and crossed the t's in my articles and texts.

Last but not least, I would like to thank my parents and in particular my girlfriend Lore for supporting me in all sorts of activities related to this PhD. Since February 2011, our son Ferre provides the entertainment at home. He will share this responsibility with a sibling by February 2014.

# Abstract

The present thesis describes various applications based on concrete questions originating from GIS practice. It aims at the design and validation of (1) methods to maintain the quality of large amounts of geographical data, and, (2) service components for spatial networks, dealing with efficiency in environments with a limited amount of resources.

In the first place, such an endeavour involves quality maintenance in geographic information systems (GIS). It is pursued through the identification of patterns, describing *relational* regularities, and corresponding outliers indicating probable inconsistencies in the data. This approach is different from classical approaches to outlier detection and quality analysis in GIS, traditionally relying on techniques prospecting for statistical and positional deviation in geographical data.

It moreover implies the efficiency enhancement of shortest path (cost) approximation components through auxiliary structures in spatial weighted graphs. Efficiency in this context mainly refers to the approximation accuracy, the calculation time required to produce an approximation and the structures' compactness. A first part describes the integration of a hierarchical shortest path algorithm in a multi-tier architecture, considerably restricting the amount of data used during a routing query. Secondly, an evaluation framework for approximate distance oracles is introduced, taking into account the overall accuracy performance of the oracle. An advanced oracle, based on graph partitioning and minimizing the absolute approximation error in spatial graphs, is presented and evaluated.

The fast automated discovery of attractive closed paths in graphs comprises the third theme of this work. This problem arises from the introduction of edge attractiveness scores in graphs. The tour suggestion problem for outdoor activities aims at optimizing the arc and vertex attractiveness of a closed path in a transportation network graph, satisfying a set of constraints. An algorithm

of low computational impact generating heuristic solutions to this problem is presented.

The underlying concepts and approaches introduced in the present thesis are applicable in more general domains. The approach to relational outlier detection applies to quality maintenance in data-rich intelligent systems. The auxiliary structures enhancing efficiency of approximative shortest path algorithms can be reused for exact algorithms. The automated discovery of structures of high attractiveness is valuable for a broader class of applications in graphs with dually weighted edges. Four software components resulting from this research have been adopted by industry.

# Beknopte samenvatting

Deze thesis beschrijft verscheidene toepassingen gebaseerd op concrete vragen afkomstig uit de GIS-praktijk. Ze beoogt het ontwerp en de validatie van (1) methoden om de kwaliteit van grote hoeveelheden geografische gegevens te handhaven, en (2) service-componenten voor spatiale netwerken, die omgaan met efficiëntie in omgevingen met een beperkte hoeveelheid resources.

In de eerste plaats heeft dit onderwerp betrekking op de kwaliteitshandhaving in geografische informatiesystemen (GIS). Hiertoe wordt bijgedragen door de identificatie van patronen die *relationele* regelmatigheden en overeenkomstige afwijkingen in de data beschrijven. Dergelijke afwijkingen zijn een indicatie voor mogelijke inconsistenties in de gegevens. Deze benaderingswijze verschilt van de klassieke benaderingen voor de detectie van afwijkingen en kwaliteitsanalyse in GIS, omdat deze traditioneel steunen op technieken die op zoek gaan naar statistische en positionele afwijkingen in geografische gegevens.

In een tweede luik wordt de efficiëntie van benaderingscomponenten van (kosten van) kortste paden bevorderd door middel van hulpstructuren in spatiale gewogen grafen. Efficiëntie verwijst in deze context hoofdzakelijk naar de benaderingsaccuraatheid, de vereiste rekentijd om een benadering te genereren en de compactheid van de structuren. Een eerste deel beschrijft de integratie van een hiërarchisch korste-pad-algoritme in een multitierarchitectuur, die de hoeveelheid gegevens gebruikt tijdens een routeringsprocedure aanzienlijk beperkt. Daarna wordt een evaluatieraamwerk voor zogenaamde *afstandsbenaderingsorakels* geïntroduceerd, dat rekening houdt met de globale accuraatheidsscores van het orakel. Een geavanceerd orakel, gebaseerd op het partitioneren van de graaf en het minimaliseren van de absolute benaderingsfout in spatiale grafen, wordt voorgesteld en geëvalueerd.

Het snel en automatisch onthullen van aantrekkelijke gesloten paden in een graaf vormt het derde thema van dit werk. Dit probleem ontstaat uit het aanbrengen van aantrekkelijkheidsscores aan de kanten van een

graaf. Het toursuggestieprobleem voor buitenrecreatie streeft een optimale aantrekkelijkheid na van de bogen en knopen van een gesloten pad, dat voldoet aan een verzameling beperkingen. Een algoritme met een lage rekentijd dat heuristische oplossingen voor dit probleem aanreikt in een transportnetwerk wordt voorgesteld.

De onderliggende concepten en benaderingswijzen die in deze thesis aan bod komen zijn toepasbaar in algemenere domeinen. De benadering voor het opsporen van relationele afwijkingen is relevant voor kwaliteitshandhaving in gegevensrijke intelligente systemen. De hulpstructuren die de efficiëntie bevorderen van approximatieve kortste-pad-algoritmen kunnen worden hergebruikt voor exacte algoritmen. Het automatisch onthullen van structuren met een hoge aantrekkelijkheid draagt bij tot een bredere klasse van toepassingen in grafen waarvan de kanten twee gewichten hebben. Vier softwarecomponenten die uit dit onderzoek voortvloeien werden overgenomen door de industrie.

# Abbreviations

| | |
|---|---|
| **ACE** | A Combined Engine |
| **ADO** | Approximate Distance Oracle |
| **API** | Application Programming Interface |
| **BTP** | Bus Touring Problem |
| **CI** | Complex Intersection |
| **CWA** | Closed World Assumption |
| **DBMS** | DataBase Management System |
| **DCW** | Dual Carriage Way |
| **DILP** | Descriptive Inductive Logic Programming |
| **EA** | Early Access |
| **EILP** | Empirical Inductive Logic Programming |
| **FOW** | Form Of Way |
| **FRC** | Functional Road Class |
| **FW** | Feasibility Window |
| **GIS** | Geographic Information Systems |
| **GUI** | Graphical User Interface |
| **HAV** | Highly Attractive Vertex |
| **IILP** | Interactive Inductive Logic Programming |
| **ILP** | Inductive Logic Programming |
| **KDD** | Knowledge Discovery in Databases |

| | |
|---|---|
| **LBS** | Location Based Services |
| **LCA** | Least Common Ancestor |
| **LE** | Late Exit |
| **LP** | Logic Programming |
| **MLHNP** | Multi-Level Heuristic Node Promotion |
| **MOSP** | Multi-Objective Shortest Path |
| **OATSP** | Outdoor Activity Tour Suggestion Problem |
| **OP** | Orienteering Problem |
| **OSM** | Open Street Map |
| **PCTSP** | Prize Collecting TSP |
| **POI** | Point Of Interest |
| **RDBMS** | Relational DataBase Management System |
| **RMSE** | Root-Mean-Square Error |
| **SAR** | Spatial Association Rules |
| **SDM** | Spatial Data Mining |
| **SP** | Shortest Path |
| **SPADA** | Spatial Pattern Discovery Algorithm |
| **TSP** | Traveling Salesperson Problem |
| **TSPLT** | Tour Suggestion Problem for Leisure and Tourism |
| **TTDP** | Tourist Trip Design Problem |
| **UML** | Unified Modeling Language |
| **WARMR** | Wanted: Association Rules over Multiple Relations |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Context

At the beginning of the 21st century, digital map-making was a specialized process, exclusively executed by cartographers, under the authority of governmental or private organisations. Offering these maps and corresponding geographical services on the web, used to be a difficult and complicated task. During the last decade, the world of *web cartography* has been subject to drastic changes.

In a first evolution, map-making became a matter of combining many pieces of map data originating from an increasing number of sources. The map-making industry or *geographic content providers* started to involve a substantially higher number of methods and sources in the collection and synthesis process of geographical data, e.g. mobile mapping and crowd-sourcing. It is the role of map merging, often referred to as *data conflation*, to combine geographical data from separate sources into one map, preserving data accuracy, avoiding redundancy and solving data conflicts, as described by Longley et al. [73]. With the advent of collaborative mapping initiatives such as OpenStreetMap (OSM), the map-making process became both more transparent and, as such, democratic (as stated by Lin [70]), and geographical data became the result of a patchwork contributed by many individuals.

A second evolution concerns the advance of easily integratable web components that facilitate offering online maps and building online services regarding geographical data. Both open-source and commercial software projects were launched easing the realisation of interactive maps on the web. Open web

standards were developed for the exchange of geographical data. The so-called *mashups* are web applications enabling the fast integration of services and data from multiple sources in a single graphical interface through open application programming interfaces (APIs). A service example is map tiling, which facilitates the instant visualisation of a map according to a specified map template. Geocoding implies the translation of a postal address into geo-referenced coordinates. Routing involves identification of the optimal path from a specified origin to a specified destination with regard to a certain criterion such as distance or travel time.

Both evolutions involve interesting challenges. The first evolution raises the concern of how the quality of geographical data can be controlled in this many-pieces-from-multiple-sources context. The latter trend raises the challenge of designing service components that efficiently answer questions regarding geographical data in instant time and with a limited amount of other resources. For example, limiting the memory usage of a web application augments the number of parallel tasks a given server infrastructure can handle. Short response times enhance the end user experience of applications such as browsing maps and trip planning. The present thesis anticipates these challenges.

## 1.2 Research context and industrial cooperation

The contributions in this thesis are based on joint research with industry.

The Quality Gate project aimed to develop a system that is capable of maintaining the quality of data located in the central database of the company Tele Atlas. At the moment of the project launch, this company[1] was market leader in the provision of geographical data to other companies. The geographical data supports three types of applications: navigation systems, geographic information systems (GIS) and applications that provide location-based services (LBS). In these application areas there is an increasing demand for data quality. In certain situations companies could even be legally liable for inaccuracies in geographical data. Moreover, Tele Atlas was a typical example of a company relying on multiple sources for geographical data acquisition. It managed a fleet of mobile mapping vans, assisting a team of specialists in the semi-automatic mapping of the delimited regions they had been assigned to. This process was occasionally subject to human mistakes and structural differences due to individual preferences.
Quality Gate was a research and development (O&O) project funded by the

[1]Since 2007, the company is a wholly-owned subsidiary of automotive navigation system manufacturer TomTom.

IWT. In this project, the company realized the infrastructural and process-based aspects of quality maintenance of geographical data. The complementary approach by the author focused on data mining techniques to identify patterns in the geographical data automatically and on tracing corresponding anomalies. This data has extensive structural characteristics: it consists of objects or *features* enclosing both a spatial representation (which typically is a point, a linestring or an area) and a tree of attribute information. The *data model* links a spatial representation type and a tree of (non-spatial) attribute types to each feature type. A petrol station is for example represented by a point, and its attribute tree encloses the station's address, brand and fuel types. Furthermore, it links *associations* types to two or more feature types and again an attribute type tree. Associations describe non-spatial relations between features, such as a forbidden traffic manoeuvre between roads for a specified type of vehicle.

The major part of this thesis was achieved during a project in cooperation with the company RouteYou.com. This company manages a web 2.0 environment enabling users to interactively create, share and use routes in the context of leisure and tourism. Besides, it offers a routing platform for various application developers and digital content providers. Much of this is realized by web components offering online maps and services regarding geographical data, such as map tiling and geocoding. The company owes its unique position in the market of recreational navigation to the maintenance of a set of transportation network graphs, tailored to the needs of several *outdoor activity modes* such as hiking and mountain biking. Each edge in these graphs encodes a score, taking into account the physical and scenic road characteristics and the traffic regulations with regard to the mode. This score will further be referred to as *attractiveness* or *suitability*. The transportation graphs annotated by attractiveness give rise to several useful applications for any outdoor activity mode. The computation of a route of interest between two points selected by the user, for instance, entails a trade-off between attractiveness and the length of the route. The derivation of a connected subnetwork consisting of edges of high attractiveness is another example. The models and methods implied by these applications often differ from traditional approaches in the field.
The industrial PhD project (Baekelandmandaat) 'Structural heuristics for personalized routes' funded by the IWT anticipates these differences. It focuses on the development of structural models in order to index, structure and facilitate dynamic geographical knowledge such that performant algorithms can use this knowledge to offer routing services in the context of tourism and leisure. The geographical data faced consists of directed graphs with a spatial representation and of which edges have associated attractiveness scores and lengths. This data is authored by the company's geomatics department and is assumed to be correct. Optionally, this data is enriched by points of interest (POIs) which a spatial position and an attractiveness. Besides, in an operational

context, the company maintains a multi-tier architecture and the geographical data is stored in a spatial database. This architecture supports a multi-purpose GIS system and enables easy geographical data updates.

## 1.3 Themes and objectives

The research in this thesis is based on concrete questions originating from GIS practice. The present doctoral dissertation aims at the design and validation of

- methods to maintain the quality of large amounts of dynamic geographical data originating from multiple sources, and,
- service components for spatial networks, efficiently answering queries regarding spatial networks in low computational time and with a limited amount of resources.

This work was performed in close collaboration with a number of companies, who each had specific goals. The dissertation reflects this: it focuses on three different themes, with as main common feature their relevance for GIS and spatial networks:

- **Geographical data quality.** This first theme pursues the maintenance of the quality of data stored in GIS, through the identification of patterns describing data regularities. The commercial and the collaborative map-makers are facing very similar challenges with regard to quality maintenance: the amount of data is huge, and the data as well as the data model is subject to continuous updates. For large content providers such as Tele Atlas, the discipline of (semi-)automated identification of outliers and quality analysis in GIS, was, until 2007, restricted to the identification of positional inaccuracies and inconsistencies or referred to the statistical deviation (cf. clustering, trend analysis) of non-spatial attributes. The present work advocates for the application of relational data mining techniques in order to find outliers indicating probable inconsistencies. This approach better integrates the structural aspects of the data, preserves scalability and is compatible with evolving data models. It should be noted that outliers in geographical data do not necessarily correspond to errors, but can exhibit specific knowledge.
- **Shortest path (cost) approximation.** A second theme implies the construction of structures from spatial weighted graphs enabling instant shortest path (cost) approximation, in resource-constrained environments

where exact shortest path approaches cannot be applied. Amongst the common (web) services components for geographical data, routing is the most computationally intensive component. As is customary, routing entails finding the path of lowest cost between two vertices in a weighted graph. Recently proposed algorithms, such as a hybrid approach by Goldberg et al. [42], the contraction hierarchy approach by Geisberger et al. [39] and the transit-node approach by Bast et al. [7], return the exact solution in an effective way. The second approach has been made publicly available in the form of a standalone open-source software component. Any of these approaches require a preprocessing (or precalculation) phase, during which indexed structures are built, enabling a fast answer to any routing query. The best performing systems expose a clear trade-off between the required preprocessing and query time. However, none of these approaches or systems integrate with RouteYou's multi-tier architecture, considerably restricting the amount of (preprocessed) data used during query time, and requiring geographical data updates. These goals can be realized when the exactness of these systems is abandoned i.e. they generate paths of low cost approximating the lowest cost. An approximate distance oracle is a very related component to this theme: it is a compact data structure or model built from a graph, that is able to return an approximation of the shortest path cost between any two vertices of a graph in instant time. It often trades compactness for approximation accuracy. This type of components has many applications where only a cost approximation by a compact component is required.

- **Discovery of structures related to attractiveness in spatial graphs.** This third theme focuses on the discovery of closed paths of high attractiveness in graphs where the edges have both a length and an attractiveness score. It gives rise to a web component for tour suggestion for the company RouteYou. The fast and automated construction of closed paths supports this component generating tours of a specified length and of optimal attractiveness with regard to one of the company's outdoor activity modes. This is a novel approach since any of the conventional approaches to tour suggestion for leisure and tourism emphasize the selection of a sequence of vertices of different attractiveness, often of a complete graph.

**Relations between the themes.** The quality of geographical data, maintained by techniques discussed in theme 1, is of high influence on the quality of services based on this data. The shortest path approximation algorithms treated in the second theme are important examples of this type of services, and are the basic component of point-to-point route planners and navigation systems. Anomalies in geographical data, which are detected by techniques

covered by the first theme, can cause serious flaws with regard to route planning and navigation, ranging from travel time estimation deviation, over detours or blocked off passages in the suggested path, to manoeuvres which are illegal or put people into danger. The shortest path approximation algorithms operate on spatial networks, extracted from this geographical data, which is rich in structure. In case of navigation in transportation networks, this extraction process mainly determines (1) which ways and junctions must be included in the network in the form of edges and nodes, and, (2) the scores or weights to be assigned to the edges. This process is different for each routing mode and each means of transport.

Conversely, formal requirements imposed by services can be mapped backwards on data regularities for quality maintenance. Full network connectivity, for example, ensures that a shortest path algorithm is able to return a path between any two nodes in the network. This property can be translated back into a set of quality rules such that a connected network is extracted for any data set satisfying these rules.

Point-to-point navigation along the most attractive route is the common setting of the second and the third theme. Although the shortest path approximation algorithms apply to spatial transportation networks in general, these algorithms support point-to-point navigation along the most attractive route. For this type of navigation, the edge weights of the graph are derived from both the edge's length and attractiveness. The tour suggestion module in theme 3 operates in spatial graphs in which the edges have both a length and an attractiveness, and has attractiveness-based point-to-point navigation as a component.

## 1.4 Structure of the thesis

The structure of this work is as follows. Chapter 2 is situated in the geographical data quality theme and accounts for the feasibility of applying inductive logic programming to identify anomalies in the geographical data of the company Tele Atlas. This relational data mining technique integrates the representation of logic programming. A small case study deals with the extraction of regular and irregular patterns in relation to attributes of road elements and junctions of a small geographic region. Chapter 3 continues this work with the formalization of outlier detection in relational data and GIS. An integrated tool for extracting a broad scope of valuable knowledge about outliers in GIS is presented, supporting an evolving data model. The added value of this tool is illustrated by two experiments and a sanity check.

Chapter 4 treats the integration of a shortest path algorithm in a multi-tier web architecture. An approximation algorithm based on the hierarchy of a road network (cf. class labels) integrates best in this architecture. It however requires an irreversible graph transformation during preprocessing, which is problematic in contemporary route planning applications. Two alternative geographical data processing steps during the preprocessing and two adaptations of the algorithm are proposed in order to restore the algorithm's effectiveness. Chapter 5 advocates for the evaluation of approximate distance oracles by the root-mean-square error of its network distance approximations. It introduces an oracle for spatial graphs based on clustering graph vertices that share similar shortest paths starting or ending in these vertices. Both chapters on shortest path (cost) approximation are supported by an experimental analysis of the path cost accuracy for an extensive set of sample routing queries.

The theme of the discovery of closed paths of high attractiveness is represented by Chapter 6. It introduces the outdoor activity tour suggestion problem in order to generate closed paths consisting of edges of high attractiveness and subject to spatial constraints in graphs where the edges have both a length and an attractiveness score. A case study on closed paths illustrates the generation of a set of tour suggestions satisfying the constraints.

This work resulted in four software components that have been adopted by industry.

# Chapter 2

# Feasibility study of applying descriptive ILP to large geographic databases

The present chapter discusses a case study aiming to discover regularities and anomalies in large databases containing geographical data, to improve and maintain the overall data quality. The application of Inductive Logic Programming (ILP) and descriptive ILP in particular to this case is discussed and motivated. In an experiment on real-world data, a classical descriptive ILP algorithm (WARMR) is applied to the hamlet of Beggen, Luxemburg, to mine for rules describing regularities. The algorithm adopts the setting of learning from interpretations. In this setting, the data is divided into smaller parts, called interpretations, and rules are learnt about multiple relations inside these parts. In a next stage, the violating interpretations of the rules could be traced to identify candidate anomalies. A rule export module was set up to feed the results to a rule checking engine for further validation of this experiment. Finally, the results are discussed, the feasibilities of the system used in the case study are assessed and possibilities with regard to a larger scale application of the experiment are discussed.

The research in this chapter is part of an R&D project funded by IWT (050730). I would like to thank Ann Nowé (Vrije Universiteit Brussel) for suggesting ILP as an interesting approach for quality control in geographic databases, and Luc De Raedt (KU Leuven) and Hendrik Blockeel (KU Leuven) for the software support and interesting suggestions. The work in the present chapter

has been published as *Maervoet, J., De Causmaecker, P., Nowé, A., Vanden Berghe, G. (2008). Feasibility Study of Applying Descriptive ILP to Large Geographic Databases. Proceedings of the Workshop on Mining Multidimensional Data (MMD). European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD). Antwerp, 15-19 September 2008 (pp. 1-15).*

## 2.1 Introduction

This chapter accounts for the feasibility of applying ILP and its subdomains to the geographical data in order to control the quality of data located in the central database of the company Tele Atlas. ILP is a relational data mining technique in which input data as well as output patterns have a relational representation: the language of logic programs (or similar). This makes ILP an interesting candidate to apply to Quality Gate, since the data of interest is rich in structure. A possible approach would be to employ supervised learning and look for explanations of given examples of irregularities. Another approach is to use unsupervised learning and look for regularities in the data in the form of rules, of which the violations are then candidate irregularities. The latter approach offers the most interesting functionality and is the one used in our case study. It is a form of indirect unexpected knowledge mining. It should be mentioned that not any set of violations to a rule indicates a set of erroneous situations in the database. The violations can exhibit exceptional geographical situations such as the end of a motorway. Plantevit et al. [84] classify methods for unexpected rule discovery into user-driven methods (intervention by a human expert) and data-driven approaches, which are again subdivided into using unexpectedness-oriented measures on classical algorithms and new algorithms integrating new concepts of unexpectedness. In the case study we investigate the data-driven approach using unexpectedness-oriented measures on classical algorithms.

Outlier detection in multidimensional data has been studied extensively in the domains of statistics and automated learning. A lot of the methods proposed are based on proximity [13] or density [57] analysis. Another approach is to cluster the data first and to identify the entities that lie outside any cluster as outliers. Aggarwal and Yu [2] proposed the use of evolutionary algorithms to discover outliers in high-dimensional input data. However, our geographical data needs a more relational approach because the input data consists of several interrelated data types. Therefore, approaches originating from the field of ILP, in which patterns are learnt from relational data, are more likely in these circumstances.

Koperski and Han [60] were the first to define the concept of spatial association rules and to introduce an appropriate mining method. Further related work in geographical rule mining has been carried out at the University of Bari, Italy. In this work, the geographic hierarchy is integrated in the mining process. In [77] and [5], census data from Stockport is analysed by learning rules about socio-economic issues (e.g. commuter habits) in small districts to support transport planners. Furthermore, Lisi and Malerba [71] designed a hybrid language, called AL-log, that allows relational and structural description of data in order to use ILP to induce multiple-level association rules. The system is more performant than classical ILP with hierarchical information as background knowledge because taxonomic reasoning is integrated directly in the search process. In [76] geographical concepts are learnt which are not explicitly modelled. The system provides end-users with a tool for the automatic recognition of morphological elements (e.g. a fluvial landscape). In [16], geographic impact factors are learnt for rent prize categories in Munich. Most of the work is supervised learning. Another difference with the approach of our case study is that eventually some spatial knowledge is put into the preconditions of our experiment and that our outcome rules do not necessarily contain literals describing spatial relationships. Furthermore, our case has special interests in unexpected knowledge mining.

Another related approach has been explored by Kuramochi and Karypis [61], who applied finding frequent patterns on graph-modelled input data successfully. In this case the mining procedure is optimised using graph properties such that uninteresting subgraphs can be pruned, candidate patterns can be generated and frequencies can be counted efficiently. Again, it differs with our work because we are not necessarily looking for rules with spatial relationships. Furthermore, ILP goes beyond the propositional rule learning employed by [61].

The next section presents the problem of quality control for geographical data and its main challenges. A rough sketch of the company's data model is given, together with some sample rules. Section 3 motivates the application of ILP and its subdomains to Quality Gate. The fourth section outlines an experiment in which the WARMR algorithm is applied to a modified subset of the database. It is shown how to look up anomalies and export outcome rules into the representation used by the company. Section 5 discusses the experiment of the previous chapter. Encountered problems are identified. It presents suggestions on how to apply the algorithm on a larger scale. The last section is the conclusion of this work and contains some suggestions for further research.


Figure 2.1: Geographic data collection with mobile mapping vehicles

## 2.2 Problem description

### 2.2.1 Quality control for geographical data

The company uses several methods to collect geographical data. Beside the processing of aerial photographs and data originating from other organisations (such as the government), the acquisition by mobile mapping vehicles is the most important way to collect data. These vehicles are controlled by mobile specialists who proceed to regions of interest to create, to validate or to update field data. The specialist is provided with a set of tools that support the semi-automatic acquisition of data.

Most of the anomalies that reside in the database originate from human mistakes or from inconsistency between different sources. Moreover, not all kinds of mistakes are immediately traceable. Some only become apparent when a considerable number of updates are combined. Others need an elaborate search in the central database.

In a recent manual test case, data engineers started to investigate the intrinsic logic behind the spatial data, based on anomalies they encountered in the past. This process yielded a substantial number of rules, despite the short period during which it has been carried out. So the number of rules makes rigid checking of DB updates (manually entered by the specialists) for complete

consistency impossible. Furthermore it appeared that more general trends are much more difficult to formulate. There is a general belief that various heuristics could help in finding inaccuracies (e.g. elaboration on same rule, checking the geographic neighbourhood, updates by the same actor). Currently, the company investigates the applicability of a rule engine in their software, which enables the automatic tracing of anomalies. However the construction of rules is a manual process, driven by knowledge about known anomalies. Due to the high number of concrete rules, the rule base is flat and hardly manageable.

### 2.2.2 Data model and rule examples

The problem data basically consists of complex geographical features e.g. a restaurant or a water area. These complex features map to simple features which are points, lines, areas or some combination of these. The simple features can be seen as a combination of nodes, edges and faces in the spatial plane. A complex feature also has non-spatial attributes, which are organised in a hierarchical manner (e.g. composite address of a restaurant). The above concepts are illustrated in Figure 3.2. Furthermore, there is aggregation and multiple inheritance between feature types. Extra relationships (e.g. allowed traffic manoeuvre) between features can be imposed using associations. The most common feature type is the road element, which represents in fact a part of a road (not necessarily from a physical junction to another). It maps to a line feature and has functional road class, specifying road importance, amongst its attributes.

Some examples of regularity rules that were manually discovered by data engineers are listed below.

1. *At each roundabout, not all connected roads have the same single direction of traffic (all inwards or all outwards).* Violations to this rule occur when a traffic direction update has not been executed over all connected roads or after a typing error. In this chapter it will further be referenced to as the graveyard rule.

2. *A road element that is at both sides connected with road elements that have name X, has name X as well.* This rule is violated when an operator executes an incomplete name update. However in some exceptional situations, such a discontinuity is realistic. Both this and the previous rule link same-type features lying closely together and compare same-type non-spatial attributes of them.

3. *Each highway road element is connected at both sides.* Violations to this rule occur for example in the intersection of the highway with the border

between two regions that are operated by two different mobile specialists and are due to coordination problems. It links same-type features lying closely together.

### 2.2.3 Challenges

For this application, there are two major challenges with regard to the mining task of rule and/or anomaly discovery, regardless of the chosen approach.

First of all, the geographic database is gigantic in size, rich in structure and highly relational. So scalability is very important to apply the KDD process successfully. Provost and Kolluri [86] have made a survey of strategies for scaling up the kind of KDD process that employs inductive algorithms. However, most of the strategies stated can be easily generalised to other types of data mining and apply for our case. The large database size could be handled by various data partitioning strategies, with regard to data instances and/or data features. A rich data structure often yields a large pattern space. Strategic cropping of, as well as heuristic search through the pattern space could help. The relational data representation, which can be integrated in the KDD mining process, can be omitted by flattening the data or can be shifted towards the database management level.

Secondly, the spatial nature of the input data should be dealt with. Knowledge discovery in geographic databases is more difficult than traditional KDD. Shekhar et al. [99] even doubt the usefulness of traditional data mining techniques in this context because of data type complexity and intrinsic spatial relationships. According to [99], spatial data mining differs from classical data mining with regard to data input, statistical foundation, output patterns and computational process. Geographical data is composed of objects with spatial and non-spatial attributes. Because “everything is related to everything else but nearby things are more related than distant things”, materialisation of spatial relationships will be necessary and will determine the usefulness of the results to a certain extent. However, finding irregularities in our geographical dataset is not restricted to a purely spatial data mining problem. Not all anomalies can be detected using a spatial framework (e.g. discovery of a wrong speed limit based on the road type).

## 2.3 Applicability of ILP to Quality Gate

### 2.3.1 Motivation

Inductive Logic Programming (ILP) is a relational data mining technique that integrates the relational representation of the input data in the mining process [31] and in which input data as well as output patterns have a relational representation: the language of logic programs (or similar). It involves the explicit construction of First-order Logic rules from inherent regularities and trends in the input data. ILP algorithms can be seen as a search in a space of hypotheses that describe or classify parts of the input datačitelavrac94book.

ILP naturally comes as a candidate technique to apply to Quality Gate because automatic rule induction embodies the process of discovering (ir)regularities and trends in the geographical data, using an explicit representation that allows to control and loop up new occurrences of anomalies in the data and offers opportunities to be integrated in the company's business process. Because the geographical data is highly relational, ILP is preferable over propositional learning systems.

### 2.3.2 ILP subdomains

Mainly, ILP systems could be categorised into systems for predictive and for descriptive induction [24].

Predictive data mining is described as a discipline in which a hypothesis is induced that correctly classifies some given positive and negative examples (observations). So the outcome hypotheses of predictive ILP will be used for classification and prediction. Of course, this representation of classifiers could be extended to classification into any finite number of classes. Some predictive ILP systems induce LP rules, others generate decision trees that can be translated to small logical programs using cuts [24] (or using negation as failure). Predictive data mining has the longest tradition. Historically, predictive ILP systems are divided into empirical and interactive ILP (respectively EILP and IILP) systems, which have different properties, as described in [66].

The aim of descriptive data mining is to find regularities within a given set of unclassified examples [24]. In particular, as many usable data characteristics as possible are collected to build a sort of most specific hypothesis that describes the entire set of examples. Note that predictive induction can be handled more efficiently (lower complexity) than descriptive induction.

### 2.3.3 Quality Gate in relation to ILP subdomains

Predictive ILP systems, and specifically EILP systems, are interesting to learn rules driven by known anomalies. Anomalies are negative examples, other data is positive (or the other way around). The outcome hypotheses can be used to classify new data into the category of anomalies or correct data. However, there can be a problem with the significance of known anomalies compared to the number of unknown anomalies that are presented to the system as positive examples. It is not possible to learn rules about regularities using EILP without the guidance by known anomalies, because a generation of positive examples using the closed world assumption (CWA) does not work here.

EILP systems are commonly non-interactive 'from scratch' learners, which enable the assumed automatic rule generation. They need a large example set, which is no problem for our application. The fact that they are single predicate batch learners requires a higher-level strategy to support scalability.

At first glance, IILP systems are interesting candidate systems for Quality Gate because of their incremental behaviour. However they are inappropriate for the intended fully automatic generation of rules due to the fact that they rather revise than construct theories and that they count on external parties to judge or review newly generated examples or hypotheses. Nevertheless a nice support tool for domain experts who have low I.T. affinity can be developed using IILP systems. It provides the user with a GUI that shows some geographic situations that should be approved or disapproved or that should be commented. Like in a Mastermind game, the system is challenged to find the solution by asking as few questions as possible.

DILP systems are interesting to learn rules that describe intrinsic logic and trends in the geographic database. It is a form of unsupervised learning. The outcome consists of sets of all possible hypotheses that satisfy the constraint set. A much larger part of the hypothesis space is searched, because DILP is not guided by positive and negative examples and because of the result diversity. This makes DILP algorithms intrinsically less performant. However, DILP is the most suitable technique to apply to our problem, because there is no knowledge about anomalies needed in advance. Therefore, this technique is applied in the experiment discussed in the next sections.

## 2.4 Case study

### 2.4.1 Our approach based on WARMR

Stolle et al. [104] offer a formal generic definition on the descriptive ILP setting: the aim of DILP is to find a set of clauses $Th(cons, E, L_h)$ in a hypothesis language $L_h$ that hold over a set of interpretations E and satisfy a predefined conjunction of constraints $cons(h, E)$. The background knowledge B is used during hypothesis construction and embodies prior knowledge about the learning problem. A DILP algorithm typically uses one specific conjunction of constraints. A non-exclusive list of possible constraints is given in [104]. The definition adopts the partition of the input data set into interpretations to scale up the algorithm (as used in [24] and [9]).

WARMR is a DILP algorithm that finds patterns satisfying the constraint $freq(covers; E) > t$, as described by Stolle et al. [104], which means that more than t elements of the set of interpretations E should be covered by the pattern. WARMR involves a level-wise discovery of frequent datalog patterns, which consists of a conjunction of function-free terms [71]. Such a pattern is called a *frequent query* since the pattern must be executed as a query on $B \bigcup E$ to verify whether it covers the interpretation E. The *support* of a frequent query is the proportion of interpretations that are covered by a frequent query to the total number of interpretations. The WARMR algorithm enumerates only those frequent queries that have a support above a specified value. The discovery process starts by the enumeration of frequent queries consisting of one term (i.e. length 1), satisfying the hypothesis language $L_h$. In the next steps, the frequent queries of length $n + 1$ are generated by the extension of the frequent queries of length $n$, until a maximum pattern length is reached. Frequent queries are commonly post-processed into if-then rules or *query extensions*. This post-processing implies the combination of a pattern 'X' and a pattern 'X and Y' into the rule 'if X then Y'. The *confidence* of this query extension is the support of the pattern 'X and Y' divided by the support of the pattern 'X'. Its support equals the support of the pattern 'X and Y'. More details on the algorithm can be found in Dehaspe [26].

In this experiment, WARMR will be used to construct query extensions that describe regularities in the input data. The approach is outlined in Figure 2.2. As an example, the top left subfigure shows a map of connected road elements of which one allowed traffic direction is anomalous and creates a so-called inverse graveyard in node $a$. In a next step, the map is partitioned into interpretations by grouping road elements around the nodes (indicated by $E$). The data is translated into a Prolog format, in which separate predicates

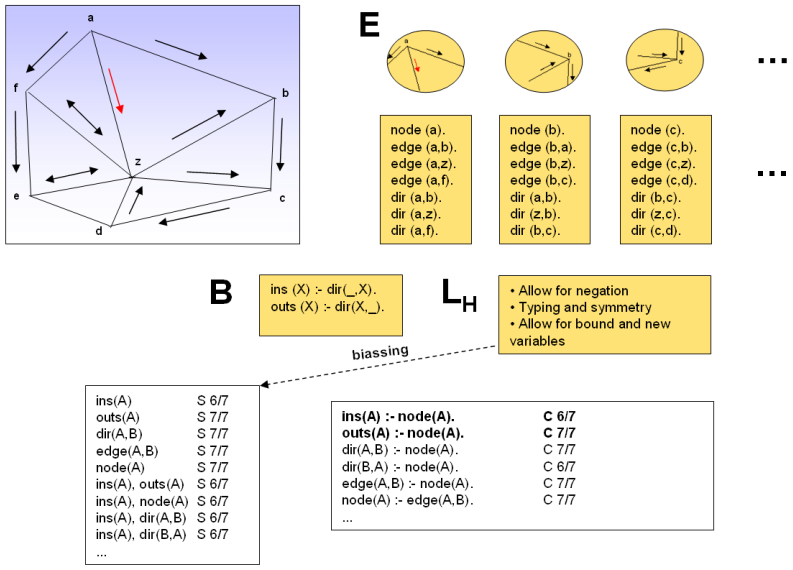



Figure 2.2: Applying WARMR to geographical data: general approach

are used to declare the existence of features and to define properties of these features. The hypothesis language is defined (indicated by $L_H$). Additionally, background knowledge $B$, which is used during rule construction, can be defined. Next, the WARMR algorithm is run to construct frequent queries (bottom left subfigure). During postprocessing, query extensions are constructed (bottom right subfigure). In a next stage, these rules are presented to human users and can be accepted or rejected. Query extensions with high but not 100% confidence are of particular interest to trace according violations. Notice that the coverage constraint employed by WARMR is existentially quantified within the interpretations. When for example interpretations symbolise separate villages containing roads, the rule *large(A) :- road(A)* does not imply that all roads in the villages are large, but that there are large roads in a village if there are roads.

The anomalies we mine can be defined as interpretations for which $B$ is true and $H$ is false given a rule $H$ :- $B$ with high confidence and support above a certain threshold. In other words, taking the WARMR coverage constraint into account, we are looking for interpretations e, given a clause of the form $h$ :- $b_1$ , ... , $b_m$ for which $\exists\theta : \{b_1\theta, ..., b_n\theta\} \subseteq e \Rightarrow \neg\exists\sigma : \{b_1\sigma, ..., b_n\sigma, h\sigma\} \subseteq e$ amongst a substantially larger group of interpretations e for which $\exists\theta : \{b_1\theta, ..., b_n\theta\} \subseteq$

$e \Rightarrow \exists\sigma : \{b_1\sigma, ..., b_n\sigma, h\sigma\} \subseteq e$, in which $\theta$ and $\sigma$ represent substitutions.

### 2.4.2 Data preparation and modelling: the Beggen data

The aim of this experiment is to evaluate the types of rules that can be induced from a dataset of realistic complexity and containing imperfect data. It is a subgoal to verify whether the graveyard rule is a feasible outcome of the WARMR approach. The dataset of the village of Beggen, Luxemburg, is small-sized reference test set provided by the company. Since this set did not completely comply with these goals, we applied two minor modifications to this data. A small part of the junctions was randomly left out resulting in imperfect data. Randomly generated allowed traffic flows were added because the Beggen data set did not contain a significant number of single flows supporting the graveyard rule(s). The experiment focusses on the induction of rules that relate one or more geographic objects and mainly non-spatial attributes of these objects, optionally using the spatial relationship of immediate proximity. We restricted ourselves to junctions (incl. geometry) and road elements (incl. names, geometry and importance). The input data was partitioned into interpretations by taking the immediate neighbourhood of each original junction. This resulted in an input data set of 68 interpretations described in Prolog format, using a separate predicates to announce the existence (by ID) and geometry of junctions and road elements and to define properties. The background knowledge only consists of predicates that are negations of predicates used in the knowledge base.

The most important WARMR parameters are: minimum allowed support of frequent queries (0.05), minimum allowed confidence of resulting rules (0.7) and search depth (5 literals). The language bias is also specified at this point. This is a set of constraints determining the hypothesis language. Type constraints are specified for the predicates described above and for their negated versions, specified in the background knowledge. Mode constraints allow for all predicate parameters to introduce new variables (+), to bind to already introduced variables (-) and to introduce new variables that are strictly different from a previously introduced same type variable (\). Functional road classes are allowed to be formulated as constants, and (not_)junction predicates can only appear once in the hypotheses because each interpretation contains at most one junction. Occurrence constraints ensure that road element flow predicates on the same variable only appear once in the hypotheses.

The WARMR run took about 15 minutes on a standard PC and, for the settings given, 39 696 frequent queries were generated (maximum length 5), out of which 13 584 query extensions were generated.

### 2.4.3 Results

About 30 interesting rules were manually selected from the large set of query extensions. Three classes of interesting regularity rules (query extensions) are listed below, together with some example rules. The classification is rather intuitive.

1. **Rules on completeness.** A first group of example rules are rules that state a simple (co)existence of one type of objects and/or its properties in the interpretations e.g. *Interpretations containing a class 1 road description, contain a name definition for that road element (confidence: 100%; support: 30.8%).* Note that, when the rule concerns properties of the same feature, the use of one interpretation per feature would have been more appropriate. Other rules in this series are rather diagnostic e.g. *Interpretations containing a road element functional road class description but without name definition for the road element involved, contain a class 7 road element (confidence: 100%; support: 7.3%).*

2. **Rules on flows.** This type of rule links the existence of allowed traffic flows within the interpretations. The graveyard rule was (re)discovered in two parts, because in our system only one literal is allowed in the consequence of an implication: *Interpretations not containing any 'both direction' road elements contain a flow from centre road element (confidence: 96.5%; support: 41.1%).* and *Interpretations not containing any 'both direction' road elements contain a flow to centre road element (confidence: 89.6%; support: 38.2%).*

3. **Rules on interpretation statistics and continuity.** This type of rule relates the existence of distinct objects and/or properties within an interpretation. Some rules can only be used for statistics e.g. *Interpretations containing a road element, contain another road element with different ID (confidence: 80.8%; support: 80.8%).* So 80.8% of the interpretations contain more than one road element. Others refer to laws on the continuity of road properties through interpretations e.g. *Interpretations containing a class 1 and a class 4 description, contain a class 1 description for a strictly different road element (confidence: 100%; support: 5.8%).* Note that most of the violations against these rules could be avoided by the use of border predicates, depicting the geographic objects located at the border of Beggen.

The outcome of this experiment is a giant number of hypotheses, which is not manageable to present to a human reader nor suitable to use immediately for anomaly detection. In [74] some suggestions are listed to reduce the number of

rules drastically either by specifying constraints in order to limit the number of rules generated or by means to filter the rules obtained.

### 2.4.4 Tracing anomalies and rule export

The case study has been extended by two modules to improve the usability and to enable further validation by experts.

A first module allows the end user to look up violating interpretations for each generated rule that has a confidence lower than 100%. This is realised by a Prolog interpreter, for which the background knowledge was asserted. If a given rule *H :- B* is looked up, for each interpretation I, sequentially, the facts of I are asserted; if the query B succeeds and the query B,H does not, the interpretation is added to the set of violating interpretations and the facts of I are retracted. The violations can be easily visualised in a spread sheet version of the Beggen map, because the interpretations were given coordinate names.

A second module is a rule export module. It translates the outcome rules into the XML format used by the company. During rule translation, the whole context in which the experiment was set up should be integrated in the outcome rules. For example, the original Prolog rule[1] *roadelement_name(C,B), not(=(C,A)) :- true, roadelement_name(A,B)* is, making abstraction of the syntax and data model details, translated into

*Check for each junction: if there exists a simpleorderarea8 containing the junction that has the officialname Beggen, and there exists a roadelement A that touches the junction and has an officialname, then there exists a roadelement C that touches the junction and has an officialname and the officialname of C equals the officialname of A and the ID of C is different from the ID of A.*

A first component that is reflected in the global rule structure is the WARMR constraint conjunction. Furthermore, the geographic area (Beggen) for which the rule applies should be adopted. Also the way the interpretations were constructed has its impact on the main root predicate of the rule and on the relation that is declared between the root feature and new geographic features in the rule body. Furthermore, cardinalities in the data model influence the lower-level formulations (not in the example; e.g. “there exists a property vs. has as property”). Also implicit Prolog bindings should be taken into account and other Prolog elements such as negation and equality should be translated correctly.

[1] The violations of this rule of low confidence do not refer to actual errors in the data. The example was solely selected to illustrate the translation mechanism.

## 2.5 Discussion

The ultimate aim of this research is to build an operational component that induces a confined set of readable rules that hold over the geographical data in a semi-automatic manner. It must be a generic component that mines the relations between given object and property types based on a given geographic region. The experiment showed us a partially successful application of a descriptive ILP algorithm. However, still a lot of problems need to be solved in order to build the intended generic component.

### 2.5.1 A reflection on the experiment

DILP goes clearly beyond intra-feature learning and is suitable to discover relationships among geographic features and attributes. The case study ended up with a rule overhead, for which we proposed suggestions to reduce the number of rules. The data representation of LP requires a serious preprocessing effort but enables an easy representation of outcome patterns and integration of background knowledge. Most of the available DILP systems require a considerable effort for the developer to get acquainted with and are rarely robust.

The resulting rules of the WARMR algorithm satisfy a constraint with existential quantification. This existential quantification yields a substantial performance improvement, but constrains the validity of the induced rules to the level of the partitioning chosen. For existential quantification, it seems preferable to omit all the rules that do not strictly correspond with the partitioning intentions. On the other hand, inducing rules with universal quantification is much less performant but the intended range of rules is much larger for one single experiment. But existential quantification requires more effort in the planning of the interpretation partitioning and in the formulation of hypothesis restrictions. In both situations results become more precise when a border predicate is used to indicate the interpretation's boundaries.

The use of negation as failure (not-literals) in the induced hypotheses caused some problems. Some algorithmic problems were encountered, because bound not-literals are order-dependent within the hypothesis. When introduced as a first part of the binding or unbound, the negated literal refers to existence within the interpretation; otherwise it refers to a failing property of an already introduced variable. In fact, for this research, both usages are interesting. Note that the use of multiple disjunct literals in the head yields the same logical expressiveness as the use of negated literals. Each disjunction in the head can be expressed as single literal headed rules using negation in the body. The use

of not-literals provides more information, whereas the use of disjunctions in the head yields a smaller hypothesis space.

The rule export module contributed to the interpretability of the resulting rules for end users, who are domain specialists but not necessarily computer scientists. The next question considers whether sets of rules can be logically combined to improve the inventory and usability of rules.

### 2.5.2 Towards a larger scale application of the experiment

At this point it is clear that it is not possible to set up one big ILP experiment that mines for all types of rules, nor one that mines for all object and property types, nor one that takes a whole country as input. So a more general strategy in planning series of ILP experiments is required. Such strategy should mainly manage input space and pattern space scalability.

A first requirement is a more generic representation of objects and properties types within the DILP experiment to mine for any object and property types of choice. This could be done by the integration of the metadata in the predicate scheme construction during the data formatting task. Measures should be taken with regard to the language bias to avoid that meta-information is mined. In fact this is a first step to integrate the existing concept hierarchies more in the search process.

Scalability of the input space is a minor problem in the organisation of series of ILP experiments. Thanks to the learning from interpretations setting, WARMR's time complexity is linear with regard to the interpretation set size for an individual experiment. Some relational data mining systems also have random sampling of input data (interpretations) as an option integrated in the search process. The necessity of presenting large geographic regions to a single ILP system depends on the support of the candidate rules in a dataset. The graveyard rule, for example, could have enough support in a large city. Maybe further checking in rural areas would not even increase its support, because of the lack of single traffic flow roads. Therefore, a possible solution is input data sampling during the data selection task. This sampling can be random, but in case of geographical data it might be better to select geographic regions for which certain conditions hold, such as variance or the presence of selected object types.

Pattern space scalability is the most delicate issue of a DILP application, because the size of the hypothesis space is exponential with regard to the number of allowed hypothesis predicates for an individual experiment. DILP systems such as ACE enable query sampling. It implies that not all possible frequent queries

are generated and executed; so it results in missing rules but also in a large reduction in execution time. However each form of knowledge about probable and improbable object type, property type and advanced features combinations is welcome. In this manner a series of smaller DILP experiments can be set up so that the global size of pattern spaces decreases enormously. Probably the concept hierarchy could help to crop useless type combinations. Another option is to organise the type combinations in relation to real-world map formats: low-scale national map, regional tourist map, map for geologists. This increases the probability to discover realistic relationships between object and property types.

## 2.6 Conclusion

The purpose of this research is to study the applicability of (D)ILP to discover anomalies in large geographical databases, by means of a case study on a realistic data sample. ILP appeared to be an interesting mining method for knowledge discovery in geographical databases because it enables the direct induction of usable first-order logic rules and copes with the relational nature of the input data. When rule learning were driven by known anomalies, which is not the case here, EILP systems would be applicable. In the case of unsupervised learning, DILP systems are used to describe intrinsic logic in the geographic database. In the latter case, it is not possible to build and consider a magna carta of regularities that hold over the input data, but it is worth to consider almost complete rules and look up their violations afterwards.

The experiment shows the intrinsic value of DILP. However, additional fine-tuning and optimisation are required before the induction of valuable knowledge can be automatised and taken to practical use. In order to apply the experiment on a larger scale, a more generic infrastructure is needed to support the automatic data extraction, preparation, modelling and evaluation. The next chapter tackles these issues by the introduction of a generic tool which is adaptive to the model of the input data.

With regard to scaling up the application, further research is needed that focuses on algorithm and representation design techniques to optimise the search. Until now, regularities were searched for in order to deduce irregularities afterwards. So a first approach to scale up is to integrate a direct search for irregularities within the process. A second opportunity is to embody the hierarchical structure of the data in the mining process. The geographical data can be represented in function of a hierarchy ranging from fine-grained up to coarse grained descriptors and the search process can be organised correspondingly. A third opportunity

is to take advantage of the spatial nature of the input data by modelling it in a graph format. In our case, graph edges could for example represent object closeness or even object similarity.

# Chapter 3

# Outlier detection in relational data: a case study in geographic information systems

Geographic information systems are commonly used for a variety of purposes. Many of them make use of a large database of geographical data, the correctness of which strongly influences the reliability of the system. In this chapter, we present an approach to quality maintenance that is based on automatic discovery of non-perfect regularities in the data. The underlying idea is that exceptions to these regularities ('outliers') are considered probable errors in the data, to be investigated by a human expert. A case study shows how the tool can be used for extracting valuable knowledge about outliers in real-world geographical data, in an adaptive manner to the evolving data model supporting it. While the tool aims specifically at geographic information systems, the underlying approach is more broadly applicable for quality maintenance in data-rich intelligent systems.

The research in this chapter is part of an R&D project funded by IWT (050730). I would like to thank Hendrik Blockeel (KU Leuven) for the interesting suggestions, and Gert Vervaet, Frank Maes, Denis Philips and Dieter Verhofstadt (Tele Atlas) for the repeated provision of data samples and the constructive feedback on the rule miner prototype. Celine Vens, who co-authored this chapter, is a postdoctoral fellow of the Research Foundation Flanders (FWO-Vlaanderen). The work in this chapter has been published as *Maervoet, J., Vens, C., Vanden*

*Berghe, G., Blockeel, H., De Causmaecker, P. (2012). Outlier detection in relational data: a case study in geographical information systems. Expert Systems with Applications, 39 (5), art.nr. ESWA7039, 4718-4728.*

## 3.1 Introduction

Outliers in a data set are commonly defined as *individuals* that are substantially different from the rest of the data. Such irregularities can indicate an error in the data, or abnormal behaviour of the underlying system. On the other hand, outliers can exhibit exceptional situations in the data, which are not fundamentally erroneous neither abnormal. In research areas such as machine learning and statistics, a great diversity of algorithms for outlier detection have been proposed in the last years [13, 57, 2, 15]. Most of them refer to a statistical deviation of the outlier values from the rest of the data set. However, a lot of contemporary applications of outlier detection such as fraud and network intrusion detection, have a relational character. The data consist of several interrelated data types, implying that the concept of outlier detection can be seen in a broader perspective. Besides the detection of deviating values for a specific variable, it is also possible to look for deviating structures in the relational data.

In this chapter, a case study of relational outlier detection on geographical data is presented. It concerns learning anomalies in the core database of the geographic content provider Tele Atlas. This company possesses a large amount of geographical road data, collected from different sources. Irregularities, e.g. a wrong speed restriction, creep in due to human mistakes or inconsistencies between different sources. Therefore, a quality maintenance system has been set up by the company enabling data engineers to manually formulate rules to which the data should conform and providing infrastructure to trace violations against these rules in a brute-force manner. An example of such a rule is "A road segment adjacent to a primary school has always a speed restriction of 30". More information about the problem context is described in Chapter 2. In the present chapter, we apply a relational frequent pattern miner to discover such rules automatically from the data. Exceptions to these rules will be considered probable erroneous data, to be presented to a human expert for evaluation.

The chapter is structured as follows. Section 3.2 presents some related work in the domains of relational outlier detection and spatial data mining. Section 3.3 thoroughly describes the geographic data quality problem and the relational outlier detection approach to it. The actual case study is presented in more detail in Section 3.4. Section 3.5 reports some regularity rules and corresponding

outliers found by the system. Finally, we indicate some directions for future work in Section 3.6 and conclude in Section 3.7.

## 3.2 Related work

### 3.2.1 Outlier detection

Given an input data set, an **outlier** is an instance or a set of instances[1] that show(s) exceptional behaviour compared to the rest of the input data set or to a local context within the input data set. **Outlier detection** is the non-trivial process of extracting a set of previously unknown anomalies from data. It is a form of data mining.

A first dimension categorises outlier detection approaches according to the type of learning. Lazarevic et al. [67] distinguish between:

- **Supervised outlier detection.** Both the outliers and regularities from the input data set are labelled. In this case, the problem can be reduced to classification.
- **Semi-supervised outlier detection.** Some examples of anomalies and/or regularities from the input data set are given. It can be applied in interactive learning systems. Zhu et al. [118] e.g. predict all the outliers from the input data set based on an outlier sample set indicated by the end user.
- **Unsupervised outlier detection.** This type of outlier detection assumes unlabeled input data. It involves fitting one or more models over the input data and identifying the model deviations as outliers.

| | **Indirect description** | **Direct description** |
|---|---|---|
| **Statistical outliers** | Clustering (distance or density based)<br>Probability distribution (histogram, Gaussian) | Nearest neighbour (distance or density based) |
| **Relational outliers** | Frequent pattern discovery | Anomaly pattern discovery |

Table 3.1: Unsupervised outlier detection.

[1] In the latter case, individual set members are not anomalous.

Table 3.1 shows a classification of unsupervised outlier detection systems. These systems can be classified into statistical and relation outlier detection and into detection by direct and indirect description.

**Statistical outlier detection.** This type of system looks for outliers with a statistical deviation from the rest of the data, assuming one global model that distinguishes the outliers from the regular data. A common technique applies clustering: the data is clustered, and the elements that do not belong to any clusters are outliers. Other methods use density [13] and/or proximity analysis [57, 89]. Aggarwal and Yu [2] introduce evolutionary algorithms for identifying outliers in data with a high number of dimensions. Frank et al. [35] mine for spatial regional outliers. These are neighbourhoods of anomalous objects that maximise the non-spatial attribute value deviation between the object and its neighbouring objects.

**Relational outlier detection.** Many contemporary outlier applications are relational. In the context of network security, for example, there is a high interest in so-called anomaly detection. This is the detection of deviant behaviour in network traffic that possibly indicates an attack. Caruso and Malerba [15] propose an adaptive model for network traffic. If a new network connection deviates substantially from the model, the system examines whether it concerns an outlier, or a legal connection, whereupon the model is adapted.
In relational outlier detection, the input data is composed of several interrelated data types and so the outliers have a relational character too. Often, it assumes multiple models that explain why an outlier differs from the rest of the data. In [4], a theory of normal behaviour is modelled using a formal knowledge representation language (first-order logic). Both outliers and so-called witness sets are searched for. Outliers are entities that are inconsistent with the given background knowledge. Corresponding witness sets describe the causes behind the outliers in the data.
Relational outlier detection is a form of relational data mining, a research area that has gained a lot of interest during the last years. A large amount of the research is carried out in the context of inductive logic programming (ILP) [66]. The data, as well as the discovered patterns and the background knowledge, are represented as logic programs. The major part of research on ILP (and on relational data mining in general) has been carried out on supervised learning. Less research has been performed on unsupervised learning, in which a set of hypotheses that describe the whole set of facts as accurately as possible are learnt. Relational clustering [90], finding frequent patterns in first-order logic [27] and clausal discovery [25] belong to the latter category.

**Discovery by direct description.** This means that patterns describing exceptional situations are looked up directly. For instance, Laros [64] looks for a substring, as short as possible, that appears exactly once in a set of strings. With

regard to relational outliers, several definitions and corresponding algorithms for anomalous pattern discovery can be found: sporadic rules [58, 59] (rules with low support but high confidence), minimal infrequent itemsets [45] and unexpected rules [84] (with a support between two thresholds). Exception rules [107] refer to the extension of the premise of a 'common sense rule', refuting the consequence of that rule. Anomalous association rules [8] refer to association rules for which anomalous itemsets exist that always contradict the rule.

**Discovery by indirect description.** Instead of looking for patterns that describe the exceptions directly, the complementary problem can be examined as well. It involves looking for regularities, followed by the identification of data that does not comply with those regularities. K-Means clustering of network traffic data followed by the identification of traffic anomalies [81] is an example of this category. Discovery by indirect description allowed us to apply state of the art techniques from the domain of frequent pattern mining.

### 3.2.2 Spatial rule mining

Spatial rule mining is a common machine learning approach to spatial data mining (SDM), which aims at extracting useful or interesting patterns from spatial databases. Shekhar et al. [99] indicate that the data input, statistical foundation, output patterns and computational process are different for SDM. Zeitouni [116] identified several generic SDM tasks, and associated existing methods with these tasks. With regard to rule learning, we can distinguish 4 types of approaches:

- **Characteristic rules.** This type of rules describes characteristic object and neighbourhood properties of a set of spatial objects in the database [32]. It is a form of summarisation.
- **Classification rules.** This form of supervised classification involves the discovery of a set of rules (often represented as decision trees) comparing the object and neighbourhood properties of a set of spatial objects of choice, called the target class, to one or more contrasting classes. For instance, Ceci and Appice [16] learn geographic impact factors for several rent prize categories from census data. Frank et al. [34] propose a Voronoi-based framework to integrate spatial relationships in the search process.
- **Association rules.** Spatial association rule mining is the identification of frequently occurring spatial-related patterns in a set of data items in a spatial database [47]. It can be categorised as spatial data dependencies mining.

- **Trend rules.** Basically, trend rules describe patterns of change of one or more non-spatial attributes of objects or objects in their neighbourhood [32].

**Spatial association rule mining.** Koperski and Han [60] defined spatial association rules (SAR) as association rules with at least one spatial predicate in the antecedents or consequent. Such a spatial predicate could refer to topological relationships, orientation and ordering and contain distance information. The spatial rule mining algorithm employs refinement in a hierarchy of topological relations i.e. starting from approximate spatial computation.

**Concept hierarchy refinement.** Spatial multi-level association rule mining extends the approach above by refinement in a (spatial or attribute) concept hierarchy. An example of *spatial hierarchy* is the refinement of a country into one or more provinces. An example of *conceptual hierarchy of attributes* is the refinement of areas into rural and urban areas.
SPADA (Spatial Pattern Discovery Algorithm) is a system for spatial association rule mining, in which the rules have a Datalog representation. It uses refinement through a concept hierarchy of objects. SPADA is used by Malerba et al. [77] and Appice et al. [5] for analysing socio-economic issues in census data in order to improve transport planning. Lisi and Malerba [71] improved this system by the design of the hybrid language AL-log, which yields a unified treatment for both relational and structural data features.

## 3.3 Problem description

### 3.3.1 System analysis

The company Tele Atlas collects geographical information from several sources, such as aerial photographs and mobile mapping. It provides the geographical data for companies active in the areas of car navigation systems, geographic information systems and location-based services. From these application areas, the company is facing an ever increasing demand for geographical data quality. It manages a large central database, which is subject to continuous updates, originating from core data collection and processing using high quality standards. There are two strategies by which the quality of the geographical data in the database can be maintained and improved:

- by processing the navigation logs of and explicit update requests by the end user community

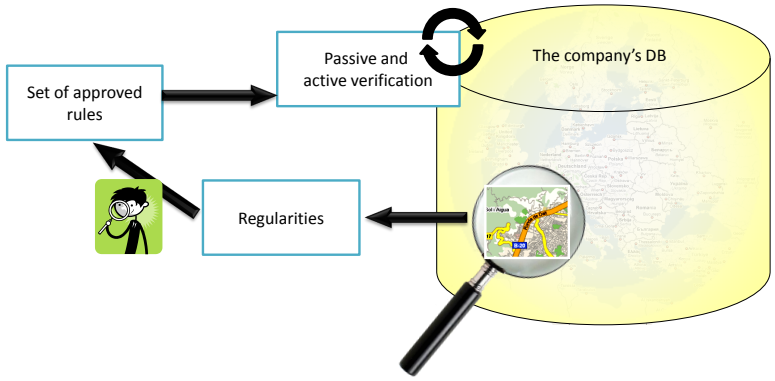



Figure 3.1: Identification of regularities and anomalies within the quality maintenance business process.

- by making quality domain knowledge explicit and verifying it against the data.

In line of the latter strategy, Tele Atlas has set up an infrastructure that allows manually building quality rules and verifying data against these rules. Passive verification implies a check of each update against a limited set of rules. Active verification is done by separate processes, checking the rest of the rules against the whole database. In this chapter, we introduce a tool that automates building quality rules.

This tool is used by data engineers and extracts previously unknown relations that are present in the data. It supports the use cases below:

- First of all, the user selects a data sample and formulates a question e.g. "How do speed restrictions of a road element (i.e. elementary piece of any road) relate to adjacent points of interest (POIs)?"
- Within a reasonable amount of time, the user receives direct and complete answers to the question regarding the selected data sample, in the form of rules, together with their statistical relevance.
- The user is able to trace the violations (outliers) against these rules and to visualise them.
- Guided by a rule's outliers, the user decides whether to accept the rule in the quality maintenance system, by exporting it, or not. Also very similar

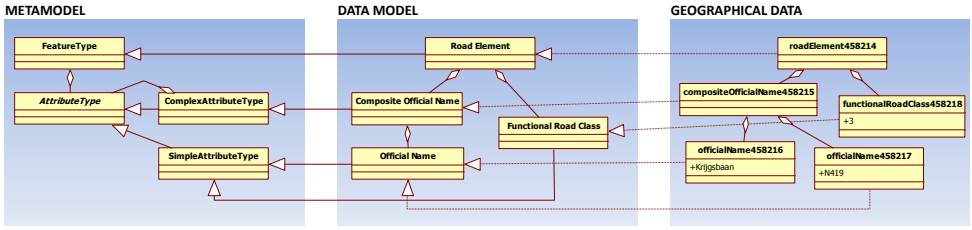



Figure 3.2: An excerpt of the metamodel, the data model, and the data (UML class diagram).

rules without violations can be considered for acceptance. Note that rules with outliers indicating exceptional true situations should not be approved by the user. After approval, the rule can be used for active and passive verification, in order to discover outliers regarding the complete database.

These functionalities clearly require a system for **outlier detection by indirect description**. Figure 3.1 shows its impact on the quality maintenance business process. Data sampling is required to cope with the large dimensions of the database. The data engineers should match data samples to questions. The sample size should be significantly large in order to avoid overfitting.

### 3.3.2 The dynamic data model

Besides the geographical data, the data model has a dynamic nature too. The data model changes, for example, when the engineers decide to adopt a new type of POI, or when, entering a new country, the address interpolation representation by integers does not apply any more. In order to design a rule miner tool that copes with this dynamic data model, the metamodel is constructed. This is the model of the datamodel, which does not change over time. It will be used to design a rule language that is independent from the data model that is currently in use. Figure 3.2 conceptually shows the relationship between the metamodel, the data model, and the data.

We explain the most important concepts in the metamodel, their implementations in the data model, and the data itself:

- **Feature type.** The company's geographical data basically consists of features of a certain type, for example restaurants, water areas, junctions or road elements.

- **Simple and complex attribute type.** Each feature type is composed of a tree of attributes types, in which the internal nodes are complex attributes and the leaves are simple attributes, i.e. containing a value. A road element has, for instance, a functional road class, which is a value indicating the road importance (highway, secondary road,...) and a composite official name, of which multiple official names contain the name strings. The most important attribute is the **geometry**, which scales down to a point, a polyline, a polygon or a combination of these. The data model also defines spatial relationships such as overlap and distance.
- **Association type** (not in figure). An association type links several feature types by specific roles, e.g. a forbidden traffic manoeuvre between two road elements or connectivity between junctions and road elements.
- **Inheritance support** (not in figure). Furthermore, the metamodel supports association, feature and attribute type inheritance. For example, restaurants, junctions and schools inherit from the POI type. All types inherit the geometry attribute type from the base feature type.

The rules, generated by the tool, are expressed in terms of the data model and describe previously unknown relationships in the data. Note that a possible data model update requires a set of data transformations, which apply to the already discovered rules as well.

### 3.3.3 Rule and outlier type analysis

At this point, we go into more detail about the type of rules that the tool is expected to extract. We received a set of example rules in advance from the company, out of which 3 typical rules are listed in the second column of Table 3.2. The company expects their type is very similar to the type of rules the tool might discover.

According to the definitions by Koperski and Han [60], this rule set contains both spatial and non-spatial association rules. Table 3.2 shows a possible experiment scenario with regard to the system analysis functionalities for each of the example rules. The questions are examples of system inquiries by users that definitely result in a concise set of rules, of which the example rule is an unexpected member. 'Unexpected' means that data engineers who do not know the examples, are not able to predict the rule from the question. The anomaly descriptions state which kind of outliers, for each of the rules, the user expects to highlight during visualisation.
The table shows that

| Question | Example rule | Anomaly description |
|---|---|---|
| "How do road element speed restrictions relate to adjacent POIs?" | "A road element adjacent to a primary school has always a speed restriction of 30 km/h" | road elements adjacent to a school with a speed restriction different from 30 km/h |
| "How do the road element's attributes interrelate?" | "A road element with a speed restriction of 120 km/h, has always functional road class 1 (i.e. high road importance)" | a road element with speed restriction 120 and functional road class $> 1$ |
| "How does a roundabout relate to the attributes of its associated features?" | "Each roundabout has at least one connection-association with a road element with a traffic flow away from the roundabout." | a roundabout without connected road elements with a traffic flow away from the roundabout |

Table 3.2: Possible experiment scenarios for the example rule set

- each rule can be mapped to a question aiming at feature and attribute type relations, given a fixed spatial relation or association type.
- for each rule, the expected anomaly descriptions refer to spatial objects for which the fixed relation or association type holds, but the rule fails.[2]

This representation situates the problem as a **relational outlier detection** problem and requires the use of relational association rule mining techniques to look up regularities in a first phase. Moreover, omitting the refinement of spatial operators during the search process speeds up the search, compared to spatial rule mining.

### 3.3.4 The integration of a relational datamining technique

**Hypothesis language requirements.** Building an operational tool that is able to discover patterns in the complete data set, independent from the data model version in use, requires a uniform representation language in terms of the metamodel, in which the data and the rules can be expressed. Besides, the hypothesis language should enable the expression of aggregate ('has-a') relationships between features and attributes.

**The algorithm.** WARMR [26] is a relational datamining algorithm that induces association rules in datalog representation, which meets the above language requirements. It uses learning from interpretations [9], which is

[2]For instance, in example 1, it would not make sense that the outlier detection comes up with pairs of primary schools and road elements with speed restriction 30 km/h that are not adjacent.

typically used for description. This learning setting assumes that the input data is presented in the form of interpretations. These are database partitions that represent a set of relational states. A candidate hypothesis describes certain interpretation properties. It covers an interpretation if and only if the interpretation is a model for the hypothesis. The algorithm is linear in the number of interpretations.
First, WARMR executes a level-wise discovery of **frequent queries** that cover the given set of interpretations. Frequent queries are conjunctions of literals that fulfil the language bias provided by the user. The **language bias** consists of a set of constraints, which determine which frequent queries are searched for. The **support** of a frequent query is defined as the number of interpretations the query covers to the total number of interpretations. Level-wise discovery involves that, at each level, the frequent queries are specialized by extending them with each of the allowed literals, until a specified maximum number of levels (literals) is reached or until the support has decreased below a specified minimal support. Note that also background knowledge, in the form of rules, can be taken into account during interpretation coverage control.
Next, frequent queries are processed into **query extensions**. A query extension is a datalog clause of the form $h :- b_1, b_2, ..., b_m$, generated from the queries $b_1, b_2, ..., b_m$ and $b_1, b_2, ..., b_m, h$. The **confidence** of the query extension is defined as the support of the latter query to the support of the first. The support of the first and the latter query are said to be the **bodyfrequence** and the support of the query extension. The discovery of **outliers** to a query extension is trivial. Outliers are the interpretations that are covered by the first query but not by the latter.
Clare and King [21] treat the distribution of levelwise rule discovery algorithms such as WARMR. They describe Farmer, Worker and Merger processes to distribute frequent query support counts within equal amounts of interpretations over multiple machines.
In our case study, we use the WARMR implementation of the ACE Datamining System [11]. It implements a set of Inductive Logic Programming (ILP) algorithms, of which the efficiency has been improved by the query pack mechanism [10].

## 3.4 System design

### 3.4.1 Rationale

The system analysis in the previous section states that the user starts an experiment from a selected data sample, which is typically a geographic area

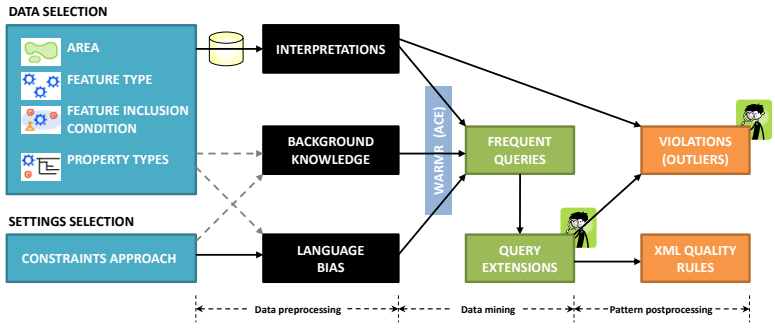



Figure 3.3: Design of the rule miner prototype.

and a question. The rule type analysis showed that rule mining with a relational approach, realised by WARMR, can formulate answers to these questions. This algorithm learns from interpretations. The user's question consists of 3 data selection items. The design choices for each of these items are motivated below.

- **Central feature type.** In the quality maintenance system of the company, a rule starts by definition by a universal quantification for the features of a specific type, as in "For all features of type x: ...". Strictly speaking, the rules produced by WARMR have a relative quantification over the interpretations, as in "For 99% of the interpretations: ...". However, this rule is adopted as a perfect rule (and thus universally quantified) by the quality maintenance system. We prefer to build interpretations for features in a geographic area and of one specific type. As a consequence, WARMR produces rules that conform to the specifications of the system. Therefore, the user has to select a feature type.

- **Feature inclusion condition.** One approach to including spatial information (e.g. distance) in the rules, would be to define a set of spatial relations in the background knowledge. This would result in a large number of spatial calculations (not any information will be processed and cached only once) during the knowledge discovery process, resulting in poor computational performance. A common technique for performance improvement in spatial data mining is the materialisation of spatial relationships, described by Shekhar et al. [99]. It involves that all necessary spatial calculations are executed during preprocessing and that the results are integrated in the input data. Therefore, we prefer the user to select a spatial relation, e.g. overlap, and this information is incorporated in the interpretations. This is realised by the inclusion of features for which

the relation holds. The same principle is applied for associations, not for performance but for uniformity reasons.

- **Attribute types.** The data engineer might not be interested in possible relations for each of the simple and complex attribute types involved. It should thus be possible to restrict the attributes types entered in the interpretations.

These data selection items are shown in Figure 3.3.
The next subsection details the design of a rule language that covers the example rules in Table 3.2. The following subsections discuss the data preprocessing, mining and postprocessing steps of the rule miner tool in Figure 3.3.

### 3.4.2 Generic rule language

In this section, a rule language is defined in terms of the metamodel, in order to support data model evolution. In Table 3.3, we define and illustrate a set of primitive functions.

The rules that are generated will test for the existence of related features, certain attributes, or certain attribute values. The rule language consists of the components defined in Table 3.4. $Feat\_rel(CF, F)$ is a boolean function,

| **Definitions** | **Examples** |
|---|---|
| $type(DataElement)$<br>returns the specific data model type of a feature, an attribute or an association. | $type(feat4497) =$ "$Road$"<br>$type(attr4498) =$ "$Address$"<br>$type(ass4499) =$ "$Forbidden\ Turn$" |
| $value(SimpleAttribute)$<br>returns the assigned value of a simple attribute. | $value(attr4498) =$ "$Elm\ Park$" |
| $has(DataElement1, DataElement2)$<br>returns *true* if the first element contains the second one. According to the metamodel, only features and complex attributes can contain other attributes. Associations contain features. | $has(feat4497, attr4498) = true$<br>$has(ass4499, feat4497) = true$ |
| $spat\_dist(Feature1, Feature2)$<br>returns the spatial distance between the "Geometry" attributes of the 2 features. Returns 0 if both geometries overlap. | $spat\_dist(feat4496, feat4497) = 20$ |

Table 3.3: Primitive function definitions

which returns *true* if the relationship between the features $CF$ and $F$ holds.

| No. | Definitions |
|---|---|
| 1 | $foreach\_feature(feature\ \underline{CF}, type\ T)$<br>$\forall\ CF : type(CF) = T$ |
| 2 | $feature\_exists(feature\ \underline{F}, type\ T, boolfunction\ Feat\_rel, feature\ \mathbf{CF})$<br>$\exists\ F : type(F) = T \wedge Feat\_rel(CF, F)$ |
| 3 | $complex\_att\_exists(attribute\ \underline{A}, type\ T, element\ \mathbf{P})$<br>$\exists\ A : type(A) = T \wedge has(P, A)$ |
| 4 | $simple\_att\_exists(attribute\ \underline{A}, type\ T, element\ \mathbf{P}, value\ V)$<br>$\exists\ A, V : type(A) = T \wedge has(P, A) \wedge value(A) = V$ |

Table 3.4: Rule language components

Example functions for $Feat\_rel(CF, F)$, will be defined in Subsection 3.4.3. Let us assume that $adjacent50(CF, F)$ is $true$ when two features are less than 50 metres apart. The ground term
$feature\_exists(church53, \text{“}Church\text{”}, adjacent50, road54)$
means that feature $church53$ of type “Church” is adjacent to feature road54. The first rule in Table 3.2 is defined as

$$
\begin{aligned}
&foreach\_feature(A, \text{“}Road\ Element\text{”}) : \\
&\qquad feature\_exists(B, \text{“}School\text{”}, adjacent50, A), \\
&\qquad simple\_att\_exists(C, \text{“}Type\text{”}, B, \text{“}primary\text{”}), \\
&\qquad complex\_att\_exists(D, \text{“}Composite\ Speed\ Restriction\text{”}, A) \\
&\qquad \Rightarrow simple\_att\_exists(E, \text{“}Speed\ Restriction\text{”}, D, 30)
\end{aligned}
$$

given the model in which speed restrictions belong to a composite attribute and primary is a value for the attribute named “Type” contained by the “School” feature type.

### 3.4.3 Data preprocessing

**Interpretation generation** First, the user enters a selection of the geographical database partitions to be inspected, whereupon these partitions are loaded from the database. Next, the interpretations are generated from the loaded data. This generation consists of the following steps (present in Figure 3.3):

- The user chooses a central feature type of interest, around which the interpretations are built. For each instance of this central feature type in the data, an interpretation is constructed. This step determines the $T$ parameter in rule language component 1.

| Approach | $Feat\_rel(CF, F)$ | Example rule |
|---|---|---|
| **Inclusion by overlap** adds all features of some types of choice (in the set $ftypeset$) that overlap the central feature. | $spat\_dist(CF, F) = 0$ $\wedge\, (type(F) \in ftypeset)$ | Each "Service Area" overlaps at least one "Service Point". |
| **Inclusion by offset distance** adds all features of some types of choice that are situated an offset distance $d$ apart from the central feature. | $spat\_dist(CF, F) < d$ $\wedge\, (type(F) \in ftypeset)$ | Rule 1 in Table 3.2. |
| **Inclusion by association type** adds all features that are associated with the central feature type for some association types of choice (in the set $atypeset$). | $\exists\ A : type(A) \in atypeset$ $\wedge has(A, CF) \wedge has(A, F)$ | Each "Slip Road" is associated with a "Forbidden Turn". |

Table 3.5: Inclusion condition: three approaches

- By formulating inclusion conditions, the user is able to include other features in the interpretations that are somehow related to the central feature. This step both constrains the $T$ parameter and defines the $Feat\_rel$ function to be included in rule language component 2. Currently, 3 types of inclusion condition, shown in Table 3.5, are supported.
- The user has to indicate an attribute type subtree for each of the feature types involved. Only the information for these attribute types is recorded in the interpretations. This step constrains the $T$ parameters in rule language components 3 and 4.

Figure 3.4 shows the relationship between the geographical metamodel and the concepts of interpretation construction. The interpretations are generated into Prolog notation, based on the rule language components.

**Language bias and background knowledge generation** The language bias and background knowledge are partially fixed, partially generated in a semi-automated manner. The language bias ensures that recurring variables only bind parameters of the same type, as listed in Table 3.4. Note that features and attributes are both elements. The same table contains annotations that indicate how variables and constants are introduced in candidate rules. By default, the underlined terms will be replaced by new variables, the normal terms by constants and the bold terms by previously introduced variables.
There are some possible variations:

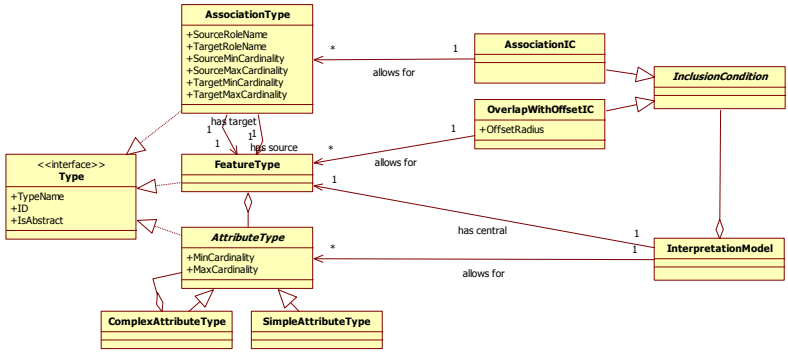



Figure 3.4: The geographical metamodel and its relation to interpretation construction (UML class diagram).

- The end user can select an alternative language bias and background knowledge pair, resulting in *data model mining*. An example rule is: "Each restaurant is overlapped by exactly one restaurant area." This setting only includes information about the existence of simple attributes instead of the values (cf. underlined parameter V of rule language component 4), and allows to mine for cardinalities. In this case, the background knowledge contains the definition of an isunique-predicate, indicating whether an attribute of a given type only occurs once for each parent attribute or feature. It is included in the language bias.

- The tool supports abstract feature and attribute types in the hypothesis language. An example rule is "Each service point (this is an abstract feature type) is adjacent to a road". In this case, the background knowledge contains the necessary rules to derive whether a feature or attribute type implements an abstract type. It is necessary to enumerate all possible abstract and non-abstract types in the language bias.

Table 3.6 shows a sample data flow during preprocessing.

### 3.4.4 Data mining and pattern postprocessing

**Data mining** The interpretations, background knowledge and language bias files are fed to the ACE Datamining System. Before mining, the user is asked a minimum support, a minimum confidence and a maximal rule length (i.e. maximal number of literals). In a level-wise manner, WARMR generates the

| | |
|---|---|
| **Raw data** | $type(f001) = type(f002) = type(f003) =$ "$Road\ Element$"<br>$type(f004) = type(f005) =$ "$School$"<br>$spat\_dist(f001, f004) = 8 \quad spat\_dist(f001, f005) = 66$<br>$spat\_dist(f002, f004) = 24 \quad spat\_dist(f002, f005) = 22$<br>$spat\_dist(f003, f004) = 16 \quad spat\_dist(f003, f005) = 44$<br>$spat\_dist(f004, f005) = 21$<br><br>$type(ass001) = type(ass002) =$ "$Connected\ Road\ Elements$"<br>$has(ass001, f001) = has(ass001, f003) = true$<br>$has(ass002, f002) = has(ass002, f003) = true$<br><br>$type(a001) = type(a002) = type(a003) =$ "$Comp.\ Speed\ Restriction$"<br>$type(a011) = type(a012) = type(a013) =$ "$Speed\ Restriction$"<br>$type(a004) = type(a005) =$ "$Type$"<br><br>$has(f001, a001) = has(f002, a002) = has(f003, a003) = true$<br>$has(a001, a011) = has(a002, a012) = has(a003, a013) = true$<br>$has(f004, a004) = has(f005, a005) = true$<br><br>$value(a011) = 30 \quad value(a004) =$ "$primary$"<br>$value(a012) = 50 \quad value(a005) =$ "$university$"<br>$value(a013) = 30$ |
| **Input settings** | - Central feature type: "Road Element"<br>- Inclusion by offset distance: all restaurants, schools and gas stations that are situated 50m apart from the central feature<br>- Attribute types: all attribute types for the feature types involved |
| **Interpretations** | %interpretation for "Road Element" f001<br>...<br><br>%interpretation for "Road Element" f002<br>$complex\_att\_exists(f002,$ "$Composite\ Speed\ Restriction$", $a002)$.<br>$simple\_att\_exists(a002,$ "$Speed\ Restriction$", $a012, 50)$.<br>$feature\_exists(f004,$ "$School$", $adjacent50, f002)$.<br>$simple\_att\_exists(a004,$ "$Type$", $f004,$ "$primary$").<br>$feature\_exists(f005,$ "$School$", $adjacent50, f002)$.<br>$simple\_att\_exists(a005,$ "$Type$", $f005,$ "$university$").<br><br>%interpretation for "Road Element" f003<br>... |

Table 3.6: Sample data flow during preprocessing for the school/road data set

frequent queries (above the minimum support), which are processed into query extensions (above the minimum confidence) afterwards. Some examples are given in the results section. These query extensions are presented to the end user and can be selected individually for outlier detection and rule export.

| | |
|---|---|
| **Language bias** | Default settings. |
| **Frequent queries** | $foreach\_feature(A, \text{“}RoadElement\text{”})$ :<br>$-\ complex\_att\_exists(B, \text{“}Comp.\ Speed\ Restriction\text{”}, A)$. Supp: 1<br>$-\ feature\_exists(B, \text{“}School\text{”}, adjacent50, A)$. Supp: 1<br>$-$ ...<br>$-\ feature\_exists(B, \text{“}School\text{”}, adjacent50, A)$,<br>$simple\_att\_exists(C, \text{“}Type\text{”}, B, \text{“}university\text{”})$,<br>$complex\_att\_exists(D, \text{“}Comp.\ Speed\ Restriction\text{”}, A)$. Supp: 0.67<br>$-\ feature\_exists(B, \text{“}School\text{”}, adjacent50, A)$,<br>$simple\_att\_exists(C, \text{“}Type\text{”}, B, \text{“}primary\text{”})$,<br>$complex\_att\_exists(D, \text{“}Comp.\ Speed\ Restriction\text{”}, A)$. Supp: 1<br>$-\ feature\_exists(B, \text{“}School\text{”}, adjacent50, A)$,<br>$simple\_att\_exists(C, \text{“}Type\text{”}, B, \text{“}primary\text{”})$,<br>$complex\_att\_exists(D, \text{“}Comp.\ Speed\ Restriction\text{”}, A)$,<br>$simple\_att\_exists(E, \text{“}Speed\ Restriction\text{”}, D, 30)$. Supp: 0.67 |
| **Query extensions** | ...<br>%query extension 20<br>$foreach\_feature(A, \text{“}Road\ Element\text{”})$ :<br>$feature\_exists(B, \text{“}School\text{”}, adjacent50, A)$,<br>$simple\_att\_exists(C, \text{“}Type\text{”}, B, \text{“}primary\text{”})$,<br>$complex\_att\_exists(D, \text{“}Composite\ Speed\ Restriction\text{”}, A)$<br>$\Rightarrow simple\_att\_exists(E, \text{“}Speed\ Restriction\text{”}, D, 30)$. Conf: 0.67 |
| **Outliers** | The outliers for query extension 20 are:<br>- interpretation for “Road Element" f002<br>Visualisation: $g002 : type(g002) = \text{“}Geometry\text{”} \wedge has(f002, g002)$. |

Table 3.7: Sample data flow during datamining and postprocessing for the school/road data set

**Outlier detection** Outliers for an individual rule are the interpretations for which the body of the rule holds, but the body extended by the head fails. Outlier detection involves the execution of these two queries on the logic program of each of the interpretations separately, each time extended by the background knowledge. The tool supports a generic visualisation of outlying interpretations on a geographical map. Interpretations always represent features that have a geometry, which scales down to a (set of) points, polylines or polygons.

**Rule export** The XML rule format used by the company is a semantical superset of the rule language defined in subsection 3.4.2. The rule export involves syntactical conversion, conversion to primitive functions and the removal of duplicate information. For example, the feature type set constraint in $Feat\_rel(FC, F)$ can be omitted, because rule language component 2 involves a feature type declaration. The rule export module allows the end user to export an accepted rule to the quality maintenance system, which uses the rule for active or passive verification.

Table 3.7 presents a sample data flow during the data mining and outlier detection steps, which is subsequent to the flow in Table 3.6.

## 3.5 Results

In this section, we present a set of example rules found by the system. We first present the outcome of two specific experiments, focussing on the query extensions that have ***almost*** 100% confidence. These rules are of particular interest, because they directly indicate possible outliers in the data sample. For each of the rules, expert feedback is given. Next, we present a sanity check, in which experiments are reconstructed for a set of rules that have been designed from specifications manually by the data engineers.

### 3.5.1 Experiment 1: discovering inter-feature relations

In a first experiment, we try to induce relationships between associated features of junctions. Therefore, we used following input settings:

- Geographical data set: northern Barcelona (consisting of 1404 junctions)
- Central feature type: junction
- Inclusion of: all features that are associated by one of the 16 association types defined on the junction feature type
- Attribute types: 10 (official names and type IDs) from the set of all attribute types for the feature types involved
- Minimal support: 0.05
- Minimal confidence: 0.90
- Maximal rule length: 5

This results in 90 frequent queries and 79 query extensions. The outcome rule with the highest confidence below 100% is:

$foreach\_feature(A, \text{“}Junction\text{”})$ :
$feature\_exists(B, \text{“}Calculated\ Prohibited\ Manoeuvre\text{”}, assoc, A)$,
$\Rightarrow feature\_exists(C \neq B, \text{“}Calculated\ Prohibited\ Manoeuvre\text{”}, assoc, A)$
Confidence: 0.9795
Support: 0.1019

**Explanation** The rule means that, if a prohibited manoeuvre is defined over a junction, also another prohibited manoeuvre exists over this junction. The ‘Calculated Prohibited Manoeuvre’ association type defines forbidden traffic turns over a set of junctions, connected by the role type ‘Via Junction’.

**Feedback** Data experts identify this rule as a promising check, although ‘Calculated Prohibited Manoeuvre’ is an attribute generated from basic attributes that are already present in the data. This rule has 3 outliers in the data, 2 of which are located at the border of the data set. These are false-positive outliers due to incomplete information. A third one triggered further study by the engineers.

### 3.5.2 Experiment 2: discovering intra-feature relations

In a second experiment, we try to find relationships amongst the attributes of road elements.

- Geographical data set: northern Barcelona (consisting of 1851 road elements)
- Central feature type: road element
- Inclusion of: none
- Attribute types: 20 attribute types (about name, postal information, speed restriction, routing classes, etc.) belonging to the road element feature type
- Minimal support: 0.05
- Minimal confidence: 0.90
- Maximal rule length: 4

This results in 190 frequent queries and 169 query extensions. The 3 most interesting outcome rules with confidence below 100% are:

$foreach\_feature(A, \text{“}Road\ Element\text{”})$ :
$simple\_att\_exists(B, \text{“}Routing\ Class\text{”}, A, \text{“}Local\ Roads\ of\ High\ Importance\text{”})$,

$\Rightarrow simple\_att\_exists(C, \text{“}Road\ Conditions\text{”}, A, \text{“}Paved\text{”})$

Confidence: 0.9981

Support: 0.5786

$foreach\_feature(A, \text{“}Road\ Element\text{”}) :$

$simple\_att\_exists(B, \text{“}Functional\ Road\ Class\text{”}, A, \text{“}Local\ Roads\text{”}),$

$\Rightarrow simple\_att\_exists(C, \text{“}Routing\ Class\text{”}, A, \text{“}Destination\ Traffic\text{”})$

Confidence: 0.9947

Support: 0.3047

$foreach\_feature(A, \text{“}Road\ Element\text{”}) :$

$simple\_att\_exists(B, \text{“}Form\ Of\ Way\text{”}, A, \text{“}Road\ in\ Pedestrian\ Zone\text{”}),$

$\Rightarrow simple\_att\_exists(C, \text{“}Functional\ Road\ Class\text{”}, A,$

$\text{“}Local\ Roads\ of\ Minor\ Importance\text{”})$

Confidence: 0.9917

Support: 0.0643

**Explanation** These rules show obvious correlations between a road's importance, its form and its actual condition. Their respective meanings are that each 'Road Element' that

1. has the 'Routing Class' label 'Local Roads of High Importance', has the 'Road Condition' label 'Paved'.
2. has the 'Functional Road Class' label 'Local Roads', has the 'Routing Class' label 'Destination Traffic'.
3. has the 'Form Of Way' label 'Road in Pedestrian Zone', has the 'Functional Road Class' label 'Local Roads of Minor Importance'.

**Feedback** According to the data experts, the first rule reveals an interesting relationship. A 'Routing Class' reflects a relative importance, whereas a 'Road Condition' however describes a physical state. This means that an individual 'Routing Class' attribute is strongly related to the global attribute distribution over a country, such that the 'Routing Class' distribution for unpaved roads varies from country to country. Note that it is not unusual to include country-dependent information in the quality rules, but that including the geographical dimension in the analysis is beyond the primary scope of this tool for automated rule discovery.

The second rule shows a correlation between two road class categorisation

systems. This correlation is already implied by internal road class production rules.
The third rule indicates an interesting correlation between the 'Functional Road Class' and 'Form Of Way' attribute. The first one indicates a relative importance regarding functional aspects of a road, whereas the latter combines both physical and functional aspects. In this case, 'Road in Pedestrian Zone' is a purely functional determinant. The single outlier, a relative important road element in a pedestrian zone, is most probably an anomaly and the rule has been accepted for further inspection.

### 3.5.3 Rule set for experiment reconstruction

In this evaluation phase, we verify whether end users would be able to discover rules that are currently in use by the quality maintenance system. This sanity check involves experiment reconstruction for this selection of rules. It assumes unawareness by end users of these rules. The top column of Table 3.8 and 3.9 show 4 rules that have been manually designed from specifications by data engineers. For each of the rules, we set the experiment parameters such that it has the rule amongst its results and such that data engineers are not able to predict the rule as an outcome of the experiment set-up.

Table 3.8 and 3.9 show some detailed information about the experiments. In practice, it is often needed to lower the minimal support in order to find the target rules. The target rules could be found in experiment 1,2 and 4. For experiment 3, the targeted relation was not present in the input data set (which was checked manually). No outliers could be detected with regard to these rules, because they had already been adopted by the quality maintenance system. Each of the rules comes with a set of other rules, most of the time containing valuable information. Most of the targeted rules are short, so the total number of rules can be kept low by lowering the maximum rule length.

## 3.6 Future work

The sanity check in Section 3.5 has shown that the tool is able to discover realistic quality rules. However, the current rule language still has limitations. This section presents two language extensions that adapt the rule expressiveness to real-world standards.

| **Quality rule** | A Road Element that is part of a Freeway Intersection, shall not be part of another Freeway Intersection (FWI). | A face shall not be part of 2 or more Postal Districts (PDs) |
|---|---|---|
| **Experiment description** | Find relations between backward associated features to each road element; in this set FWI is unique | Find relations between backward associated features to each face; in this set PD is unique |
| **geographical data set** | Crisler | Crisler |
| **Central feature type** | Road Element | Face |
| **# interpretations** | 1851 | 674 |
| **Inclusion condition** | Association | Association |
| | All non-abstract backward associations | All non-abstract backward associations |
| **Attribute Types** | - | - |
| **Constraint approach** | Datamodel mining | Datamodel mining |
| **Minimal support** | 0.02 (FWI has low support) | 0.05 |
| **Minimal confidence** | 0.90 | 0.90 |
| **Maximal rule length** | 4 | 2 |
| **Targeted rule** | $foreach\_feature(A,$ "$Road\ Element$") : $feature\_exists(B,$ "$FWI$", $assoc, A)$ $\Rightarrow is\_unique(B, A)$ | $foreach\_feature(A,$ "$Face$") : $feature\_exists(B,$ "$PD$", $assoc, A)$ $\Rightarrow is\_unique(B, A)$ |
| **Confidence and support** | 1.0 0.0427 | 1.0 1.0 |
| **Violations** | 0 | 0 |
| **Number of rules per level** | 4+26+105+289 | 7+59 |

Table 3.8: Sanity check details (experiments 1 and 2).

| | | |
|---|---|---|
| **Quality rule** | Road Elements having a Functional Roadclass (FRC) attribute 'Motorway', 'Major Road', 'Other Major Road', 'Secondary Road' or 'Stubble' shall have a 'No Obstruction' Blocked Passage attribute. | A Junction can bound exactly 2 or 0 Road Elements with Form of Way (FOW) Roundabout. |
| **Experiment description** | Find relations between attributes of each road element | Find relations between (attributes of) road elements that overlap each junction |
| **geographical data set** | Crisler + Elzie + Malta + Nilsson (reason: FRC variation) | Nilsson |
| **Central feature type** | Road Element | Junction |
| **# interpretations** | 6765 | 1611 |
| **Inclusion condition** | - | Overlap |
| | - | Road Element |
| **Attribute Types** | Everything from Composite Blocked Passage + FOW and FRC from Road Element | (Composite) Official Name, FOW and FR from Road Element |
| **Constraint approach** | default | default |
| **Minimal support** | 0.01 (FRC 2 3 4 8 have low support) | 0.01 (FOW 3 has low support) |
| **Minimal confidence** | 0.7 (to show invalidity of target rule) | 0.90 |
| **Maximal rule length** | 5 | 3 |
| **Targeted rule** | not found | $foreach\_feature(A,$ "$Junction$") : $feature\_exists(B,$ "$Road\ Element$", $overl, A)$, $simple\_att\_exists(C,$ "$FOW$", $B,$ "$Roundabout$") $\Rightarrow feature\_exists(C \neq B,$ "$Road\ Element$", $overl, A)$ |
| **Confidence and support** | - | 1 0.02 |
| **Violations** | - | 0 |
| **Number of rules per level** | 1+0+9+16+28 | 2+2+7 |

Table 3.9: Sanity check details (experiments 3 and 4).

**Association** Presently, association is only used as a condition to include other features in an interpretation. Full integration means that the rule language is able to capture associations (by name and by role) between features and to list properties of associations. This would enable:

- the discovery of recurring patterns in association roles and association properties. An example could be: if a junction is the first junction of a manoeuvre, it is always the last junction of another manoeuvre.
- the combinatorial application of different inclusion conditions. For example, this would enable finding that a junction's associated intersection also overlaps this junction.

**Spatial functions and concepts** There is a number of functions and concepts, tailored to the domain of geographic databases, that would be very useful when integrated in the current system.

- Feature count, for example, supports the discovery of certain types of anomalies in geographical data, such as erroneous duplication of data. An example rule is: the number of hotels in a city is lower than the number of restaurants.
- Spatial distance (for feature sizes as well as distances between features) can be realised by calculation during preprocessing, and making it explicit in the rule language. This measure would enable finding that the distance between a gas station and a motorway is always between 10 and 100 metres.

## 3.7 Conclusion

We have built a tool to mine for relational regularities and corresponding outliers in geographical data. This tool assists a geographic content providing company in reasoning about the structure of the data and about the data itself. It is able to extract previously unknown knowledge in an automated way, which can be integrated in the quality maintenance process directly. An example of this knowledge is the rule stating that local roads of high importance are paved. The tool anticipates the process of manual rule formulation driven by individual reporting of anomalies in the data. Moreover, it is independent from the data model currently in use.

The WARMR algorithm is the central component of this tool. Its input consist of interpretations, a background knowledge and a language bias, generated from the end user's data selection and mining preferences. Its output is used for relational outlier detection by indirect description i.e. first WARMR mines for rules that describe regularities and next, violations of these rules are identified as outliers.

The case studies show that relatively simple experiments yield valuable information about regularities and outliers in the sample data. Three out of 4 manually designed example rules were reconstructed using the tool. Only one rule was not found because it had very low confidence over the sample data. The validation shows that the system requirements of our tool are met.

The approach to outlier detection and quality maintenance introduced in the present chapter, is generally applicable in data-rich intelligent systems. It was shown that this approach supports the discovery of valuable knowledge in geographical data, which cannot be discovered by traditional techniques for data quality analysis in GIS.

In the next chapter, the theme of shortest path approximation algorithms proceeds from the present theme of geographic data quality maintenance. This type of algorithm operates on a spatial graph, which is extracted from the geographical data treated in the present and the previous chapter. The extraction of time- or distance-weighted graphs entails a straight-forward decision model based on the road elements' and junctions' attributes in the data, as illustrated in Appendix A. The extraction of graphs taking into account attractiveness requires more advanced decision models integrating spatial relations between features of various types e.g. proximity of a forest to a way. Hochmair and Navratil [49] discuss specifically proximity in this context. Typical applications of shortest path algorithms are point-to-point route planners and navigation systems. The quality of these service highly depend on the quality of the geographical data from which the applicable spatial graph was extracted.

# Chapter 4

# Wayfinding by multi-level heuristic node promotion in real road networks

The present chapter introduces the application of the multi-level heuristic node promotion algorithm to real road networks for vehicle navigation. In contrast with many classical shortest path algorithms, this hierarchical shortest-path approximation algorithm integrates in a multi-tier web architecture in such a way that routing queries as well as a minor data updates are processed in a short amount of time. The multi-level heuristic node promotion algorithm was first proposed by Jagadeesh et al. [52] for two levels, although it is only effective when an irreversible graph transformation has been applied on the road network during preprocessing. This irreversible transformation, which consists of slip road removal and dual carriage way reduction, is problematic in contemporary route planning applications. Several heuristic adaptations to both the data preprocessing and the algorithm are introduced and motivated. These adaptations bypass the irreversible graph transformation and restore the effectiveness of the heuristic node promotion algorithm.

A computational experiment shows the application of the hierarchical algorithm to the 5-level time-weighted road network graph of Belgium, in combination with node pruning using a rectangular area. It analyses the effects of each of the adaptations on the routing performance. This experiment is conducted in the context of a routing web application for tourism and leisure purposes, but the suggested approach is effective for hierarchical shortest-path applications in

general.

The research in this chapter has been carried out as part of the industrial PhD project 'Structural heuristics for personalised routes' funded by the IWT (090726) and the company RouteYou. The present chapter is available as a technical report.

## 4.1 Introduction

RouteYou manages a web 2.0 environment enabling users to interactively create, share and use tourist routes. Besides, it offers a routing platform for various application developers and digital content providers. One of its basic components is a routing engine that computes a route of interest, mainly intended for vehicle navigation, over the road network between two points selected by the user. This calculation entails finding the path with the lowest cost in a directed weighted graph. The engine supports several routing modi, each of them referring to another type of edge weights. For the modus **'shortest'**, the weights represent the edge length. A time-weighted graph enables finding the **'fastest'** route. These time weights are estimates of the average time it is necessary for a vehicle to traverse the edge. Moreover, the engine offers a gamut of modi, further referred to as **'nicest'**, tailored to subdomains of leisure and tourism. For these modi, the weights correspond to the multiplications of the edge length and an unsuitability factor $\geq 1$, which is inversely proportional to the suitability of the edge to the subdomain. For cycling, for example, unsuitability is determined by both physical and scenic road characteristics and traffic regulations.

The path calculation infrastructure is a multi-tier architecture consisting of the following properties.

- All road network graph information is stored exclusively in a relational database with a spatial extension.
- The actual path calculation is executed on an application server.
- The road network graph is subject to minor updates such as new road segments or changing costs regarding the traffic situation.

This architecture supports a multi-purpose GIS system with high scalability with regard to the number of parallel users of services offered by the system. It gives rise to the following processes supporting the routing application:

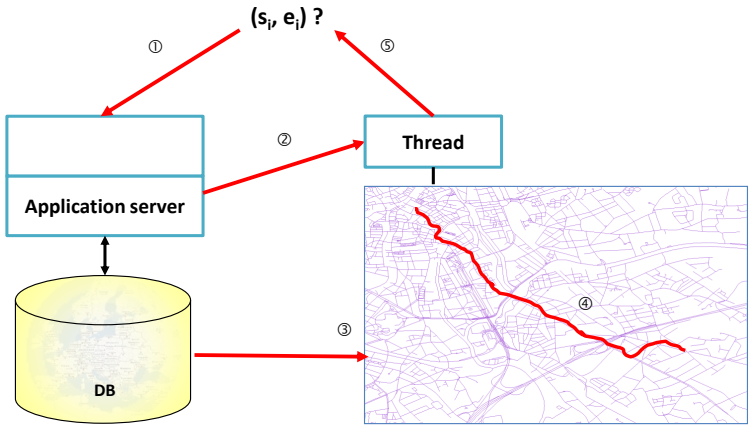



Figure 4.1: Routing process diagram. Upon each routing request from starting point $s_i$ to end point $e_i$ (step 1), the application server starts a thread (step 2). Once or repeatedly, this thread loads the portion of the road network nodes and edges that spatially intersect a bounding box (step 3). When the path finding process has terminated (step 4), the thread returns the result to the client and dies, releasing the memory occupied by the thread (step 5).

1. **preprocessing** the road network. This step involves the transformation of geographical data into an indexed and annotated road network graph. The word *indexed* refers to structures that optimise data consultation in process 2a, e.g. static clusters or a spatial index, such as an R-tree or QuadTree. The word *annotated* refers to the enrichment of the data by heuristic information (e.g. node coordinates, obstacle segmentation lines, hierarchical labels) in order to accelerate the search in process 2b. This step needs only to be executed once.

2. the routing process, triggered by end user requests. It is shown in Figure 4.1 and consists of:

   (a) **consulting** and reindexing relevant data. Reindexing in its turn involves building structures in the annotated graph, which facilitate the search process in process 2b.

   (b) the shortest path finding process itself.

3. **updating** the indexed and annotated road network. An atomic update involves the update of an edge weight or the insertion or deletion of an edge or a node.

**Peculiarity of the infrastructure.** Process 2a does not apply to traditional infrastructures. In memory-based infrastructures, all data is loaded in memory only once, after preprocessing. In disk-based infrastructures, the indexed and annotated data is written on a disk in an adapted format during preprocessing, and disk access is integrated in the shortest path finding process. When the shortest path calculation is written as an extension of the RDBMS, elementary graph operators and indices can be integrated in order to improve performance and avoid reindexing [38].

**Requirements.** Given the online nature of the application, the twofold routing process is evaluated based on the 3 following **performance criteria**: *(1) computational time*: end users need the answer in less than 5 seconds; *(2) memory footprint*: a low amount of memory used by the routing process, increases the number of parallel requests the web application allows handling; and *(3) quality of the route*. Quality is expressed in terms of relative error with regard to the cost of the exact solution. It is required that *(4) the computational time* of one atomic update is less than 1 second. Other performance criteria are: *(5) the computational time* of preprocessing, and *(6) the size* of the indexed and annotated data generated during preprocessing.
Process 2a in the multi-tier architecture is time consuming (criterion (1)). Its time complexity is at least linear to the size of the consulted data and each query introduces a constant latency time. Moreover, the size of the consulted data is proportional to the memory footprint (criterion (2)). This imposes that the routing algorithm requires only a concise subset of the data, transferred from the database in a low number of efficient queries, irregardless of the routing distance.

In the overview of heuristic approaches to one-to-one wayfinding in Section 4.2.1 of the present chapter, we show that classical hierarchical shortest path algorithms suit the requirements imposed by the application best. Section 4.2.2 discusses the applicability of these approaches to several routing modi and Section 4.2.3 further outlines this type of approaches in the field. Section 4.3 of this chapter introduces a new hierarchical shortest path algorithm, which is called multi-level heuristic node promotion (MLHNP).
Similar to other hierarchical path finding approaches, this algorithm can only be effective on real road network data if the road network graph is transformed during preprocessing. The application of this transformation is problematic in contemporary routing applications. Section 4.4 introduces a set of heuristic adaptations to the MLHNP approach such that it is effective in untransformed

network graphs. Section 4.5 evaluates the performance of hierarchically finding the 'fastest' routes for a test set of 1000 routing queries in Belgium. Section 4.6 is the conclusion of this chapter.

## 4.2 Related work

### 4.2.1 Heuristics for one-to-one shortest path finding

During the last two decades, one-to-one shortest path algorithms for realistic road networks have received extensive attention. Zhan and Noon [117] compared 15 optimal shortest path algorithms on two test sets covering the U.S. The most effective one-to-one algorithms are variants of the algorithm proposed by Dijkstra [30], which differ with regard to the data structure for finding the node with the smallest label for examination during search. Heuristic shortest path algorithms use extra knowledge about the road network in order to make the search process less computationally expensive.

The A* algorithm, first proposed by Hart et al. [48], limits the search area to an ellipse by including a heuristic estimation of the cost to the end node in the node traversal priority function. The extra knowledge consists of the node's geographical coordinates. The algorithm is guaranteed to find the optimal solution when the heuristic is admissible i.e. always underestimates the cost. In distance-weighted networks, Euclidean distance is an admissible heuristic. Jacob et al. [51] showed that the introduction of an overdo parameter, multiplying the Euclidean distance to the end node, drastically decreases computational time while preserving the solution quality to a large extent. The idea behind this 'overdo heuristic' is a better estimate of the conservative lower bound of the A* algorithm. Other A* variants aim at improving the cost estimate itself by: using a learning algorithm [6], looking ahead one node further [1], precomputing the shortest paths between a selection of landmarks [41] or modelling obstacles through segmentation lines [46].

Branch pruning is another form of limiting the search area, which involves ignoring the nodes during the search process that have low probability to being on the shortest path from starting to end node. This could be realised by the use of a stochastic model [36] or just by restraining the search space to a rectangular area containing start and end node, in the direction of their connecting line [55], or parallel to the axes of the coordinate system [56]. More recent work focuses on heavily precomputed data structures defining a set of prunable nodes for each of the individual edges. Wagner and Willhalm [115] approximate the set of nodes containing a shortest path by a geometry (so-called consistent

container). Lauther [65] saves for each of the edges $e$ and each of the predefined regions $r$ whether there exists a shortest path through $e$ to $r$. Gutman [44] introduces the reach of a node $v$ i.e. for all shortest paths $P$ through $v$, the maximum of the cost of the prefix and the suffix of P. Next, the nodes are pruned, for which the reach is higher than the underestimation of both the prefix and suffix of the shortest path under construction. The same author also presents a faster preprocessing method, which computes the upper bounds of the maxima. A bi-directional version of this algorithm has been improved by Goldberg et al. [42]. Moreover, the introduction of shortcut arcs drastically increased the preprocessing efficiency.

Hierarchical shortest path algorithms use a road network divided into multiple levels (e.g. the road classes) in order to limit the search link space drastically: only close to the start and end node the lower level links are taken into account. This divide-and-conquer strategy was first introduced and related to the field of cognitive psychology by Car and Frank [14]. Initially, the preprocessing phase of this type of approaches started from a hierarchically labelled network, and most algorithms were approximative. In the context of these *classical hierarchical approaches*, Fu et al. [37] identified the hierarchical shortest path algorithm as the most efficient heuristic in real transportation networks. They refer to the experiments by Liu [72], which show that the hierarchical algorithm reduces the average computational time of Dijkstra to one tenth, whereas A* only reduces it to half and bi-directional search to 80% (as first introduced by Dantzig [22]). A second type of approaches involves hierarchy computation during preprocessing, and does guarantee optimal results. Sanders and Schultes [92, 93] obtain a highway hierarchy by iteratively selecting all edges that appear in shortest paths between any nodes $u$ and $v$, but outside the local neighbourhood of $u$ and $v$. It integrates highway node contraction similar to shortcut arcs. A contraction hierarchy [39] arises from iteratively contracting any node in a heuristic order of importance. For the latter approach, path calculation times of 0.2-0.3 ms are reported. The transit-node approach [7] starts from a map divided into a set of disjoint regions. It involves identification - for each of the regions - of a small set of nodes where any shortest path leaving or entering the region passes. Next, all shortest paths between any two transit nodes of different regions and between any node of a region $r$ and any transit node of $r$ are precalculated. This approach further reduces the path calculation time to the order of 5-20 $\mu$s. Goldberg et al. [42] found that the performance of their algorithm based on reach pruning is very similar to the performance of a hierarchical algorithm. They introduced the concept of *cardinality reach*, referring to the traversal order during search, in order to emulate hierarchical behaviour in terms of a reach algorithm.

**Multi-tier and requirement suitability of the heuristics.** Both the

variants of the Dijkstra and A* algorithm can easily be integrated in the multi-tier architecture, in combination with pruning the nodes by a rectangular area. This is illustrated in Figure 4.1. Efficient bounding box selection is supported by almost every spatial DBMS. Nevertheless, these algorithms do not scale for routing queries over higher distances $d$, since the size of the data consulted by process 2a has a worst-case complexity of $\mathcal{O}(d^2)$, assuming a uniform spatial distribution of the road network graph.
Branch pruning heuristics have typically been designed for pruning during the shortest path finding process. Only when the pruning technique can be adopted by process 2a, i.e. integrated in a few efficient database queries, it might scale for routing queries over higher distances. This integration is possible for the algorithms by Wagner and Willhalm [115] and Lauther [65], but does not realise a significant scalability improvement. In the case of reach pruning, contemporary spatial DBMS technology does not enable to efficiently query the spatial points of which one of the attributes (reach number) is higher than the euclidean distance to two given points. Moreover, a dynamic version of this algorithm has only been considered for edge weight changes, for the version without shortcuts and without experimental evaluation [96].
The multi-tier integration of both the highway and contraction hierarchy approaches is problematic since there is no sound mechanism to reduce the number of consulted edges in advance in a scalable way. Data updates can be executed in short computational time [96], but are constrained to changing the edge weights. The transit-node approach is compatible with the multi-tier architecture, but burdens the precalculated data size, and is unsuitable for efficient data updates.
Classical hierarchical approaches integrate generally well in the multi-tier architecture because the algorithms operate on strongly reduced parts of the road network graph, which can be indexed by location and level. This enables a low memory footprint and computational time for process 2a. During preprocessing, the road network annotation is derived from the source data by simple operators such as intermediate node reduction and spatial containment. As a consequence, preprocessing times are rather short, the generated data size is small and minor data updates can be processed in less than 1 second. The limitations of this type of approaches are the approximative character of the resulting paths, and the necessity of a predetermined network hierarchy, which is discussed below.

### 4.2.2 Applicability of classical hierarchical approaches

Classical hierarchical approaches are mostly applied in transportation network graphs, in which edge weights represent travel time. The road classes then simply determine the levels of the network. This road class heuristic is a good

indicator for the actual speed over the edges and the subnetwork of the $n$ highest classes is fully connected by nature. The road classes can be easily obtained from geographic content providers.

Furthermore, classical hierarchical approaches are applicable to a wider domain of *routing modi*. For some countries, the road class is also a good heuristic to find the shortest path in a distance-weighted road network graph. Jagadeesh et al. [52] showed for Singapore that it was due to the road class reflecting the inherent hierarchy of the graph topology. Alternatively, Chou et al. [20] suggest to promote edges with a high length to the higher level for this type of weights. The same classical hierarchical algorithms can be used in this context of finding the 'nicest' routes, but the generation of hierarchy for these modi is beyond the scope of this chapter. For hierarchical networks in general, we assume that

- each subnetwork of the $n$ highest classes is fully connected.
- the number of edges in the higher classes is restricted.
- the edges of a higher class are more probable to be included in the shortest path from any node $a$ to any node $b$ of the graph. This results typically in a selection of edges of lower unsuitability (higher speed) enriched by edges that realise topological connectivity (e.g. slip roads, 'unsuitable' ('low speed') path that is an essential shortcut).

### 4.2.3 Classical hierarchical shortest path finding

Classical hierarchical wayfinding requires a network graph to be divided in two or more (connected) graphs. **Leave points** denote the nodes that enable traversal from a given level network to a higher level network. The term **entry points** is used for the ones that give entry to a lower level network. The term **transition points** refers to both entry and leave points. Several algorithms have been proposed that optimally conduct the search process over different levels.

Car and Frank [14] proposed a bottom-up approach to multi-level hierarchical routing. It uses a hierarchical tree of meshes i.e. a set of edges bounded by higher-level edges, and starts from the lowest level. It involves that first the shortest path from the start node to the closest leave point and from the closest entry point to the lower level are found, using a classical algorithm, on the lowest level network. Next, the infix path is recursively searched for on the higher level network. The recursion ends when start and end node are situated in the same or in adjacent meshes, or when no higher level is available. In this case, the shortest path from start to end node is found using a classical

algorithm. In the end, the resulting subpaths are concatenated. Later on, Cho and Lan [19] adopted the bottom-up approach and studied the memory-error trade-off for different combinations of Dijkstra and A* with 'overdo heuristic' on the different levels of operation.

One major drawback of the bottom-up approach is that the choice of the closest transition point is not necessarily the best choice to produce the path with the lowest cost. Therefore, Liu [72] introduced the stitching approach, i.e. the meshes in which the start and end node are located are added to the higher-level network, and a classical algorithm is used to find the shortest path from the start to the end node in the stitched network graph. Chou et al. [20] suggested to calculate the shortest paths from the start node to all leave points and from all entry points to the end node and to select the combination with the lowest cost afterwards. Jagadeesh et al. [52] introduced the two-level hierarchical algorithm with heuristic node promotion. It is a top-down approach that starts routing at the higher level if start and end node are situated in the same or in adjacent meshes. This higher level routing stage involves finding the shortest path from start to end node on the higher level network, enriched by a set of *virtual links* from the start node to each of the leave points and from each of the entry points to the end nodes, using a classical algorithm. The virtual link weights are cost estimates based on Euclidean distance. Next, the two virtual links selected in the resulting path are replaced by the shortest path from start to end node of the virtual links, obtained by routing on the lower level network. The authors showed that their method outperforms (computation time vs. result quality) the two-level version of the bottom-up approach.

The algorithms described above typically integrate a pruning technique in order to speed up the search during each of the subroutings. Jagadeesh et al. [52] constrain the search space to the subgraph formed by the $N$ closest nodes to the start and end node. Liu [72] uses cell meshes to prune the search graph. In the first case, it may be difficult to find a general measure for $N$ or an accurate distance function for scenic or time-weighted routing modi. The latter case will certainly work for planar graphs, but may not be effective in non-planar graphs without intensive preprocessing of overlapping cells.

Strauss [105] raised some issues concerning unconnected higher level network graphs that apply to the hierarchical approaches above. He identified the case of (1) a disconnected higher-level graph due to the data selection, (2) a disconnected higher-level graph because of the topographic restrictions of the dataset, and (3) the low-level shortcut which connects two high-level subnetworks. Examples of the latter case are a footpath connecting the railway system with the airport or a shortcut between two highways which often appears in paths with the lowest travel time. In this context, Liu [72] suggests to add 1-edge shortcuts within the higher network during preprocessing. Strauss [105] introduces the multi-level

hopping algorithm to cope with all these issues. This method alternates between synchronous and asynchronous search tree spreading in a bidirectional manner. We argue that each of the issues raised can be avoided by the following measures: (1) expand the data selection in case the routing algorithm detects a disconnection, (2) organise the global higher network in such a way that it is a connected graph (this could be imposed without any loss of route quality)[1], and (3) classify the edges according to the probability of being included in any shortest path. The last measure has already been suggested by Jagadeesh et al. [52] for slip roads between motorways.

**Concluding remark.** Classical hierarchical algorithms can be categorized into four main algorithmic strategies: top-down, bottom-up, stitching and level hopping. In combination with a spatial or mesh-based pruning, the first three strategies integrate well in the multi-tier architecture. Given the upper bound number $u$ of edges contained in a cell or spatial container, the worst-case size complexity of the data consulted by process 2a is limited to $\mathcal{O}(2 \cdot l \cdot u)$, with $l$ the number of levels. Section 4.3 introduces a variant of the top-down approach, which was shown to yield the optimal trade-off of computational time and result quality. The multi-level hopping algorithm requires process 2a to consult as much data as a Dijkstra or A* algorithm and therefore it does not fulfil the requirements.

## 4.3 Multi-level heuristic node promotion in transformed network graphs

The present section introduces the multi-level heuristic node promotion algorithm (MLHNP). It is a recursive version of the two-level algorithm by Jagadeesh et al. [52]. Whereas slip roads and dual carriageways typically occur in networks for vehicle navigation, the algorithm is only effective on road networks that have been subjected to the following graph transformations: (1) slip road removal, and, (2) dual carriage way and multi-lane reduction into single-track ways. These processes reduce complex intersections to a single node connecting the main directions.

[1]Note that a disconnected highway graph due to dataset restriction often coincides with the inability to produce the *real-world* fastest path with any algorithm, because this path runs through one or more neighbouring datasets. This is not a valid excuse for the algorithm to fail, but it indicates that this type of case applies to rather experimental than operational planners.

| | Concept | Definition |
|---|---|---|
| **Graph** | $G = (V, E)$ | a directed road network graph in which each of the edges $e \in E$ is assigned a weight $w(e)$ and a level $l(e) \in [0, maxlevel]$. Any directed edge $e$ starts in node $from(e) \in V$ and ends in node $to(e) \in V$. We will use the symbols $\dot{\in}$ and $\widetilde{\in}$ to denote set membership of $V$ and $E$. |
| | $G_L$ | the subgraph of $G$, consisting of edges $e \widetilde{\in} G : l(e) \geq L$. $G_L$ should be connected for each $L \in [0, maxlevel]$. |
| | $edge(n_1, n_2, w)$ | an edge from vertex $n_1$ to $n_2$ of weight $w$ |
| | $reduced(G, X)$ | the transformed version of $G$ in which all intermediate nodes (i.e. nodes that only connect to two physical ways) have been removed. Intermediate node reduction is achieved by iteratively replacing each pair of edges $edge(a, b, w_1)$ and $edge(b, c, w_2)$ with intermediate node $b$, by edge $edge(a, c, w_1 + w_2)$ . The exception list $X$ is an optional parameter. Any node of $X$ that is a node of $G$ must not be removed. |
| **Paths** | $\mathcal{P}(a, b, G)$ | the set of paths from vertex $a$ to $b$ in graph $G$. It consists of any sequence $P$ of subsequent edges from graph $G$, for which the first edge starts in $a$ and the last edge ends in $b$ |
| | $P_i$ | the edge with position $i$ in path $P$ |
| | $P_{i...j}$ | the subpath of $P$ starting by $P_i$ and ending by $P_j$ |
| | $\|P\|$ | the cost of path $P$ is defined as the sum of the weights of all its edges |
| | $dist(a, b, G)$ | the shortest path distance from vertex $a$ to $b$ in $G$ is defined as $min_{P \in \mathcal{P}(a,b,G)} \|P\|$. The set of shortest paths $shortest(a, b, G)$ contains any path $P$ for which $\|P\| = dist(a, b, G)$ |
| | $error_{rel}(Q)$ | the relative error of an approximate shortest path $Q \in \mathcal{P}(a, b, G)$ is defined as $\frac{\|Q\| - dist(a,b,G)}{dist(a,b,G)}$ |

Table 4.1: MLHNP concept definitions (part 1)

| | Concept | Definition |
|---|---|---|
| **Cells** | cell $c$ | a spatial region containing nodes of a graph $G$. |
| | $cells(L)$ | the set of cells of level $L$ in graph $G$. Each polygon that is enclosed by edges in $G_L$ and eventually the dataset boundary, and does not overlap any other edges in $G_L$, is an element of $cells(L)$. |
| | $l(c)$ | $L \vert c \in cells(L)$ |
| | $cells(n, L)$ | the subset of $cells(L)$ of cells that spatially overlap node $n \dot{\in} G$. $\forall n \dot{\in} G, L \in [1, maxlevel] : cells(n, L) \neq \emptyset$ |
| | $contiguous(c_1, c_2)$ | is true when the cells $c_1$ and $c_2$ are contiguous i.e. their spatial intersection is at least a line. |
| **Node promotion** | $leavepoints(c)$ | the set of candidate transition points between any route starting in cell $c$ planned in any $G_m$ with $m < l(c)$ and $G_{l(c)}$ $leavepoints(c) = \{n \in c \vert \exists e \widetilde{\in} reduced(G_{l(c)}) : from(e) = n \wedge to(e) \notin c\}$ |
| | $entrypoints(c)$ | the set of candidate transition points between $G_{l(c)}$ and any route ending in cell $c$ planned in any $G_m$ with $m < l(c)$ $entrypoints(c) = \{n \in c \vert \exists e \widetilde{\in} reduced(G_{l(c)}) : to(e) = n \wedge from(e) \notin c\}$ |
| | $estimate(n_1, n_2)$ | a cost estimate of the shortest path from $n_1$ to $n_2$ |
| | $promote_{st}(n, G')$ | $\Leftrightarrow \neg(\exists e \widetilde{\in} G' : from(e) = n)$ |
| | $promote_{end}(n, G')$ | $\Leftrightarrow \neg(\exists e \widetilde{\in} G' : to(e) = n)$ |

Table 4.2: MLHNP concept definitions (part 2)

$mlhnp(n_1, n_2, l_{restr}, G)$ :
  $L_{cand} \leftarrow \{l \vert \forall c_1 \in cells(n_1, l), c_2 \in cells(n_2, l) : \neg(c_1 = c_2 \vee contiguous(c_1, c_2))\}$
  $l_{exec} \leftarrow min(l_{restr}, max(L_{cand} \cup \{0\}))$
  $G' \leftarrow reduced(G_{l_{exec}}, \{s_{global}, e_{global}\})$
  **if** $l_{exec} > 0$ **then**
    **if** $promote_{st}(n_1, G')$ **then**
      $G' \leftarrow G' \cup \{edge(n_1, n, estimate(n_1, n)) \vert n \in leavepoints(cells(n_1, l_{exec}))\}$
    **end if**
    **if** $promote_{end}(n_2, G')$ **then**
      $G' \leftarrow G' \cup \{edge(n, n_2, estimate(n, n_2)) \vert n \in entrypoints(cells(n_2, l_{exec}))\}$
    **end if**
  **end if**
  $R \leftarrow P \in shortest(n_1, n_2, G')$
  **if** $l(R_0) =$ '$v$' **then**
    $R \leftarrow concatenation(mlhnp(from(R_0), to(R_0), l_{restr} - 1, G), R_{1 \ldots size(R)-1})$
  **end if**
  **if** $l(R_{size(R)-1}) =$ '$v$' **then**
    $R \leftarrow concatenation(R_{0 \ldots size(R)-2}, mlhnp(from(R_0), to(R_0), l_{restr} - 1, G))$
  **end if**
  **return** $R$

Figure 4.2: Multi-level heuristic node promotion.

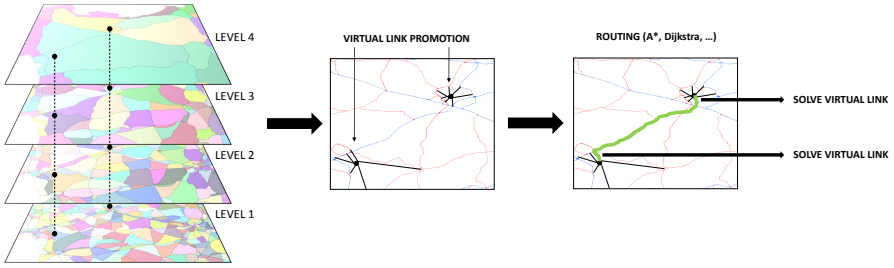



Figure 4.3: Recursive run of the multi-level heuristic node promotion algorithm.

### 4.3.1 Algorithm

MLHNP assumes the availability of the concepts listed in Table 4.1 and 4.2, applied to a network divided in $maxlevel+1$ levels. This algorithm approximates the path of lowest cost from vertex $s_{global}$ to $e_{global}$ in graph $G$, returning an approximate path $Q$ with a minimal $error_{rel}(Q)$. It is initiated by the call $mlhnp(s_{global}, e_{global}, maxlevel, G)$ to the algorithm listed in Figure 4.2. First, it determines its level of execution $l_{exec}$ as the highest level for which $n_1$ and $n_2$ are not located in contiguous or same cells. Next, the search graph is pruned and constrained to the edges of level $\geq L$. When $l_{exec} > 0$ and $promote_{st}(n_1, G')$, the search graph is extended by a set of virtual edges, from $n_1$ to $n_1$'s cell leave points. These virtualisations are part of the so-called *node promotion* because $n_1$ is promoted to $G_{l_{exec}}$. This promotion is *heuristic* because each of the virtual edge weights is a heuristic cost estimate of the shortest path from $n_1$ to the leave point. Analogously, when $l_{exec} > 0$ and $promote_{end}(n_2, G')$, the search graph is extended by a set of virtual edges, from $n_2$'s cell entry points to $n_2$. Next, a classical routing algorithm is applied from $n_1$ to $n_2$. This step is further referred to as **the basic shortest path calculation**. In the resulting path, each selected virtual link is replaced by the path resulting from the recursive call to this function. The recursive decrement of the level of execution guarantees that the algorithm is finite.

Figure 4.3 shows an example top recursive run of the algorithm. The left subfigure situates the origin and destination node $n_1$ and $n_2$ in the cell configuration $cell(L)$ for any level $L > 0$. Note that a cell of level $L$ is enclosed by edges of a level greater than or equal to $L$. $l_{exec} = 3$, since 3 is the highest level for which $n_1$ and $n_2$ are located in non-contiguous cells. Next, in the middle subfigure, the edges of $G_3$ are loaded only the edges of level 3 and 4 are loaded. Blue/red edges belong to level 4/3. This network is enriched with virtual links (denoted in black). The shortest path from $n_1$ to $n_2$ in this network

is shown in the right subfigure. It contains two virtual links, which need to be replaced by the shortest path from the start to the end node of the link. This path is obtained by a recursive call to the MLHNP algorithm.

### 4.3.2 The role of transition points and the heuristic estimate

Each cell corner connecting to other edges of the higher level network, is registered as a transition point. The transition point definitions in Table 4.2 realize this pattern likewise. A recursive run of MLHNP typically generates a route of the type
$(startnode) - virtuallink - (transitionpoint) - fixedroute - (transitionpoint) - virtuallink - (endnode)$,
in which the virtual links are solved in deeper recursions. So the solution of the virtual link typically contains higher level cell border edges and the transition point in the resulting route is the node where the route switches between the cell border and rest of the network. This means that the shortest path search on the border is preferably executed downlevel. This is a scalable approach that complies with hierarchical road network data containing more than two levels.[2].

The ultimate algorithmic choice of transition point is influenced by the virtual link cost estimate function $estimate(n_1, n_2)$. The better the approximation, the higher the chance that the transition point generating the shortest path is selected. Finding the shortest path in a graph with systematic overestimation of the virtual link cost, yields preference of transition points close to the starting node/end node. Systematic underestimation tends to minimise the cost of the global path apart from the virtual link. Jagadeesh et al. [52] approximate the real cost by the Euclidean distance in case of distance-weighted edges, and by the Euclidean distance divided by the average speed of the weights of the cell's edges in case of time-weighted edges. The latter approach assumes a homogeneous speed distribution over the cell. This is certainly not the case for cells of levels greater than 1. Individual variance between the real virtual link cost and its approximation leads more easily to a wrong transition point choice (i.e. resulting in a path with higher cost) than systematic deviation. Moreover, it requires extra preprocessing time and storage costs. Therefore we just use an

[2]We have considered alternative transition point definitions. (1) Each transition from the cell border to the lower level network. This approach involves that the transition points are candidate points where the shortest path switches between the cell border and the network inside of the cell. This approach is promising with regard to any of the routing performance criteria, but is not scalable because at high levels we could easily obtain cells of 10000 transition points. (2) Each transition from the cell border to edges of one level lower. This approach scales but imposes some additional requirements. Each cell of level $n$ should overlap edges of exactly level $n-1$. Moreover, it only supports paths that have a level increase/decrease by one, which deviate greatly from shortest paths in common hierarchies.

arbitrary average resistance $r_{avg}$ in order to estimate the virtual link cost for each of the routing modi as follows:
$estimate(n_1, n_2) = dist_{eucl}(n_1, n_2) \cdot r_{avg}$
This value is a single estimate of the ratio of the real path cost from any $n_1$ to any $n_2$ to the euclidean distance between $n_1$ and $n_2$.

### 4.3.3 Preprocessing overview

Starting from a road network graph cleared from slip roads and dual carriage ways, the following transformations and annotations are necessary during the preprocessing phase.

1. Intermediate node reduction of each graph $G_L : L \in [0, maxlevel]$. This transformation is necessary to speed up the search process of the shortest path algorithm and it supports higher level scalability of MLHNP. It is a reversible transformation because each transformed edge can be unambiguously translated into the original sequence of edges.
2. Appropriate indexing of each of the resulting graphs $reduced(G_L)$.
3. Cell generation and cell contiguity registration
4. Appropriate indexing of the cells and contiguities by node.
5. Generation and indexing of the transition points by cell.

Step 3 involves building polygons formed by the minimal cycles of the planar version of the road network graph $G_L$, for each $L \in [1, maxlevel]$. Planarisation involves that edge crossings are replaced by dummy vertices, for each subnetwork $G_L$, and that duplicate line parts are merged into a single line belonging to the highest level of both sources. Moreover, all directed edges have been made undirected in the planar graph. The planar graph has been extended by the border of the routable area, its level labelled by $maxlevel + 1$. Non-border edges may end at the border, but must not exceed it. As a result, this planar graph has a spatial representation in which none of the edges intersect, and any edge (not part of the border) encloses two minimal cycles of the graph.
This cell generation can be implemented as an iterative top-down process, which uses $G_N$ to split the cells of level $N + 1$ into cells of level $N$. At the top level it starts from one or more cells representing the routable area. In order to split one cell of level $N + 1$, all topology of $G_N$ overlapping the cell is loaded, and minimal cycle navigation starts at a random edge of exactly level $N$ (if not available, the cell can be copied down level). During navigation, the algorithm

chooses the leftmost edge at each node. Navigation ends when a visited node is encountered, closing the polygon. The next navigation starts in a random edge of exactly level $N$ that has only been visited once. The split-down finishes when no more start edges are available. Each time an edge is visited twice, a cell contiguity is registered. Note that choosing the same direction (leftmost or rightmost) for all navigations will guarantee that all minimal cycles are visited when any edge is visited in forward and backward direction.

### 4.3.4 Updating overview

Processing an atomic data update is quite straightforward. In case of a weight update of an edge of level $m$, it is only required to update the accumulated weight in any graph $reduced(G_L)$ with $L \leq m$. In case of insertion or removal of an edge $e$ of level $m$, it can be required (1) to break up one or two edges and (2) to reduce a novel intermediate node in any $reduced(G_L)$ with $L \leq m$. When an edge of level $m > 0$ that encloses both two cells in $cells(L)$ with $0 < L \leq m$ is removed, the cells (and their contiguities) should be merged. The remaining subset of the union of transition points of both cells should be determined. When the insertion of an edge of level $m > 0$ realizes a full separation of a cell in $cells(L)$ with $0 < L \leq m$ into two new cells, the cells (and their contiguities) should be split. The new cells inherit a subset of the original transition point set, and new transition points are amongst the nodes of the separation boundary.

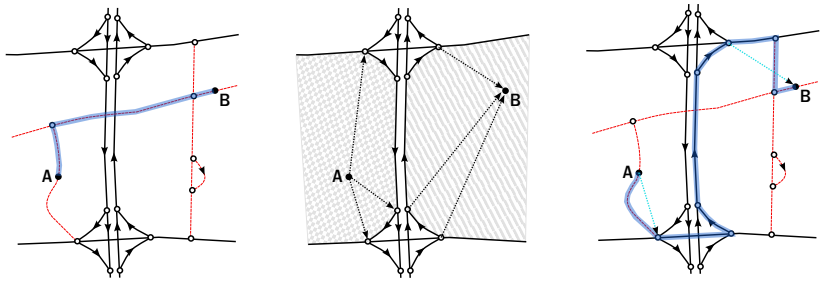
## 4.4 Multi-level heuristic node promotion in real road networks

Geographic content providers commonly model physically separated lanes as well as slip roads as separate edges. It is unclear how slip road removal and dual carriageway reduction, as suggested in the previous section, can be dealt with in a routing application for vehicle navigation.

- The transformation processes may be ambiguous, or only realisable in combination with turn restrictions, for instance when a complex intersection does not interconnect all directions.
- Contemporary routing applications enable visually selecting a starting or end point located on any of the edges originally provided. So the user may select a specific lane or slip road. The transformation projects many of such edges to one edge or node. So the routing in the transformed graph

| | |
|---|---|
| ○ | node |
| ——— | a high level edge (an arrow restrains the navigation direction) |
| - - - - | a low level edge (an arrow restrains the navigation direction) |
| A• | starting node |
| •B | end node |
| ▬ | path |
| ▩ | cell of starting node |
| ▨ | cell of end node |
| ·····→ | virtual link involved |
| ·····→ | selected virtual link that is replaced after recursion |
| ·····→ | virtual link that causes problems |
| - - -→ | useful virtual link |

Table 4.3: Legend.



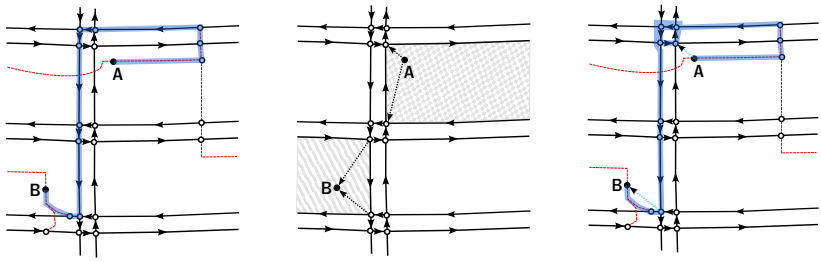

(a) Shortest path (b) Cells and virtual links (c) Resulting path

Figure 4.4: Scenario 1. The corresponding legend is in Table 4.3.

does not comply with the *route-by-click* philosophy of the application, unless the start and end neighbourhood in the graph is enriched by original data, or the route is corrected after the routing process.

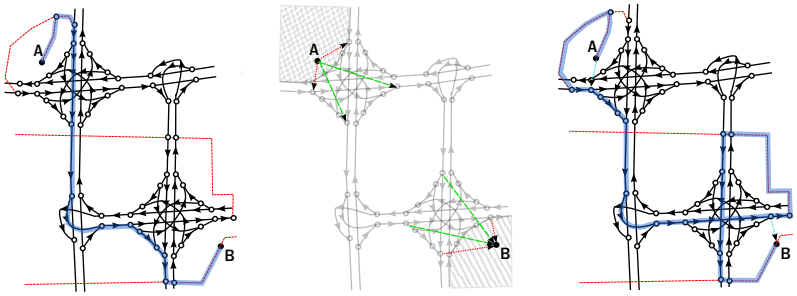
- Contemporary routing services return a line string, which is the exact route's geometry including the appropriate lanes and slip roads to be taken. It is not clear how this geometry can be reconstructed after the routing process.

This section introduces several heuristic adaptations to the MLHNP approach in Section 4.3 such that it is effective in network graphs that have not been subjected to slip road removal and dual carriageway reduction.

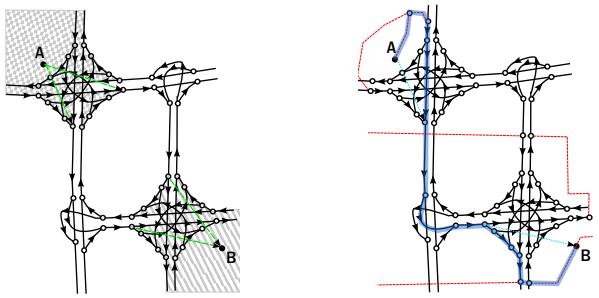



(a) Shortest path (b) Cells and virtual links (c) Resulting path

Figure 4.5: Scenario 2. The corresponding legend is in Table 4.3.




(a) Shortest path (b) Cells and virtual links (c) Resulting path




(d) Corrected cells and virtual links (e) Resulting path after correction

Figure 4.6: Scenario 3. The corresponding legend is in Table 4.3.

| | Concept | New definition |
|---|---|---|
| **Cells** | $cells(L)$ | the set of cells of level $L$ in graph $G$. Each polygon that is enclosed by edges in $G_L$ or edges of the dataset boundary, but does not overlap any other edges in $G_L$, is an element of $cells^{old}(L)$. Each merge of each cell in $cells^{old}(L)$ that is not part of a DCW or CI, with all of the contiguous DCWs and CIs, is an element of $cells(L)$. |

Table 4.4: Adaptation 1

### 4.4.1 Adaptation 1: cell classification and merge

#### Problem

First, when applying MLHNP to a transformed graph, the cell contiguity function in Table 4.2 is a good heuristic to determine the level of execution $l_{exec}$. This is not the case in a graph with dual carriageways and complex intersections. The routing scenario in Figure 4.4 is a typical routing example that should be executed at the lowest level because the cells in which start and end node are located would be contiguous after transformation.
Second, as described in Section 4.3.2, transition points determine where MLHNP switches between high and low level routing. In graphs with complex intersections, the cell and transition point definitions do not always generate optimal transition point locations. Figure 4.5, 4.6a, 4.6b and 4.6c illustrate that the transition points are located between the (candidate) virtual links and the complex intersection they are part of. This often restricts the cell border trajectory, which is planned at the lower level, to one direction.

#### Solution

Both issues have been sorted out by a heuristic cell definition adaptation, illustrated in Table 4.4. This adaptation reshapes the cell polygons, emulating the form they would have after road network graph transformation, using the classical cell definition. A first effect is that the cells in $cells(L)$ overlap near dual carriage ways (DCWs) and complex intersections (CIs). The contiguity of these overlapping cells can be used as an approximation of the contiguity of the routing graph which has been subject to the removal of DCWs and CIs. The other effect is that the transition points are located at the right side of the CIs, such that high level routing only concerns with routing between CIs and low level routing with routing over the CI, as illustrated in Figure 4.6d and 4.6e. In other words, the cell form is used as a heuristic to find unique transition points from/to the outside world of the cell.

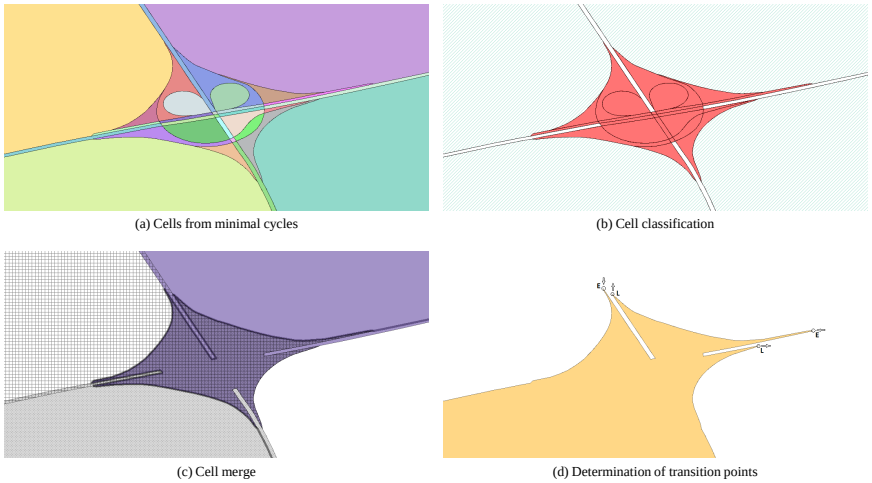
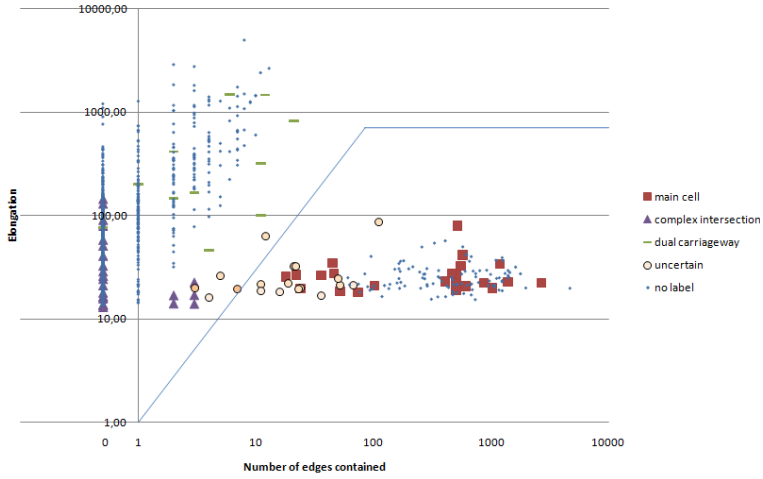



Figure 4.7: Cell builder and merger states in the neighbourhood of a complex intersection.

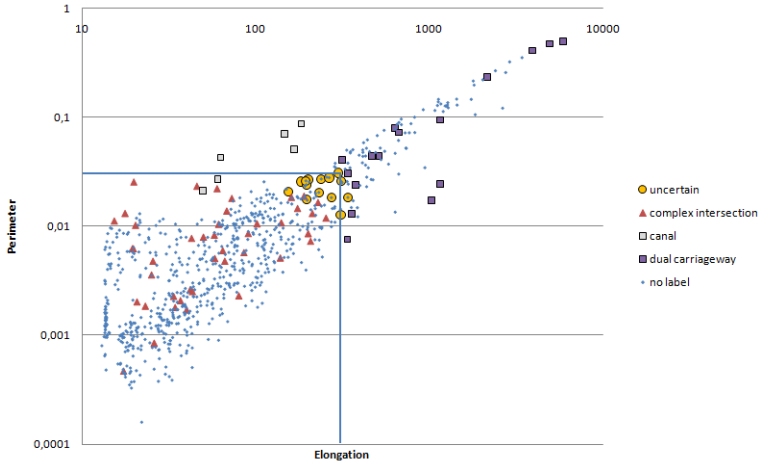
### Realisation

The cell classification and merge procedure need to be executed for achieving this adaptation, after the iterative top-down cell generation (Section 4.3.3). Figure 4.7 shows the cell processing phases that produce the adapted cells and transition points in the neighbourhood of a CI. First, all minimal cells are generated from the minimal cycles of the planar graph. These cells are depicted in Figure 4.7a, in which cells have a different colour. Next, 2-step classification is applied to each of these cells. Figure 4.7b shows regular cells in hatched blue, CI parts in red, and DCW parts in white. This classification is used by the cell merge algorithm. It results in only four cells, represented in Figure 4.7c, each containing as much as possible of the complex intersection. Figure 4.7d shows how entry points (E) and leave points (L) are assigned to the bottom left cell.

**Cell classification.** This process discriminates between parts of CIs, parts of DCWs and regular cells. The classification is twofold. For each of the two stages, a selection of level 1 cells of a sample area, has been tagged manually in order to construct a classifier.
A first classifier has been constructed in order to discriminate regular cells from CI and DCW parts. The graph in Figure 4.8a shows 78 manually tagged cells as a function of 1) the number of level 0 edges contained in the cell (X-axis) and 2) the cell's elongation $= perimeter^2/area$ (Y-axis), both on a logarithmic scale.




(a) Classification of regular cells versus CI or DCW parts. The elongation is depicted as a function of the number of edges contained, for 78 manually labelled cells.




(b) Classification of CI versus DCW parts. The perimeter of 78 manually labelled cells is depicted as a function of the elongation.

Figure 4.8: Classification of regular cells, CI parts and DCW parts.

These parameters are similar to the area and elongation parameters identified by Delafontaine et al. [29] in order to detect sliver polygons in map overlaying. Figure 4.8a contains labels for regular cells, CI parts, DCW parts and cells that have been described as uncertain. Examples of uncertain cells are those that are cut off by the global border and cells that are located between the two edges of a dual carriageway at one side but contain a considerable portion of urban area at the other side. Note that all unlabelled cells of the sample area are represented as well.
DCW parts are characterised by high elongation and a relatively small number of edges. They form a cluster with CI parts, characterised by a small number of edges. This cluster is disjoint from the cluster of regular cells, which have high numbers of edges and low elongation. Figure 4.8a depicts a linear classifier, generated in the logarithmic plane. It identifies a cell as being regular when $elongation < \#edges^{1.3424}$. In practice, the number of edges contained within a cell is counted by an SQL query. In order to speed up this query, the count is limited to 1000, before it is passed to the classifier. This is indicated by the plateau in the classifier.
The classifier scales well to cells of higher levels. Higher level DCW parts are even longer, containing more crossing edges. Thus, this category tends to move up right in the graph. CI parts are quasi static. Regular cells will move to the right because they become larger. So none of these categories tends to interfere with the classifier.

Next, non-regular cells are categorised into CI and DCW parts. The graph in Figure 4.8b shows 78 manually tagged cells as a function of the cell's elongation $= perimeter^2/area$ (X-axis), and, the cell's perimeter, here in digital degree (Y-axis), both on a logarithmic scale.
Figure 4.8b shows the manually assigned labels for CI parts, DCW parts and cells that have been described as uncertain. A special type of uncertain cells, i.e. DCW parts in which a canal is situated between the traffic flows, have been assigned a separate label. Note that all unlabelled non-regular cells of the sample area are represented as well.
DCW parts are characterised by high elongation and a long perimeter, whereas these parameters have low numbers for CI parts. Canals in DCWs cause lower elongation. Figure 4.8b depicts the classifier $elongation < 300 \wedge perimeter < 0.3$, which is true in case of a CI part.
The classifier scales well to cells of higher levels. DCW parts of higher levels tend to have longer perimeters and higher elongation. Thus, this category tends to move up right in the graph, away from the classification boundaries. CI parts are static.

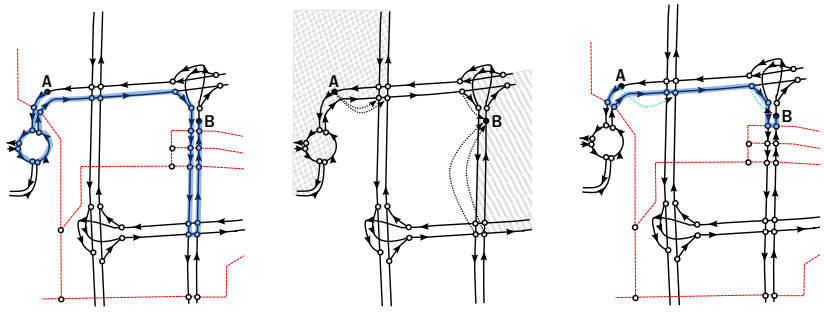
**Cell merge.** For each of the levels, the cell merge algorithm is applied to the set of classified cells. The algorithm in Figure 4.9 takes the complete set of cells

```
merge_cells(ALL, CIS, DCWS) :
  Candidates ← CIS
  while Candidates ≠ ∅ do
    for each complexInt ∈ Candidates do
      if ∃n : contiguous(complexInt, n) ∧ n ∈ Candidates then
        for each n : contiguous(complexInt, n) ∧ n ∈ Candidates do
          complexInt ← merge(complexInt, n)
          CIS ← CIS \ {n}
          Candidates ← Candidates \ {n}
          ALL ← ALL \ {n}
        end for
      else
        Candidates ← Candidates \ {complexInt}
      end if
    end for
  end while
  Candidates ← CIS ⋃ DCWS
  while Candidates ≠ ∅ do
    planning ← new array
    for each cell ∈ Candidates do
      planning[cell] ← {n : contiguous(cell, n) ∧ n ∉ Candidates}
    end for
    for each cell ∈ keys(planning) do
      for each colonist ∈ planning[cell] do
        colonist ← merge(colonist, cell)
      end for
      Candidates ← Candidates \ {cell}
      ALL ← ALL \ {cell}
    end for
  end while
  return ALL
```

Figure 4.9: Cell merge algorithm.

$ALL$ and its subsets $CIS$ and $DCWS$ that have been identified as CI / DCW parts, as its input. The algorithm first clusters neighbouring CI parts in singular cells. Next, all CI clusters and DCW parts are merged iteratively by each of their neighbouring regular cells. The outcome is illustrated in Figure 4.7c. This algorithm tries to maximise the cell overlap at CIs but prevents that cells absorb continuing DCWs along neighbouring regular cells. Note that a cell merge of $A$ by $B$ involves that $B$ inherits $A$'s neighbours.

**Transition point assignment.** Once all cells have been merged, the leave points and entry points of each cell can be easily determined, according to the definitions listed in Table 4.2. This is shown in Figure 4.7d.




(a) Path without virtualisation

(b) Cells and virtual links in case of virtualisation

(c) Shortest path with virtualisation

Figure 4.10: Scenario 4. The corresponding legend is in Table 4.3.

| | Concept | New definition |
|---|---|---|
| | $promote_{st}(n, G')$ | $\Leftrightarrow n = s_{global}$ |
| | $promote_{end}(n, G')$ | $\Leftrightarrow n = e_{global}$ |

Table 4.5: Adaptation 2

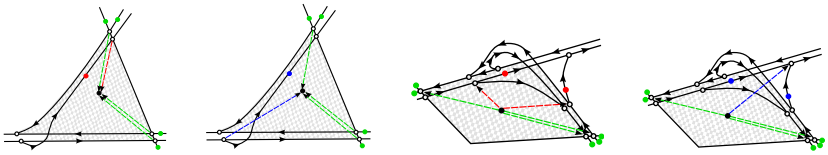
### 4.4.2 Adaptation 2: easing the node promotion condition

#### Problem

In a transformed road network graph, MLHNP node promotion takes place only if starting node $n_1$ or end node $n_2$ are missing in graph $G'$. In a graph with dual carriageways, this can result in paths with extra costs which can easily be avoided. This is shown in Figure 4.10. The shortest path from A to B starts and ends at higher level edges, while it consists of lower level links in the neighbourhoods of A and B.

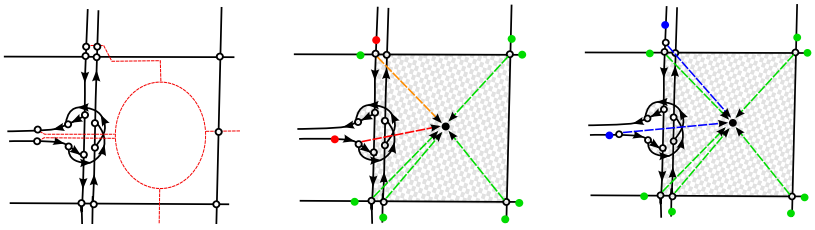
#### Solution

This issue is compensated by a change of the node promotion conditions, as proposed in Table 4.5. It involves node promotion for each recursion with $l_{exec} > 0$, producing virtual links from the global start and to the global end. This intervention enables MLHNP to take into account the lower level neighbourhood around the global start and end node, as illustrated in Figure 4.10b and 4.10c. From now on, the MLHNP algorithm in Figure 4.2 no



(a) Incorrect (b) Correct (c) Incorrect (d) Correct

Figure 4.11: Transition point corrections by length and by couple. The basic legend is in Table 4.3. Arrows represent the virtual links created from/to any of the entry/leave points to/from a point within the cell. Green arrows refer to correct, red to incorrect and blue to corrected transition points. The dots mark the witness-edges of the transition points.



(a) Network graph (b) Incorrect (c) Correct

Figure 4.12: Transition point correction in case of early access/late exit. The basic legend is in Table 4.3. Arrows represent the virtual links created from any of the entry points to a point within the cell. Green arrows refer to correct, orange to partially correct, red to incorrect and blue to corrected transition points. The dots mark the witness-edges of the transition points.

longer requires $\{s_{global}, e_{global}\}$ as the exception list for the intermediate node reduction of $G_{l_{exec}}$.

### 4.4.3 Adaptation 3: transition point corrections

#### Problem

Despite Adaptation 1, some of the transition points are still not on locations that support the effectiveness of the MLHNP algorithm. This can be caused by (a) incorrect classification of certain regular, DCW part or CI part cells, (b) the inability of the cell merge algorithm to merge a cell with the complete CI or

DCW along its boundaries, or (c) the fact that a very commonly used connection with the inside of a cell is located outside the cell, as illustrated in Figure 4.12b. Note that issue (b) is not unlikely to occur because the merge algorithm does not support self-tangent or multiple polygons as cell representation (Figure 4.11a), or because the mechanism that prevents the merge of DCW parts located along contiguous cells is too restrictive (Figure 4.11c). We identified three anomalous transition point patterns of a cell $c$.

1. *A cell $c$ does not have any entry (leave) point located on a complex intersection along the cell border.* Figure 4.11a is an example of this pattern. MLHNP becomes less effective with $l_{exec} = l(c)$ in this case because it does not allow that a path enters (leaves) the lower level graphs of $c$ via the missing intersection.

2. *The choice of an entry (leave) point constrains the subsequent (preceding) path direction on the cell border.* This has been exemplified in Figure 4.6a, 4.6b and 4.6c, and occasionally still occurs, as illustrated in Figure 4.11c. This makes MLHNP less effective because the choice of transition point at the higher level brings along decisions that should be made at the lower level.

3. *A transition point is located between its cell and a very commonly used connection with the cell.* The network in Figure 4.12a contains examples of this pattern for both the late exit and the early access case. It is shown for the latter case in Figure 4.12b. In both cases, MLHNP will never allow routes from/to this cell using this connection, whereas a significant number of shortest paths starting (ending) in this cell and ending (starting) in a non-contiguous cell pass through this connection.

### Solution and realisation

Both pattern 1 and 2 can be countered by moving entry points one step backwards and leave points one step further, out of the cell area, under certain conditions. We cannot simply apply this correction to any transition point. This would increase the chance of generating paths of higher cost. It makes the predictions returned by the cost estimate function more inconsistent, and causes transition point conflicts in case the algorithm is called between nodes of cells that have a common neighbouring cell. Consider a set of witness-edges of a leave point n: $\{\forall e \,\widetilde{\in}\, reduced(G_{l(c)}) : from(e) = n \wedge to(e) \notin c\}$ and of an entry point n: $\{\forall e \,\widetilde{\in}\, reduced(G_{l(c)}) : to(e) = n \wedge from(e) \notin c\}$. A transition point $t$ of cell $c$ is moved when

| | Concept | New definition |
|---|---|---|
| **Node promotion** | $leavepoints(c)$ | $witness_{lp}(c) = \{\forall e \widetilde{\in} reduced(G_{l(c)})\vert$ $\exists n \in c : from(e) = n \wedge to(e) \notin c\}$<br><br>$outer_{lp}(c) = \{\forall e \in witness_{lp}(c)\vert$ $(length(difference(e,c)) < length(e)/2) \vee (\exists e_2 \in$ $witness_{lp}(c) : e_2 \neq e \wedge to(e_2) =$ $to(e) \wedge length(difference(e,c) < \sigma_1)\}$<br><br>$leavepoints(c) = \{\forall n\vert\exists e \in outer_{lp}(c) : to(e) =$ $n\} \bigcup \{\forall n\vert\exists e \in witness_{lp}(c) \backslash outer_{lp}(c) : from(e) = n\}$ |
| | $entrypoints(c)$ | $witness_{ep}(c) = \{\forall e \widetilde{\in} reduced(G_{l(c)})\vert$ $\exists n \in c : to(e) = n \wedge from(e) \notin c\}$<br><br>$outer_{ep}(c) = \{\forall e \in witness_{ep}(c)\vert$ $(length(difference(e,c)) < length(e)/2) \vee (\exists e_2 \in$ $witness_{ep}(c) : e_2 \neq e \wedge from(e_2) =$ $from(e) \wedge length(difference(e,c) < \sigma_1)\}$<br><br>$entrypoints(c) = \{\forall n\vert\exists e \in outer_{ep}(c) : from(e) =$ $n\} \bigcup \{\forall n\vert\exists e \in witness_{ep}(c) \backslash outer_{ep}(c) : to(e) = n\}$ |

Table 4.6: Adaptation 3a

- *one of its witness-edges has a longer line length outside c than inside c.* In practice, this heuristic measure intercepts any instance of pattern 1. Figure 4.11b is an example of this correction.
- *there exist witness-edges of other (same type) transition points of c that have the replacement point in common with one of its own witness-edges which has a line length outside c smaller than* $\sigma_1$. Figure 4.11d is an example of this correction. In practice, this measure intercepts a reasonable number of instances of pattern 2. However, there exist cases of pattern 2 that cannot be corrected without generating other anomalous patterns or violating other constraints. Figure 4.13a shows an example of pattern 2 that requires moving the entry points towards the neighbours of the neighbours of the cell.

This first adaptation can be achieved by introducing a correction step to the preprocessing procedure, or by changing the transition point generation step, as suggested by the new concept definitions in Table 4.6.
Concerning pattern 3, both early accesses and late exits (EA/LE) can be intercepted by moving the transition point to the EA/LE point. An EA/LE node of a cell of level L usually is an intermediate node in $G_L$ and thus not contained in $reduced(G_L)$. Therefore the new transition point can only be used when the intermediate node is reintroduced in $reduced(G_L)$.

| | New concept | Definition |
|---|---|---|
| **Cell** | $parent(c_2)$ | the singleton of the cell $c_1$ that was used to generate $c_2$ during the iterative top-down cell generation process i.e. the only cell of level $l(c_2)+1$ that contained $c_2$ before the cell classification and merge processes. As CI and DCW part cells have been merged with regular cells, this function is applied exclusively to regular cells. It is possible that $c_1$ does not completely contain $c_2$ anymore or that it has been classified as non-regular and therefore does not exists anymore. |
| | $has\_siblings(c)$ | $\Leftrightarrow \exists c_s : parent(c_s) = parent(c) \wedge c_s \neq c$ |
| | $ancestors(c)$ | $= parent(c) \cup ancestors(parent(c))$; $ancestors(\emptyset) = \emptyset$ |
| **Node promotion** | $early\_accesses(c_a)$ | $on\_backlink(n_1, n_2, l, e) \Leftrightarrow e \,\widetilde{\in}\, reduced(G_l) : to(e) = n_2 \wedge intersects(n_1, e)$<br>$has\_forwardlink(n_1, l) \Leftrightarrow \exists e \,\widetilde{\in}\, G : from(e) = n_1 \wedge l(e) = l$<br><br>$early\_accesses(c_a) = \{(n_c, n_a, e) \mid \exists c_c : c_a \in ancestors(c_c) \wedge n_c \in entrypoints(c_c) \backslash entrypoints(c_a) \wedge n_a \in entrypoints(c_a) \wedge distance(n_c, n_a) < \sigma_2 \wedge \exists e : on\_backlink(n_c, n_a, l(c_a), e) \wedge has\_forwardlink(n_c, l(c_c)) \wedge has\_siblings(c_c)\}$ |
| | $late\_exits(c_a)$ | $on\_forwardlink(n_1, n_2, l, e) \Leftrightarrow e \,\widetilde{\in}\, reduced(G_l) : from(e) = n_2 \wedge intersects(n_1, e)$<br>$has\_backlink(n_1, l) \Leftrightarrow \exists e \,\widetilde{\in}\, G : to(e) = n_1 \wedge l(e) = l$<br><br>$late\_exits(c_a) = \{(n_c, n_a, e) \mid \exists c_c : c_a \in ancestors(c_c) \wedge n_c \in leavepoints(c_c) \backslash leavepoints(c_a) \wedge n_a \in leavepoints(c_a) \wedge distance(n_c, n_a) < \sigma_2 \wedge \exists e : on\_forwardlink(n_c, n_a, l(c_a, e)) \wedge has\_backlink(n_c, l(c_c)) \wedge has\_siblings(c_c)\}$ |
| | $entrypoints'(c)$ | $= \{n_c \mid \exists (n_c, n_a, e) \in early\_accesses(c)\} \cup \{n_a \mid n_a \in entrypoints'(c) \wedge \neg(\exists (n_c, n_a, e_1) \in early\_accesses(c) \wedge \neg(\exists e_2 : e_2 \,\widetilde{\in}\, reduced(G_{l(c)}) \wedge to(e_2) = n_a \wedge e_1 \neq e_2))\}$ |
| | $leavepoints'(c)$ | $= \{n_c \mid \exists (n_c, n_a, e) \in late\_exits(c)\} \cup \{n_a \mid n_a \in leavepoints'(c) \wedge \neg(\exists (n_c, n_a, e_1) \in late\_exits(c) \wedge \neg(\exists e_2 : e_2 \,\widetilde{\in}\, reduced(G_{l(c)}) \wedge from(e_2) = n_a \wedge e_1 \neq e_2))\}$ |
| | $reduced'(G_L)$ | $= reduced(G_L, \{(n_c, n_a, e) \mid \exists c : l(c) = L \wedge \exists (n_c, n_a, e) \in early\_accesses(c) \cup late\_exits(c)\})$ |

Table 4.7: Adaptation 3b

An EA/LE pattern is detected by the following heuristic. An EA/LE point typically connects with a cell $c_a$ of level $L$ by a way of a level $M$, with $0 < M < L$. Moreover, this node of $G_M$ belongs to a CI, which has been merged by a cell $c_c$ during the cell merge procedure, and this point typically is a transition point of $c_c$. Pattern 3 can efficiently be detected by matching the transition points $n_a$ of a cell $c_a$ and the transition points $n_c$ of the cells $c_c$ that originally were generated by splitting $c_a$. These points form an EA(LE) pattern when:

- the distance between $n_c$ and $n_a$ is lower than $\sigma_2$,
- $n_c$ is located on an edge in $reduced(G_L)$ ending/starting in $n_a$,
- there is an edge in $reduced(G_M)$ of exactly level $M$ starting/ending in $n_c$, and,
- $c_c$ has sibling cells i.e. there must exist a way of level $M$ that crosses $c_a$.

This pattern covers all instances of pattern 3 identified so far. Figure 4.12 shows two examples of an EA pattern and their corrections. Table 4.7 shows the extension of preprocessing by an EA/LE correction step. It assumes that from now on the MLHNP algorithm uses the concepts $entrypoints'(c)$, $leavepoints'(c)$ and $reduced'(G_L)$ instead of the concepts without apostrophe.

### 4.4.4 Adaptation 4: improving the transition point selection

#### Problem

Given any shortest path starting and ending in two non-contiguous cells of level $L$, MLHNP is able to find the shortest path if (a) all of its edges outside the cells belong to $G_L$, and (b) it chooses the transition points that generate this path. Currently, the transition point choice is based on the minimum of the sum of the cost of the candidate higher level path and the cost estimation(s) of the lower level path (Section 4.3.2). In certain cases, this choice may not be the optimal one, because the cost estimation is inconsistent with the real lower level cost. The inconsistency risk is notably higher when anomalous transition points of pattern 2 are involved. Figure 4.13e and 4.13f illustrate the wrong choice of transition point given the transition point situation in Figure 4.13a. The current adaptation addresses the algorithmic improvement of the transition point choice.

#### Solution and realisation

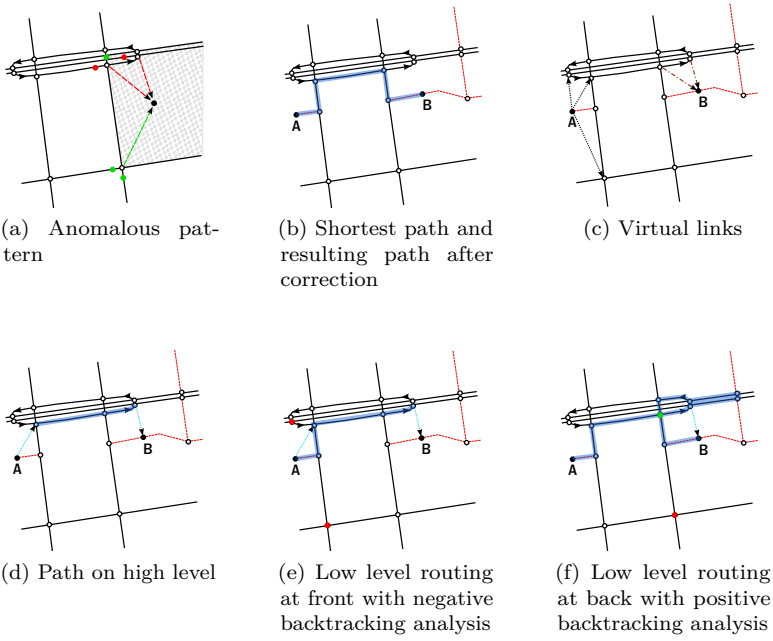
**Higher level correction** applies when the non-virtual part of any resulting path $R$ in the algorithm in Figure 4.2 passes any of the promoted transition points. This symptom indicates that another transition point was used to enter or leave the cell border. In this case $R$ is replaced by the path that would result from the choice of the transition point in which the original non-virtual path passes. This intervention has few risks because the lower level routing is able to generate the removed subpath on the cell border.
The applicability of the **lower level correction** depends on the ability of the algorithm behind the basic shortest path calculation to generate shortest paths from or to the other transition points after calculation. If they were all known, MLHNP would not be the preferred hierarchical strategy in general. This correction only applies if the algorithm knows some of them. The Dijkstra algorithm, for example, is able to produce all shortest paths from/to the other transition points that have a lower cost than the path from $a$ to $b$. Suppose that MLHNP has solved a virtual link at execution level $m$, that was created at execution level $l > m$. If it detects a shortest path from or to a nearby (distance lower than $\sigma_3$) transition point generated at level $l$ with a significantly lower cost than the current solution of the virtual link, the algorithm tracks back to level $m$, replaces all virtual links by the calculated paths so far and recalculates the shortest path. If the calculated paths still contain virtual links, these are solved at a level lower than $m$. This backtracking mechanism is illustrated in Figure 4.13.

## 4.5 Experiment

### 4.5.1 Objectives and methodology

The goal of this experiment is to validate the MLHNP approach for a database containing a real road network of a considerable size. In addition, the experiments have been set up to measure the performance for each of the adaptations presented in the previous section on a representative test set. A test set consists of $n$ routing queries represented by a start and end node, which are part of the road network graph $G$. Performance on a test set is expressed in terms of averages or percentiles of the following criteria for the routing operations generated by all queries in the test set. Note that a single MLHNP run may consist of several recursive runs.

- The **relative error**, as defined in Table 4.1. It reflects the quality of the




(a) Anomalous pattern

(b) Shortest path and resulting path after correction

(c) Virtual links

(d) Path on high level

(e) Low level routing at front with negative backtracking analysis

(f) Low level routing at back with positive backtracking analysis

Figure 4.13: Example of an uncovered anomalous transition point pattern (leftmost subfigure) and the backtracking mechanism. The basic legend is in Table 4.3. In the leftmost subfigure, arrows represent the virtual links created from any of the entry points to a point within the cell. Green arrows refer to correct and red to incorrect transition points. The dots mark the witness-edges of the transition points. In the two rightmost subfigures, red/green dots indicate transition points subject to a positive/negative backtracking analysis.

resulting route. The number of optimal solutions found is equal to the number of 0.00% deviations in the test set.

- The **number of recursive runs.**
- The **number of loaded nodes** is the sum of the number of graph nodes that are loaded from the database over all recursive runs.
- The **number of visited nodes** is the sum of the number of graph nodes for which the basic routing algorithm has calculated a cost, over all recursive runs. If the same node has been visited in $n$ runs, it is counted $n$ times. It indicates the search space reduction of our approach.

- Two criteria depend on the hardware and software infrastructure[3] on which the routing is executed. The **calculation time** is the sum of the basic routing algorithm's computation time over all recursive runs. It is strongly related to the number of visited nodes. The **memory footprint** is the maximal amount of memory consumed by the routing process. Since most of the loaded data is flushed before new data is loaded in a new run, this criterion only depends on the most memory-intensive run.

A test set of routing queries was generated as follows. First, an orthonormal grid was created over the routable area. The grid distribution is assumed representative for routing for tourism and leisure purposes. Then, the generator creates start and end nodes by selecting random grid points. Once a node has been picked, it cannot occur in any other query. Before routing, the grid nodes are replaced by the closest node in the road network graph.

### 4.5.2 Geographical dataset

The network studied here is the time-weighted road network of Belgium, divided in 5 hierarchical levels. The edge weights were set as the edge distance multiplied by a discrete speed class value. The test set used in this experiment contains 1000 routing queries, generated from an orthonormal grid (approx. 4 km) superimposed on the routable area. The road network graph consists of 1190220 directed edges and 526253 nodes. Preprocessing results in 1357/515/211/37 cells of level 1/2/3/4 respectively, after a cell merge with a reduction rate of ca. 91%. The total numbers of transition points per level are 22693/8493/3296/572 (22564/8390/3175/466 after Adaptation 3).

### 4.5.3 Basic wayfinding and MLHNP settings

MLHNP applied Dijkstra as the basic shortest path algorithm in this experiment. In order to speed up the search process, we use a rectangular pruning area parallel to the axes of the coordinate system (Karimi et al. [56]) supported by the indexing structure of the spatial DBMS. As the infrastructure diagram in Figure 4.1 may suggest, each basic shortest path calculation is preceded by a fetch of all links of $G'$ that start within the rectangular pruning area from the database. The size and position of this area is determined by the following rules.

[3] All tests have been executed using PHP (Command Line Interface) 5.3.3 and a PostGis database on a Debian 6.0 environment with an Intel Xeon E5520 @ 2.27GHz processor.

- Construct the rectangle formed by start node $a$ and end node $b$.
- Extend the rectangle such that it contains any virtual edge in $G'$.
- Extend the rectangular by $e\%$ equivalent meridian degree in both directions.
- If the rectangle is too elongated for the current level i.e. $min(breadth, height)/max(breadth, height) < min(r_{level} \cdot (l_{exec} \cdot 2 + 1), r_{max})$, elongate the smallest sides such that this condition does not hold.
- When the Dijkstra algorithm returns an infinite cost, it means that the pruned area does not contain any path from $a$ to $b$. In this case, the rectangular selection is extended by $e\%$ digital equivalent meridian degree again. After a cost reparation process, the Dijkstra algorithm is continued. This extension is repeated until the algorithm finds a solution.
  In this experiment, the road network graph $G$ (and also any directed subgraph $G_L$) is not a strongly connected graph. Therefore, the number of repetitions should be limited. When this number is reached in any of the recursive runs, MLHNP is halted and does not generate a solution, except in Adaptation 4. In this version, a subcall to the basic shortest path calculation is able to return an infinite cost path which is taken into account for a lower level correction.

This heuristic approach tends to minimise the probability that the path with the lowest cost is not located in the rectangle. It is tailored to the hierarchical approach. Two rectangular pruning settings are tested during the experiment: the *small box* setting with $e = 6$, $r_{level} = 0.1$ and $r_{max} = 0.6$ and the *large box* setting with $e = 12$, $r_{level} = 0.2$ and $r_{max} = 1$.
MLHNP's average resistance $r_{avg}$ has been set to 4. The settings of Adaptation 3 and 4 are $\sigma_1 = 3km$, $\sigma_2 = 1km$ and $\sigma_3 = 2.5km$.

### 4.5.4 Results

Table 4.8 and 4.9 show the routing performance after the stepwise introduction of each of the MLHNP adaptations over the testset presented in Section 4.5.2, for both rectangular pruning settings. The optimal costs (on which the error rate is based) have been obtained by the application of a Dijkstra algorithm without pruning involved. Three out of 1000 queries in the test set have start and end nodes that are not connected in the road network graph. These have not been taken into account in the statistics, neither have the queries that failed because one of the MLHNP recursive runs could not find a finite path. Their

| Small box pruning | | | | |
|---|---|---|---|---|
| | **Adapt. 1** | **Adapt. 2** | **Adapt. 3** | **Adapt. 4** |
| **# solutions not found** | 3 | 0 | 1 | 0 |
| **1% pctl calc. time (s)** | 0.027 | 0.036 | 0.037 | 0.039 |
| **avg calc. time (s)** | 0.161 | 0.169 | 0.158 | 0.173 |
| **99% pctl calc. time (s)** | 0.732 | 0.727 | 0.563 | 0.566 |
| **avg mem.footprint (MB)** | 19.286 | 19.496 | 19.081 | 18.342 |
| **avg # routing runs** | 4.243 | 4.408 | 4.514 | 5.848 |
| **avg # loaded nodes** | 6966.03 | 7337.83 | 7461.49 | 8621.27 |
| **1% pctl # visited nodes** | 628.30 | 852.84 | 884.50 | 924.88 |
| **avg # visited nodes** | 3591.65 | 3773.92 | 3660.50 | 4123.67 |
| **99% pctl # visited nodes** | 10902.66 | 11018.52 | 9607.15 | 10192.68 |
| **# exact solutions** | 137 | 145 | 192 | 240 |
| **avg error rate** | 11.85% | 11.08% | 8.81% | 7.43% |
| **99% pctl error rate** | 77.76% | 66.89% | 58.09% | 52.74% |
| **# error decr. / increase** | - | 72 / 25 | 300 / 117 | 274 / 23 |

Table 4.8: Experiment results (small box pruning). Any average/percentile number is only based on the found solutions.

| Large box pruning | | | | |
|---|---|---|---|---|
| | **Adapt. 1** | **Adapt. 2** | **Adapt. 3** | **Adapt. 4** |
| **# solutions not found** | 3 | 0 | 1 | 0 |
| **1% pctl calc. time (s)** | 0.033 | 0.040 | 0.042 | 0.050 |
| **avg calc. time (s)** | 0.259 | 0.279 | 0.263 | 0.295 |
| **99% pctl calc. time (s)** | 1.051 | 1.081 | 0.900 | 1.057 |
| **avg mem.footprint (MB)** | 28.830 | 29.347 | 29.296 | 28.970 |
| **avg # routing runs** | 4.2263 | 4.394 | 4.503 | 5.821 |
| **avg # loaded nodes** | 16019.19 | 16831.94 | 17305.26 | 20526.69 |
| **1% pctl # visited nodes** | 864.00 | 1050.56 | 1065.70 | 1173.72 |
| **avg # visited nodes** | 5239.80 | 5498.72 | 5401.31 | 6103.70 |
| **99% pctl # visited nodes** | 15256.91 | 15719.72 | 14617.90 | 15990.08 |
| **# exact solutions** | 167 | 175 | 234 | 300 |
| **avg error rate** | 9.44% | 8.73% | 6.47% | 4.91% |
| **99% pctl error rate** | 63.58% | 53.05% | 50.11% | 36.82% |
| **# error decr. / increase** | - | 63 / 23 | 301 / 114 | 291 / 17 |

Table 4.9: Experiment results (large box pruning). Any average/percentile number is only based on the found solutions.

numbers are given in the first row. The large box pruning setting yield lower error rates than the small box setting, at the expense of the average number of visited nodes. As expected, larger boxes extend the search space but small boxes may cut away solutions of lower cost. Below, only the large box pruning numbers are given.

Three out of 997 queries failed in the case of **Adaptation 1**. Each of these three queries fails at the top recursion with execution level $L_{top}$. The queries have a start/end node located on a motorway, which, in $reduced(G_{L_{top}})$, only connects to a long edge ending/starting at the data set border, whereas paths from the start node to the rest of the graph exist in $G$. This issue is fixed by the next adaptation. Adaptation 1 yields a trade-off between an average number of visited nodes of 5239.80 and an error rate of 9.44%.

The application of **Adaptation 2** improves the solution quality by 8.73%. The node promotion condition relaxation introduces an extra number of recursive runs, reflected in the average number, leaving the other runs roughly as they are. This causes a higher number of visited and loaded nodes. Some individual solutions show a higher error rate because a higher number of node promotions increases the risk of choosing a wrong transition point.

**Adaptation 3** is the most effective adaptation. It lowers the number of visited nodes, whereas the error rate is reduced to 6.47%. Most of the transition point corrections move the points away from their cells. This could explain the higher average number of runs[4] and loaded nodes. Before the introduction of Adaptation 3, malicious transition point locations often caused detours in the lower level paths. The elimination of these detours probably decreases the number of visited nodes. For 1 query, an entry point was selected that only connects to a motorway dead end in $G$, making the query fail.

**Adaptation 4** further reduces the error rate to 4.91%. Higher level correction generates a higher average number of runs (for the same reason as Adaptation 3). Lower level correction introduces an extra number of reparation runs. Therefore, both corrections increase the average number of loaded and visited nodes. Some queries generate worse solutions for this adaptation than for the previous one. This occurs in case of higher level correction because the risk of choosing a wrong transition point at the lower level increases. It also occurs in case of lower level correction when the replacing path contains again a virtual link, which has an estimate-based cost.

[4]For instance, suppose a front-side virtual link from $s_{global}$ to $lp \in leavepoints'(c)$, which is selected in path $R$. When $lp$ is moved out of $c$, the level of execution in the recursive run $mlhnp(s_{global}, lp, l_{restr}, G)$ is on average higher, because on level $l_{restr}$, the (previously same) cells of $s_{global}$ and $lp$ become neighbours.

## 4.6 Conclusion

We have introduced the MLHNP algorithm, enabling finding the shortest path in hierarchically organised road network graphs, while supporting several types of edge weights. The authors of the two-level version of this algorithm have suggested a graph transformation in order to cope with dual carriageways and slip roads in real road networks. This transformation does not fully comply with the requirements of contemporary routing applications.
Therefore, four adaptations of the MLHNP approach have been proposed that bypass the graph transformation. The first adaptation introduces a cell classification and merge during preprocessing. The classification method can be more generally used for functional road class prediction in GIS. The hierarchical routing algorithm uses the resulting cell set to emulate contiguity and transition points as if the network graph were subject to the original graph transformation. In the here presented application domain, an effective cell merge is to be preferred over cell grouping because it reduces storage costs and optimises the routing data indexation. Adaptation 2 makes the algorithm take into account the local context of dual carriageways and slip roads when the start or end node is located on these types of way. The third adaptation fine-tunes the transition point locations of the first one. A last adaptation enables the algorithm to revise its transition point choices throughout the recursion tree. It applies to hierarchical node promotion algorithms in general.
The adapted MLHNP approach has been designed for a web application that uses a spatial database and that enables hierarchical routing for several routing modi. Here, database operations have a high impact on routing performance. Both storage costs and calculation time of preprocessing are restricted.
The MLHNP algorithm has been successfully applied to a medium-sized hierarchical road network with time-weights, using Dijkstra and rectangular pruning for individual routings. The experiment showed the effectiveness of each of the adaptations. The combined MLHNP and pruning approach achieves a low average number of visited nodes, maintaining an average relative error of 4.91%.

The present chapter presented the successful integration of a top-down hierarchical shortest path algorithm in an environment where database operations heavily burden the performance of the calculation process. This subject originated from a concrete question from industry tackling the design of a shortest path approximation component in a multi-tier architecture. The next chapter continues the theme of the application of shortest path (cost) approximation algorithms in resource-constrained environments. It introduces a component which approximates the cost of the shortest path in instant time, using low amounts of precomputed data.

# Chapter 5

# Least squares approximate distance oracles for spatial networks

An *approximate distance oracle* is a compact data structure, which is generated from a graph in low computational time. Once it is constructed, it returns network distance approximations between any two vertices of the graph in instant time. Distance oracles are of particular interest for applications that require real-time network distance approximations and only have limited time to build or space to store a predictive model. A conventional distance oracle employs a guaranteed upper and lower bound to the relative error of the approximation and often pegs down its required size on these bounds. These bounds do however not clearly represent the overall performance of the oracle, and the relaxation of these bounds can produce a reduced complexity of the precalculation time, the oracle space and the query time. The present chapter introduces the concept of a distance oracle that minimizes the root-mean-square error of its network distance approximations. An appropriate distance oracle of size $O(1)$ is presented for both general and spatial networks. A more advanced least squares approximate distance oracle is introduced for spatial networks. It is based on clustering graph vertices that share similar shortest paths starting or ending in these vertices. Its prediction accuracy is demonstrated in the context of travel time approximation in a transportation network for vehicle navigation.

The advanced ADO presented in this chapter was designed, developed and evaluated by Jan Christiaens. This work is available as a technical report.

## 5.1 Introduction

As introduced by Thorup and Zwick [109], an *approximate distance oracle* (ADO) is a data structure that immediately answers network distance queries between any two vertices of a graph. In contrast to data structures arising from precomputing the distances between any pair of vertices of the graph, ADOs require limited space and can be built in limited computational time. Real-time distance approximation is of particular interest for applications such as calculating the distance between two people in a social network or retrieving the distances from the current location to a set of alternative locations [95], e.g. restaurants, in a transportation network. In these domains an estimate of the network distance is often sufficient.
Many combinatorial optimization problems from logistics, public transportation and tourism, require a transportation cost matrix for a set of locations. The vehicle routing problem [63], for instance, is about finding the path of lowest cost from one or several depots visiting a set of customer locations. The goal of the orienteering problem [111] is to find a path, of a travel cost below a given threshold and along a selection of locations, maximizing the sum of collected location scores. The matrix mentioned above contains the transportation cost, often travel time or fuel cost, between any origin-destination pair of the problem in a transportation network. Using an ADO to determine this matrix is advantageous when the problem's locations often change and the transportation cost calculation must be offered as a portable software component.

We now introduce the following definitions. A graph, further also referred to as a general network, is denoted $G = (V, E)$, where $V$ is the set of vertices and $E$ the set of edges of the graph. The number of vertices $|V| = n$ and the number of edges $|E| = m$. By default, $G$ is a *weighted* graph, which means that any edge $e \in E$ has a weight $w(e) > 0$. A path between the vertices $u$ and $v$ in $G$ is a sequence of edges consecutive in $G$, starting in $u$ and ending in $v$. Its cost equals the sum of the edge weights. The *network distance* $d_G(u, v)$ is the cost of the path of lowest cost between $u$ and $v$ in $G$. A spatial network is a general network where each $v \in V$ corresponds to a position $p(v) \in \mathbb{R}^d$. The *spatial distance* $d_S(u, v)$ is a function of $p(u)$ and $p(v)$ in $\mathbb{R}^d$. An Euclidean network is a spatial network where the weight of any edge $e$ connecting to vertices $u$ and $v$ is defined as $w(e) = d_S(u, v)$, with $d_S(u, v)$ taking the Euclidean distance between $p(u)$ and $p(v)$. Distances in this type of network satisfy the triangle inequality i.e. $\forall u, v, w \in V : d_G(u, w) \leq d_G(u, v) + d_G(v, w)$, and the network distance has a lower bound: $\forall u, v \in V : d_G(u, v) \geq d_S(u, v)$.

In any of these networks, an ADO is a data structure or model that answers network distance queries $d_G(u, v)$ by a *reported distance* $d_R(u, v)$. Conventional ADOs assume a guaranteed upper and lower bound to the error of the reported

distance. In this context, a distance oracle has a *stretch factor* $\alpha$ when $\forall u, v \in V : d_G(u,v) \leq d_R(u,v) \leq \alpha \cdot d_G(u,v)$. Alternatively, an ADO is called $\epsilon$-approximate when the relative error of any approximation generated by the oracle is not more than $\epsilon$ i.e. $\forall u, v \in V : abs(d_G(u,v) - d_R(u,v))/d_G(u,v) \leq \epsilon$. Thorup and Zwick [109] showed for any integer $k$, that building an oracle of stretch $2k-1$, answering queries in $k$ time, requires a space of at least $n^{1+1/k}$. Sommer et al. [100] argue that this bound does not provide useful information for sparse graphs, and prove for any ADO that a space of at least $n^{1+\Omega(1/t\alpha)}/lg(n)$ is required to build an oracle of stretch $\alpha$ and query time $t$. Network distance approximation for spatial networks can be seen as fine-tuning "as the crow flies" distances towards network distances. ADOs for spatial networks exploit the spatial coherence of source and destination vertices of similar shortest paths. Sankaranarayanan and Samet [95] introduce an $\epsilon$-approximate oracle of this type based on well-separated pair decomposition in a $d$-dimensional space. The oracle's space requirements are $O(n/\epsilon^d)$ answering queries in time $O(log(n))$. Sankaranarayanan and Samet [95] conducted an experiment on a publicly available US transportation network with distance-weights. It confirmed the linear storage requirements and yielded an average relative error of 0.9% for $\epsilon = 0.1$, which is 10%. With regard to Euclidean networks, Gudmundsson et al. [43] refer to a number of solutions based on polyhedral surfaces and obstacles. They notice that none of these can be used to build an ADO of subquadratic space answering queries in constant time. They present an oracle for Euclidean networks with $m = O(n)$ of size $O(n \cdot log(n))$ and query time $O(1)$. It is based on partitioning the graph into a set of clusters with a fixed radius. Mainly for networks satisfying the triangle inequality, many variants of the *landmark embedding* technique have been proposed. It involves that a set of landmarks $R$ is selected from $V$ and that the network distances between any vertex and $V$ and one or more vertices in $R$ are computed. Real-time network distance approximation is based on these precomputed distances. In the ADO proposed by Qiao et al. [88], each landmark covers the set of vertices located within a given radius. This radius can be linked to the upper bound of the absolute approximation error. They show that finding the minimal set of landmarks is NP-hard. Qiao et al. [87] improve the approximation accuracy significantly by looking up the *least common ancestor* (LCA) of the two queried vertices i.e. the last vertex shared by the paths of lowest cost starting in a global landmark, in efficient indices.

In what follows, we raise several concerns in regard to the practical application of ADOs.

1. The basic assumption in many ADOs is a guaranteed upper and lower bound to the relative approximation error, whereas many applications only require a low average error and a low error variance. Although the

latter is a weaker condition than the first, the latter condition is a better indicator of the overall accuracy of the oracle. This duality manifests itself in the fact that experimental approaches in the field of ADOs starting from guaranteed error bounds end up in reporting the average error (e.g. [95]). Qiao et al. [88] showed that the relaxation of these bounds results in a reduced complexity of the precalculation time, the oracle space and the query time.

2. Some applications require a minimal absolute approximation error, instead of a minimal relative one. This is the case when the transportation matrix for the traveling salesperson problem (TSP) is generated by an ADO. The stability regions [69] of the TSP edge lengths, i.e. the length domains within which the optimal sequence of nodes is identical, better fit an absolute than a relative deviation pattern.

3. Many popular applications of network distance approximation apply to transportation networks with time weights, supporting travel time estimation of the **'fastest'** path between two vertices. These spatial networks do not satisfy the triangle inequality, which is assumed to hold or applies to the evaluation network in most ADO research. The ADO for spatial networks in general by Sankaranarayanan and Samet [95] is solely validated by experiments on distance-weighted transportation networks. To the best of our knowledge, not any ADO designed for spatial networks has been validated on time-weighted transportation networks before.

In order to evaluate an oracle based on the average error and its variance (concern 1), Section 5.2 formalizes the concept of an ADO minimizing the root-mean-square error (RMSE) of its network distance approximations. The rationale of this concept is to abandon the guarantee-based ADO design to increase the overall approximation accuracy (depending on the intrinsic characteristics of the network) or to reduce the space complexity of the oracle. Considering the second concern, this concept is introduced for both relative and absolute deviation. Section 5.3 comprises an ADO design for spatial networks minimizing the absolute RMSE. This oracle is constructed by partitioning the network into clusters of vertices that share similar paths of lowest cost. In Section 5.4, the oracle's quality is evaluated for a time-weighted transportation network extracted from the OpenStreetMap project.

## 5.2 Least squares approximate distance oracles

The following definitions apply to weighted graphs in general as well as to *directed* weighted graphs in general. In the latter type of graph $G = (V, E)$,

any directed edge (or *arc*) $e \in E$ is defined as an *ordered* pair of vertices, corresponding to its associated direction. A path in a directed graph is a sequence of consecutive edges of forward direction.

### 5.2.1 Basic concepts

Given an ADO approximating a network distance query $d_G(u, v)$ by the *reported distance* $d_R(u, v)$ for any vertex $u$ and $v$ of a general network $G = (V, E)$. We define two inverse accuracy criteria, the absolute and the relative RMSE, as follows.

$$RMSE_{abs}(G) := \sqrt{\frac{\sum_{\forall u,v \in V, u \neq v} (d_G(u, v) - d_R(u, v))^2}{|V|^2 - |V|}}$$

$$RMSE_{rel}(G) := \sqrt{\frac{\sum_{\forall u,v \in V, u \neq v} \left[\frac{d_G(u,v) - d_R(u,v)}{d_G(u,v)}\right]^2}{|V|^2 - |V|}}$$

An oracle has an optimal approximation accuracy on a graph G, if the RMSE on G is minimal. These definitions however imply that the ADO accuracy evaluation requires an all-pairs shortest path approach, of which the processing time has a (nearly) cubic complexity: the time complexity of the classical Floyd-Warshall algorithm is $O(n^3)$; Chan's algorithm [17] requires $O(n^3 \cdot (log(log(n)))^3/(log(n))^2)$ time. Therefore, in practice, a representative query[1] sample set $S \subset V \times V$ is determined. $S$ has a random distribution in $V \times V$ or an expected query distribution for a certain application domain e.g. long-distance queries. The ADO's accuracy can be evaluated quickly based on the sampled RMSE on G as follows.

$$RMSE^S_{abs}(G) := \sqrt{\frac{\sum_{\forall (u,v) \in S} (d_G(u, v) - d_R(u, v))^2}{|S|}}$$

$$RMSE^S_{rel}(G) := \sqrt{\frac{\sum_{\forall (u,v) \in S} \left[\frac{d_G(u,v) - d_R(u,v)}{d_G(u,v)}\right]^2}{|S|}}$$

[1]This query is an unordered/ordered pair, since $d_G(u, v)$ is a symmetric/asymmetric function in undirected/directed graphs.

### 5.2.2 Oracles of unit size

The concept of an ADO of unit size was first coined by Sankaranarayanan and Samet [95], referring to an oracle that computes the network distance approximation in $O(1)$ time. We now introduce oracles of unit size for both the absolute and relative cases. These oracles are mainly intended as a reference test to compare the intrinsic characteristics of different networks, and their difficulty level of establishing an ADO. We expect for instance that the accuracy of a unit size oracle for spatial networks is considerably better on a distance-weighted than on a time-weighted transportation network. Both for general and spatial networks a unit size configuration based on a single constant is introduced. Next the constant value generating the highest accuracy for a query sample set $S \subset V \times V$ is determined. Straightforward mathematical methods produce the following constant values.

**Oracles of unit size in general networks.** An ADO of size $O(1)$ minimizing $RMSE^S_{abs}(G)$, responds to each query $d_G(u, v)$ by a constant distance $d_C$:

$$d_C = \frac{\sum\limits_{\forall(u,v)\in S} d_G(u, v)}{|S|}$$

Analogously, it can be shown that an ADO of size $O(1)$ minimizing $RMSE^S_{rel}(G)$ and answering to each query a constant distance $d_C$ requires

$$d_C = \frac{\sum\limits_{\forall(u,v)\in S} (1/d_G(u, v))}{\sum\limits_{\forall(u,v)\in S} (1/d^2_G(u, v))}$$

**Oracles of unit size in spatial networks.** The distortion for an individual vertex couple in a spatial network is defined as

$$\gamma(u, v) := \frac{d_G(u, v)}{d_S(u, v)}$$

Note that the maximum value of $\gamma(u, v)$ for any $u, v \in V$ is often referred to as the (maximum) distortion, the dilation or the stretch factor of $G$ [95, 82]. Sankaranarayanan and Samet [95] state that a *distortion spectrum* usually exhibits large distortion values only for low spatial distances. An ADO of size $O(1)$ for spatial networks entails the approximation of a query $d_G(u, v)$ by the product $\gamma_C \cdot d_S(u, v)$, where $\gamma_C$ represents a constant distortion. The following

condition yields an oracle of optimal approximation accuracy with regard to the absolute RMSE.

$$\gamma_C = \frac{\sum\limits_{\forall(u,v)\in S} (d_S^2(u,v) \cdot \gamma(u,v))}{\sum\limits_{\forall(u,v)\in S} d_S^2(u,v)}$$

An optimal ADO of the same configuration minimizing the relative RMSE requires

$$\gamma_C = \frac{\sum\limits_{\forall(u,v)\in S} (1/\gamma(u,v))}{\sum\limits_{\forall(u,v)\in S} (1/\gamma^2(u,v))}$$

## 5.3 An advanced ADO based on clusters and transit nodes

The classical landmark embedding techniques based on spatial coverage, described in Section 5.1, are less effective in networks that do not necessarily satisfy the triangle inequality. The core of the ADO introduced in the present section is therefore based on graph partitioning into disjoint clusters of vertices. Furthermore, a set of *transit nodes* of minimal size is determined for each of the clusters. The network distance approximation algorithm is based on the precalculated distances between the transit nodes. The design of this oracle is aimed at minimizing the absolute RMSE. Related approaches to graph partitioning are discussed in the next paragraphs.

### 5.3.1 Related work

Several graph clustering algorithms have been proposed optimizing different objectives. Both Monien and Diekmann [80] and Pothen [85] minimize the number of edges connecting different partitions. Edge length based clustering of edges in graphs was proposed by Das and Narasimhan [23] for constructing sparse spanners in complete Euclidean networks. The Markov cluster algorithm by van Dongen [110] minimizes the probability of leaving the cluster during a random walk. It is an iterative algorithm on a matrix of transition probabilities. The conductance of a graph indicates the number of steps a random walk in the graph requires for converging to a uniform distribution. Clusters of low conductance can be seen as bottlenecks. Iterative Conductance Cutting [54, 12] maximizes the conductance of these bottlenecks.

In the following cases, graph partitioning has been applied in order to obtain the path of lowest cost in graphs. This discipline is somewhat different from network distance approximation because it mainly focusses on returning the complete path of the exact result, which implies a different trade-off between space and query time complexity. Lansdowne and Robinson [62] were the first to apply the concept of spatial decomposition to the shortest path problem in sparse directed graphs. During query time they assign the vertices of the spatial graph to a set of regions in order to optimize the performance of the exact calculation of the $n$ paths of lowest cost. More recent approaches divide the complete graph into clusters during the preprocessing phase. This phase also implies that the partitioning description is stored together with a set of precomputed paths or distances between the clusters. During a shortest path query, this stored information is used in order to drastically reduce the search space of the path calculation procedure. Huang et al. [50] advocate the application of the Spatial Partition Clustering technique in order to minimize I/O costs in routing systems that require to load the edges from secondary storage to a main memory buffer. It partitions the arcs of a directed spatial graph such that the origin vertices of the arc of a partition are bound by a quasi-square polygon. The routing algorithm by Flinsenberg et al. [33] only considers the edges (1) belonging to the cells of the start and destination vertex, (2) connecting two so-called *boundary nodes* of different cells, and, (3) representing precalculated paths between boundary nodes of same cells. A hierarchical version of this algorithm has been presented by Jung and Pramanik [53]. Flinsenberg et al. [33] introduced the partitioning problem as finding the cell configuration that yields a minimal average number of loaded edges. Their preprocessing phase consists of several runs, repeatedly merging 1-vertex-cells until one cell of size $|V|$ remains. This greedy merge procedure is managed by a priority function containing a random element. The cell configuration over all the runs that suits the partitioning problem best is selected. When, however, the A* algorithm is applied to a transportation network, Flinsenberg et al. [33] discovered that minimizing the number of loaded edges does not result in the fastest query results, and reformulated the partitioning problem's objective value minimizing the algorithm's search space. Maue et al. [79] assign each node to the cluster of the closest node (with regard to the network distance) of a set of $k$ centre nodes. Their routing algorithm integrates pruning of complete clusters based on distance bounds. The resulting search space has the shape of a corridor around the shortest path, of which the narrowness is determined by the number of clusters $k$. Note that these approaches are different from recent successful partition-based approaches to exact shortest path calculation since the preprocessing phase starts from arbitrary graph partitions such as administrative divisions. In order to lower the number of precalculated paths, Bast et al. [7] search for the minimal set of *transit nodes* outside the partition,

such that any long-distant shortest path from/to the partition passes one of these nodes.

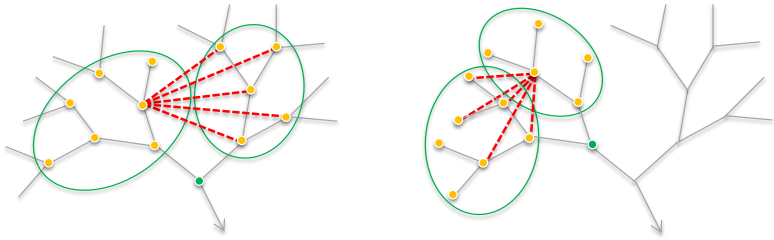
### 5.3.2 Definitions

We first introduce a few definitions supporting the description of the advanced ADO. A random sample of $k < |S|$ elements of the set $S$ is denoted $random(S, k)$. The *partition* of a set $S$ is a collection of pair-wise disjoint subsets of this set such that the union of these subsets equals $S$. A directed graph $G(V, E)$ is *connected* when for each pair of nodes $(u, v) \in V \times V$ there exists a path from $u$ to $v$. The *forward shortest path tree* of a connected directed graph $G(V, E)$ rooted at vertex $r \in V$ is a tree $T$ in $G$, such that, for each node $v \in V$ the downward path from $r$ to $v$ in $T$ corresponds with a path of lowest cost from $r$ to $v$ in $G$. The *backward shortest path tree* of a directed graph $G = (V, E)$ rooted at vertex $r \in V$ is a tree $T$ in $G$, such that, for each node $v \in V$ the upward path from $v$ to $r$ in $T$ corresponds with a path of lowest cost from $v$ to $r$ in $G$.

### 5.3.3 ADO construction

The construction of the advanced ADO from the connected directed spatial network $G(V, E)$ comprises stepwise generation of the following elements:

1. two partitions of $V$, of which $P_O$ consists of *origin clusters* and $P_D$ of *destination clusters*,
2. a set $T \subset V$ of minimal size, containing transit nodes for each cluster $C$ in $P_O$ and $P_D$ (we say $t \in T$ is a transit node of $C$),
3. the precalculated distances $d_G(s, t)$ for any transit node $s$ of an origin cluster and any transit node $t$ of a destination cluster (close cluster pairs excluded, see element 6),
4. the precalculated distances $d_G(u, s)$ between any $u$ in an origin cluster $C_O$ and any transit node $s$ of $C_O$,
5. the precalculated distances $d_G(t, v)$ between any transit node $t$ of a destination cluster $C_D$ and any $v$ in $C_D$,
6. a set of close cluster pairs $(C_O, C_D) \in P_O \times P_D$, and,
7. the constant distortions $\gamma_{C_O,C_D}$ for any close cluster pair $(C_O, C_D)$.



(a) (b)

Figure 5.1: Association graph construction. The subfigures show two consecutive runs of the recursive algorithm. The backward shortest path tree is indicated by a solid line. The association edge creation or counter increment (dashed lines) is only illustrated for one vertex. Ellipses delimit the subtrees with $maxDepth = 2$. Note that any pair of vertices is linked at most once during the processing of one tree.

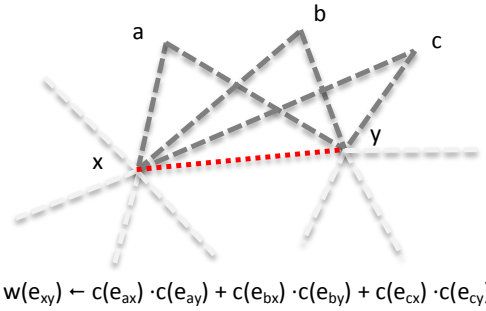



Figure 5.2: Association edge weight calculation example. The weight of the edge between vertices $x$ and $y$ is calculated as the sum of the counter products of the edges connecting $x$ and $y$ with a common vertex.

**Procedure** `GenerateOriginPartition`($G$)
  **input** : $G \leftarrow$ spatial network $G(V, E)$
  **output** : $P_O \leftarrow$ set of subsets of $V$
  **parameters** : *numberOfRoots*, *popularityThreshold*

  ▷ *construction of sample backward shortest path trees*;
  roots $\leftarrow$ `random`($V$, *numberOfRoots*);
  trees $\leftarrow$ tree set (initially empty);
  **foreach** $r$ *in* roots **do**
    tree $\leftarrow$ `backwardShortestPathTree`($G$,$r$);
    `insert`(trees, tree);

  ▷ *association graph construction*;
  $G_a \leftarrow$ association graph $G_a(V_a, E_a)$ (initially empty);
  ▷ *an association graph is an undirected graph where each edge $e$ has a weight* $\mathsf{w}(e)$ *and a counter* $\mathsf{c}(e)$*, and each vertex $v$ has a vote* $\mathsf{v}(v)$*, all equal to 0 by default*;
  **foreach** $t$ *in* trees **do**
    $G_a \leftarrow$ `generateAssociations`($G_a$, $t$, `root`($t$));

  ▷ *association edge weight calculation*;
  **foreach** *edge* $e_a$ *in* $E_a$ **do**
    $\{v_1, v_2\} \leftarrow$ `getVertices`($G_a$, $e_a$);
    **foreach** $e_1$ *in* `getEdges`($G_a$, $v_1$) **do**
      **foreach** $e_2$ *in* `getEdges`($G_a$, $v_2$) **do**
        **if** (`getVertices`($G_a$, $e_1$) $\bigcap$ `getVertices`($G_a$, $e_2$)) $\setminus\{v_1, v_2\} \neq \emptyset$ **then**
        $w(e_a) \leftarrow w(e_a) + c(e_1) \cdot c(e_2)$ ;

  ▷ *determination of principal nodes*;
  **foreach** *vertex* $v_a$ *in* $V_a$ **do**
    $\{v_h\} \leftarrow$ `getVertices`($G_a$, `edgeMaxWeight`(`getEdges`($G_a$, $v_a$))) $\setminus \{v_a\}$ ;
    $\mathsf{v}(v_h) \leftarrow \mathsf{v}(v_h) + 1$ ;
  principalNodes $\leftarrow$ a subset of $V_a$ (initially empty);
  **foreach** *vertex* $v_a$ *in* $V_a$ **do**
    popularity $\leftarrow \mathsf{v}(v_a)/|$`getEdges`($G_a$, $v_a$)$|$;
    **if** popularity $>$ *popularityThreshold* **then** `insert`(principalNodes, $v_a$) ;

  ▷ *cluster construction*;
  $P_O \leftarrow$ set of singletons of elements in principalNodes;
  **foreach** *vertex* $v_a$ *in* $V_a \setminus$principalNodes **do**
    $E_p \leftarrow$ subset of `getEdges`($G_a$, $v_a$) connecting to a vertex in principalNodes;
    **if** $E_p \neq \emptyset$ **then**
      $\{v_h\} \leftarrow$ `getVertices`($G_a$, `edgeMaxWeight`($E_p$)) $\setminus \{v_a\}$ ;
      `addToSubsetContaining`($P_O$, $v_a$, $v_h$);
  **while** $V_a \setminus (\bigcup_{C \in P_O} C) \neq \emptyset$ **do**
    **foreach** *vertex* $v_a$ *in* $V_a \setminus (\bigcup_{C \in P_O} C)$ **do**
      $E_c \leftarrow$ subset of `getEdges`($G_a$, $v_a$) connecting to a vertex in $\bigcup_{C \in P_O} C$ ;
      **if** $E_c \neq \emptyset$ **then**
        $\{v_h\} \leftarrow$ `getVertices`($G_a$, `edgeMaxWeight`($E_c$)) $\setminus \{v_a\}$ ;
        `addToSubsetContaining`($P_O$, $v_a$, $v_h$);
  **return** $P_O$;

**Algorithm 1:** Generation of the origin cluster partition. The set, tree and graph functions are explained in Table 5.1.

| | Function | Definition |
|---|---|---|
| **Sets** | $random(S, k)$ | returns a random sample of $k < \|S\|$ elements of the set $S$ |
| | $insert(S, e)$ | inserts $e$ in the set $S$ |
| | $addToSubsetContaining(S, e_{add}, e_{ref})$ | inserts $e_{add}$ in the subset of S containing $e_{ref}$ |
| **Trees** | $root(T)$ | returns the root of tree $T$ |
| | $depth(e, T)$ | returns the depth of element $e$ in tree $T$ (the root of $T$ has depth 0) |
| | $next(e, T)$ | returns the set of children of element $e$ in tree $T$ |
| | $getSubtrees(e, T, d)$ | returns the set of trees rooted at the child elements of $e$ in tree $T$ and chopped of at depth $d$ relative to the child elements |
| **Graphs** | $getVertices(G, e)$ | returns the set of 2 vertices connected by edge $e$ in graph $G$ |
| | $getEdges(G, v)$ | returns the set of edges connecting vertex $v$ in graph $G$ |
| | $edgeMaxWeight(E)$ | returns the edge $e$ that has the highest weight $w(e)$ in edge set $E$ |
| | $connected(G, v_1, v_2)$ | there exists an edge between $v_1$ and $v_2$ in graph $G$ |
| | $constructEdge(G, v_1, v_2)$ | inserts an edge between $v_1$ and $v_2$ in graph $G$ |
| | $backwardShortestPathTree(G, r)$ | returns the backward shortest path tree (defined in the definitions paragraph of Section 5.3) of graph $G$ rooted at vertex $r$ |

Table 5.1: Set, tree and graph functions

Element 1 implies clustering of vertices sharing many similar paths in the set of all paths of lowest cost starting (partition $P_O$) or ending (partition $P_D$) in these vertices. This similarity can be described best in terms of the LCA of two vertices in a forward/backward shortest path tree. Two nodes are similar when their LCA is only a few edges away for many paths of lowest cost. Algorithm 1 describes a sampling-based method of low computational time to generate the origin cluster partition $P_O$. It generates an association graph $G_a$ from a set of $numberOfRoots$ backward shortest path trees in $G$ rooted in a sample of $V$. This undirected graph registers in its edge counters how many times two vertices have a close LCA, over the set of trees. This registration (Algorithm 2) is realised for one tree $t$ by linking the vertices between any pair of subtrees, cropped at depth $maxDepth$, of any vertex located at least at depth $minDepth$ in $t$. This is shown in Figure 5.1. The edge weight calculation propagates the edge counters over triangle subgraphs of $G_a$, as illustred in Figure 5.2. Principal

```
Procedure generateAssociations(G_a, t, el)
    input      : G_a ← association graph G_a(V_a, E_a)
    input      : t ← tree of elements in V_a and branches in E_a
    input      : el ← an element in t
    output     : G_a ← association graph G_a(V_a, E_a)
    parameters : minDepth, maxDepth
    if depth(el, t) > minDepth then
        subtrees ← getSubtrees (el, t, maxDepth)
        foreach 2-element combination (tree_1, tree_2) in subtrees do
            foreach element el_x in tree_1 do
                foreach element el_y in tree_2 do
                    if not(connected(G_a, el_x, el_y)) then
                        constructEdge(G_a, el_x, el_y)
                    c(getEdge(G_a, el_x, el_y)) ← c(getEdge(G_a, el_x, el_y)) +1
    foreach element el_n in next(el, t) do
        G_a ← generateAssociations(G_a, t, el_n)
    return G_a
```

**Algorithm 2:** Recursive generation of the association graph. The set, tree and graph functions are explained in Table 5.1.

nodes are the vertices of $V_a$ around which the clusters will be built. Principal node determination starts by a voting mechanism. Any vertex in $V_a$ submits a vote for the vertex connected by the edge of heaviest weight. Next, vertices that have a vote-degree ratio above the *popularityThreshold* parameter, become the principal nodes. Any vertex connected in $G_a$ to a principal node is assigned to the cluster of the principal node connected to the vertex by the edge of heaviest weight. Finally, the other vertices are iteratively added to the clusters corresponding to their edge of heaviest weight. The algorithm generating the destination cluster partition $P_D$ is the same but starts from a set of forward shortest path trees. Note that both types of shortest path tree can be generated by the Dijkstra algorithm. In case of a backward shortest path tree, it is required to invert any of the graph's edge directions before calculation, and to interpret the results accordingly.

For element 2, a minimal set of transit nodes for any cluster is determined. In case of origin/destination clusters, the paths in the sample backward/forward shortest path trees starting/ending in any vertex of the cluster are considered. The coverage of the transit nodes of a cluster is the proportion of these paths that passes at least one of the transit nodes. The minimal set of transit nodes for any cluster is determined, which has at least a specified coverage *tnCoverage* (e.g. 95%). This minimal set is retrieved through a mixed integer programming approach. The precalculated distances in the elements 3, 4 and 5 are calculated using a one-to-many shortest path algorithm. While the above distances are used to approximate long network distance queries, the probability that short paths of lowest cost pass through the transit nodes is remarkably lower. For

element 6, a set of close cluster pairs is determined. Two clusters are close if they have at least one vertex in common. For queries from cluster $C_O$ to $C_D$, the network distance approximation will not be based on the precalculated distances, but on the constant distortion minimizing the absolute RMSE for queries in $C_O \times C_D$. These distortions (of the last element) are calculated for a sample query set $random(C_O \times C_D, k_{avdist})$.

### 5.3.4 Network distance approximation algorithm

During a query for the network distance $d_G(u, v)$, the origin cluster $C_O$ and destination cluster $C_D$ are retrieved where $u \in C_O$ and $v \in C_D$. When $(C_O, C_D)$ is not a close cluster pair, $d_R(u, v)$ equals the minimal value of $d_G(u, s) + d_G(s, t) + d_G(t, v)$ for any combination of transit node $s$ of $C_O$ and transit node $t$ of $C_D$. Otherwise, $d_R(u, v)$ is $\gamma_{C_O,C_D} \cdot d_S(u, v)$.

### 5.3.5 Complexity

$c$ denotes the average number of partitions in $P_O$ and $P_D$, $t$ denotes the average number of transit nodes per cluster, and $l$ denotes the average number of clusters in $P_D$ that forms a close cluster with a cluster in $P_O$. Suppose that any of the 7 elements of the advanced ADO is stored in a hash table of space complexity $O(k)$ and average time complexity $O(1)$. The space complexity built up by these elements is: $O(2 \cdot n + 2 \cdot c \cdot t + c \cdot (c - l) \cdot (t)^2 + n \cdot t + n \cdot t + c \cdot l + c \cdot l) = O(n + c^2)$, assuming that $l$ and $t$ are much smaller than $c$ and $n$. This means that the space complexity is linear with regard to the number of nodes and quadratic with regard to the number of clusters. The query time complexity is $O(1)$ in case the origin and destination are located in a close cluster pair. Otherwise, it is $O(t^2)$.

## 5.4 Experiment

The advanced ADO introduced in the Section 5.3 is evaluated on a time-weighted transportation network extracted from OpenStreetMap, which is a source of publicly available geographical data. Mapping individual *way* objects in an OpenStreetMap map extract to a directed weighted spatial graph is described in Appendix A. After this extraction, a minimal number of vertices is removed from the graph such that the graph becomes connected. This process starts by the manual selection of a vertex $r$ which is known to be in the largest connected subgraph. Next, both a forward and a backward shortest path tree rooted in $r$

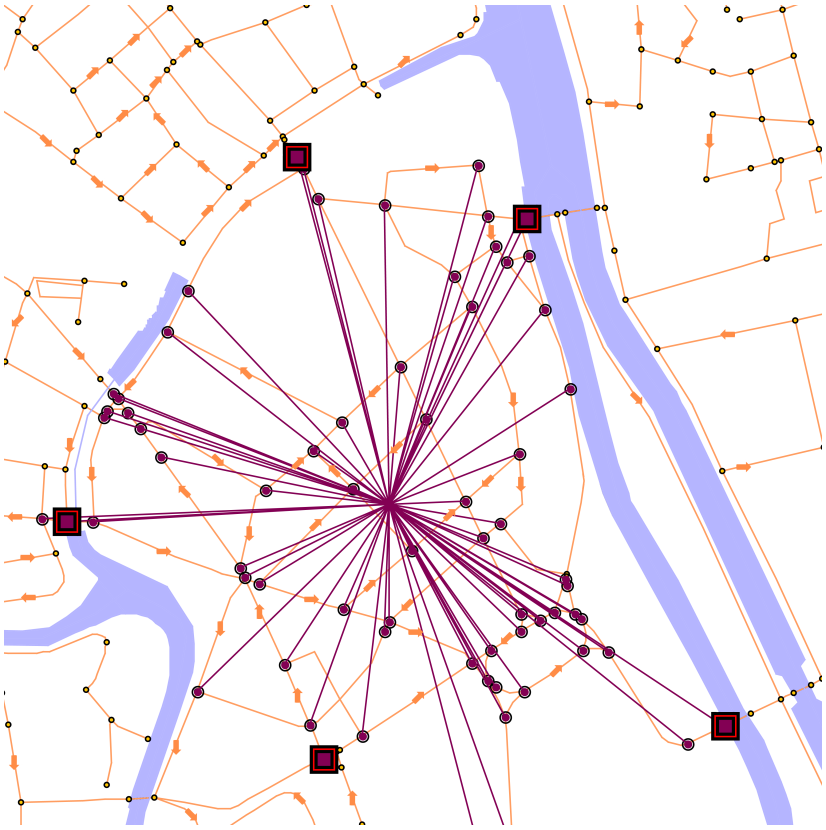


Figure 5.3: Instance of an origin cluster in the Ghent dataset. All vertices belonging to the cluster are connected by a star (trivial centre point). The other lines indicate the edges of the transportation network. Arrows indicate the major part of edges that are traversable in single direction. The 5 boxes represent the cluster's associated transit nodes, covering at least 90% of the sample paths leaving the cluster. Network and waterways derived from © OpenStreetMap contributors.

is constructed. The subgraph consisting of the nodes belonging to both trees is a connected graph.

The *Ghent dataset* is the OpenStreetMap map extract of latitude range [51.0258, 51.0834] and longitude range [3.6685, 3.7856], dated *May* 16, 2012. The derived connected graph contains 4296 vertices.

| | | | percentile | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *RMSEabs* | *avg* | *1%* | *5%* | *10%* | *90%* | *95%* | *99%* | *max* |
| Unit size ADO | 2.814 | 2.207 | 0.03 | 0.17 | 0.33 | 4.64 | 5.60 | 7.42 | 13.60 |
| Advanced ADO | 0.777 | 0.253 | 0.00 | 0.00 | 0.00 | 0.90 | 1.79 | 3.63 | 9.01 |

Table 5.2: Absolute error statistics (in minutes) of network distance approximations for the query sample set $S$ in the Ghent dataset.

The basic evaluation for this directed weighted spatial graph $G(V, E)$ assumes construction of two independent sets $S$ and $T$ of random ordered pairs (queries) in $V \times V$. Both sets contain 10000 elements. The positions of the individual vertices in the pairs of $S$ and $T$ have a uniform distribution over the rectangular area of interest. We assume that this is the expected query distribution for main-purpose applications of ADOs. Next, an ADO of unit size minimizing $RMSE^T_{abs}(G)$, and the advanced ADO introduced in Section 5.3 are constructed. The construction parameter settings for the latter oracle were $numberOfRoots = 50$, $popularityThreshold = 0.07$, $minDepth = 15$, $maxDepth = 10$, $tnCoverage = 0.90$, $k_{avdist} = 20$. These settings resulted in 87 origin and 79 destination clusters. One of the origin clusters, generated by this oracle, is shown together with its associated transit nodes in Figure 5.3. Table 5.2 shows the absolute error $RMSE^S_{abs}(G)$ and some other absolute error statistics for both oracles.
In order to analyse the oracle's approximation accuracy for different categories of "as the crow flies" distance between the origin and destination vertex of a query, we introduce the set $U$ of 10000 random queries $(u, v)$ in $V \times V$ where $d_S(u, v)$ has a uniform distribution. Both $RMSE^U_{abs}(G)$ and the average absolute error for $U$ were found to be lower sc. 0.730 and 0.211. Figure 5.4 shows the unit size oracle's absolute approximation error as a function of the query's spatial distribution for any query in the sample set of universal spatial distance distribution $U$. This data is averaged for discrete spatial distance ranges in the histogram of Figure 5.6. The same data on the advanced oracle's approximation accuracy is represented in Figure 5.5 and 5.7. The first histogram shows that the unit size oracle's absolute error average is around 2 minutes for medium-distant queries, but increases for longer distances up to 8 minutes. The other oracle's absolute error average is close to 0 minutes for long-distant queries, shows a peak of about 1.2 minutes around queries of distance 500m. This shows that the latter oracle is able to drastically reduce the absolute error and that it is the most susceptible to absolute errors when the origin and destination vertices are located in a close cluster pair. While the approximation mechanism for non-close cluster pairs yields a reasonable chance to have an exact approximation, the mechanism for close cluster pairs is based on spatial

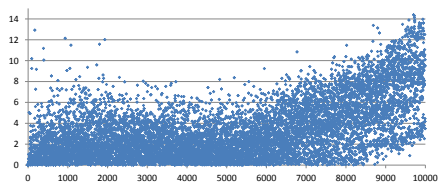



Figure 5.4: Unit size oracle's absolute error as a function of spatial distance.

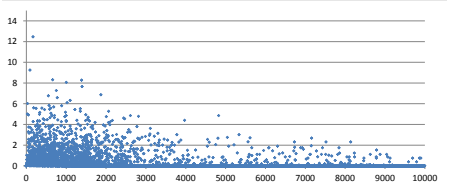



Figure 5.5: Advanced oracle's absolute error as a function of spatial distance.

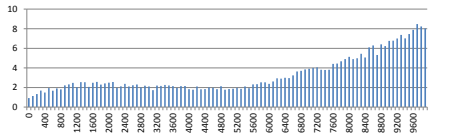



Figure 5.6: Unit size oracle's average absolute error histogram for spatial distance ranges.

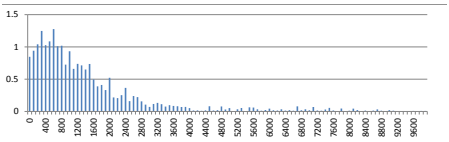



Figure 5.7: Advanced oracle's average absolute error histogram for spatial distance ranges.

distance multiplication by a constant. The latter mechanism can be seen as an improved version of the unit size oracle. For queries of length 400-500m, its approximations error is the halve of the one of the unit size oracle.

## 5.5 Conclusion

The present chapter introduces a framework for the evaluation of ADOs based on the root-mean-square error (RMSE) of the absolute or relative error of a sample set of network distance approximations in a graph $G(V, E)$. This framework applies both to general and spatial networks. An RMSE minimizing unit size oracle for general networks generates a constant approximation. The version for spatial networks approximates the network distance by the product of the spatial distance and a constant distortion.

An advanced ADO was introduced. It organizes the vertices $V$ into a partition of origin clusters and one of destination clusters. Each cluster is assigned a set of transit nodes. A close cluster pair is a couple of an origin and a destination cluster sharing at least one vertex. Network distance approximation between vertices of a close cluster pair is based on the constant distortion minimizing the absolute RMSE for a sample set of queries from the first to the second cluster of the pair. Approximation between remote vertices is based on a set of

precalculated distances between the vertices and the transit nodes and between transit nodes of different clusters. The oracle's average space complexity is $O(n+c^2)$ and its average query time complexity is $O(t^2)$, where $c$ is the average number of clusters in a partition and $t$ the average number of transit nodes of a cluster. Its prediction accuracy was evaluated in terms of the RMSE of the absolute error of the reported distances. The comparison with a unit size oracle for spatial networks was made for the Ghent dataset for a query sample set of size 10000. The positions associated with the individual vertices in this query sample set are uniformly distributed over the rectangular dataset area. The advanced ADO realizes a reduction of 3.5 times the RMSE and of 8.5 times the average absolute error, in comparison with the unit size oracle.

This work on distance oracles concludes the theme of shortest path (cost) approximation in resource-constrained environments. In Section 1.2 the concept of *attractiveness* has been denoted, supporting the retrieval of the *nicest* or most suitable path between two nodes in a network. Section 4.1 details the realization of this type of navigation in graphs where the edges have both a length and an attractiveness score. The next theme focusses on novel applications and services in this type of graphs. A tour suggestion module for outdoor activities is an example of such a service and is introduced in the next chapter. It has attractiveness-based point-to-point navigation as a component.

# Chapter 6

# Tour suggestion for outdoor activities

The present chapter introduces the outdoor activity tour suggestion problem (OATSP). This problem involves finding a closed path of maximal attractiveness in a transportation network graph, given a target path length and tolerance. Total path attractiveness is evaluated as the sum of the average arc attractiveness and the sum of the vertex prizes in the path. This problem definition takes its rise in the design of an interactive web application, which suggests closed paths for several outdoor activity routing modi, such as mountain biking. Both path length and starting point are specified by the user. The inclusion of POIs of some given types enrich the suggested outdoor activity experience.

A fast method for the generation of heuristic solutions to the OATSP is presented. It is based on spatial filtering, the evaluation of triangles in a simplified search space and shortest path calculation. It generates valuable suggestions in the context of a web application. It is a promising method to generate candidate paths used by any local search algorithm, which further optimizes the solution.

The research has been carried out as part of the industrial PhD project "Structural heuristics for personalized routes" funded by the IWT (090726) and the company RouteYou. The present chapter is a slightly adapted version of *Maervoet, J., Brackman, P., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G. (2013). Tour Suggestion for Outdoor Activities. In Liang, S. (Ed.), Wang, X. (Ed.), Claramunt, C. (Ed.), Lecture Notes in Computer Science: Vol. 7820. W2GIS 2013. Banff, AB, Canada, 4-5 April 2013 (pp. 54-63). Heidelberg, Germany: Springer.*

## 6.1 Introduction

The company RouteYou offers recreational navigation for several outdoor activity modes such as hiking and mountain biking. This involves maintenance of a set of transportation network graphs, in which each arc $r$ has a length $l_r$ and attractiveness $0 < a_r \leq 1$. The latter parameter models suitability to the applicable outdoor activity mode, in terms of the arc's scenic context, physical condition and relation to traffic. Point-to-point wayfinding is the key functionality of the company's web application. It implies finding the path with the lowest cost between two nodes of the graph, in which the arc costs equal $l_r/a_r$.

The motive behind the present chapter is the design of a tour suggestion module, which plans attractive round trips for any of the company's outdoor activity modes. This module requires that the user chooses an outdoor activity mode and specifies a target path length and a starting point. Within some seconds, it returns a closed path, consisting of arcs of optimal attractiveness and satisfies the length and starting point constraints. The desirable path has some constraints because users tend not to accept paths with a considerable number of self-intersections or with recurring subpaths. Paths in clockwise direction are usually prefered in countries with right-hand traffic because they ease turn traffic manoeuvres. Moreover, the user is able to select one or several POI types of preference, such that the output path results from a trade-off of arc attractiveness and the number of contained POIs of the preferred types. Examples of interesting mode - POI type combinations are hiking with mountains and motorcycling with scenic viewpoints. Another scenario is that external organizations set the mode and POI types of preference in a customized planner. This sort of planner aims at promoting a peculiar type of tourism in a specific region e.g. cycling along the Châteaux of the Loire Valley in France. In some cases POIs have a degree of membership to multiple categories.
An extension to this module is the generation of multiple suggestions. This means that the user can browse through a set of $m$ path suggestions, for the same set of preferences and constraints. It involves finding the set of $m$ most attractive tours that are spatially different.

The company determines the attractiveness in graphs by means of a linear combination of a set of parameters referring to an arc's scenic, physical and traffic-related context. The parameters have been extracted from a third-party geographical dataset. When a new outdoor activity mode is launched, the scalars of the linear combination are fine-tuned during a manual point-to-point navigation sensitivity analysis in the company's labs. The POIs originate from end user contributions on the web platform of the company. The types of these POIs have also been provided by the users. The contributions in the present

chapter do not depend on this data composition method and even apply to a more general setting where any form of edge-based and vertex-based suitability can be mapped to edge attractiveness and vertex prizes.

The next section gives a literature overview of models applied in the domain of leisure and tourism. Section 6.3 introduces the OATSP, which formalizes the tour suggestion problem described above. The approach presented in Section 6.4 enables generating a set of heuristic solutions to the OATSP in a low computational time. The following section discusses two sets of individual tours obtained by this approach. Section 6.6 is the conclusion of this work.

## 6.2 Tour suggestion models for leisure and tourism

The tour suggestion problem for leisure and tourism (TSPLT) involves generating a path through a transportation network visiting some arcs and/or points of interest (POIs). The path should optimally match the end user's preferences or some general recreational preference, given a set of constraints (adapted from the itinerary planning problem formulated by Shcherbina and Shembeleva [98]). It has applications in several subdomains of leisure and tourism. *Recreational point-to-point navigation* aims at generating a path from A to B, which is tailored to a specific recreational navigation mode, such as nordic walking. *Individual city trip planning* involves providing a tour schedule along a selection of POIs, satisfying the personal preferences of an individual intending to visit a city for a certain amount of time. This kind of services are often realized as a web-based application or a mobile client-server application (e.g. [18, 101]), generating on-the-fly suggestions for the end user. This is usually not required for applications in collective tourism planning (bus tour planning, cruise itinerary planning). The path generated in the TSPLT may be open or closed. In individual city tour planning for instance, the path is - in most cases - closed. The trip typically both starts at and ends in the tourist's hotel, a parking lot or a train station.

A first type of systems model the problem as a shortest path (SP) problem. This model generates exclusively open paths and focuses on the suitability of the arcs for a certain purpose. In the most common recreational SP approach, the inverse suitability is encoded in single arc weights of a directed weighted graph, corresponding to the transportation network. Path generation involves that common SP algorithms are used to find the path with the lowest cost between two nodes of this graph. Traditional SP algorithms are Dijkstra [30] and its variants [117] and A* [48]. The use of single scenic/attractive weights has often been suggested in this context (e.g. [94, 75]). The company presented in the introduction adapted this concept by introducing weight attractiveness

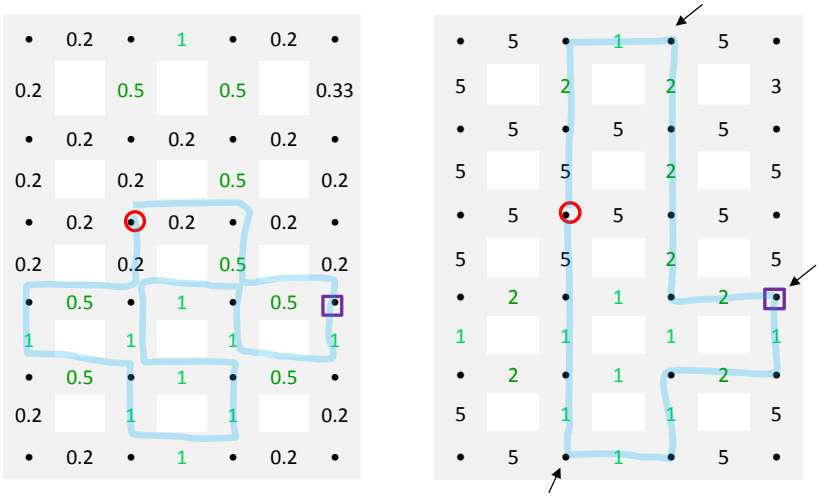
for a series of recreational routing modi. Attractiveness also deals with physical conditions and traffic aspects of the roads. Rogers and Langley [91] model the attractiveness of weights by a linear combination of criteria that reflect the end user's preferences. Niaraki and Kim [83] developed an ontology-based technique that generates network weights for personalized routing planning. Tarapata [108] states that single-objective functions are not sufficiently adequate to model real SP problems. He presents a classification of multi-objective shortest path (MOSP) problems, which are used in other real application domains, such as routing with quality-of-service in computer networks. The author identifies six general solution methods to MOSP problems, including mathematical optimization and objective function hierarchization. Hochmair and Navratil [49] argue that the computation of attractive routes is generally beyond the ability of SP algorithms, since an SP algorithm is not able to find a route that maximizes a benefit criterion. However, they demonstrate the practical value of single criterion SP computation for finding this type of routes.

A second type of systems use a problem model that trades off POI selection with time or distance. It originates from the field of Operations Research. It involves selecting a sequence of POIs from an eventually larger input set of POIs. This sequence should meet certain preferences and/or must satisfy a set of constraints. The POI sequence selection is often preceded by a POI filtering mechanism, improving the sequence selection performance. The model takes into account the travel time or distance between candidate POIs, aided by a precalculated travel time/distance matrix or a heuristic estimation function. The resulting path is obtained by concatenating precalculated (shortest) paths between the selected POIs, or by recalculating the complete path using via-points. The latter approach is useful in order to avoid undesirable U-turns and forbidden traffic manoeuvres passing through a selected POI. Godart [40] presented a version of the traveling salesperson problem (TSP) that integrates activity selection and lodging availability for trip planning problems. Deitch and Ladany [28] introduced the bus touring problem (BTP). It requires an undirected graph in which the vertices represent visiting sites and the edges represent connecting scenic routes. Both edges and vertices have associated attractivity values and require traveling/visiting times. The goal is to find a (closed path) bus tour of maximal total attractivity, below a given maximal tour time. The attractivity of recurrently visited vertices and edges is only counted once. The authors show that the BTP can be transformed to the Orienteering Problem (OP). Both Suna and Lee [106] and Maruyama et al. [78] have built tourist trip recommender systems based on variants of the prize collecting TSP (PCTSP). The original model minimizes the total travel cost minus the sum of the benefit criterion values of the POIs along the selected path. Suna and Lee integrate a personal interest factor in the travel cost weights. The two following query models in the field of spatial databases focus on trip

planning with *typed* locations. Li et al. [68] introduce the trip planning query, which involves a request for the shortest route from and to a given point that passes through at least one point of any of the specified set of location types. The optimal sequenced route query [97] looks for the shortest path that visits locations according to a specified sequence of POI types. An example of a POI type sequence for leisure is: (1) hair dresser (2) restaurant (3) cinema. Vansteenwegen and Van Oudheusden [114] introduced the tourist trip design problem (TTDP), which is modelled as an OP with time windows. This model starts from a fully interconnected distance/time-weighted graph in which the vertices represent POIs with a personalized [103] score. It involves finding the sequence of POIs that maximizes the total score of the selected POIs, while the total path weight must not exceed a given value. Each POI can only be visited once. Moreover, certain POIs are only available within certain time windows (cf. opening hours). Very good approximate solutions are found with iterated local search by Vansteenwegen et al. [112, 113]. They have shown the practicability of the TTDP in city trip planning.
All problems in this second class are $\mathcal{NP}$-hard. Both Shcherbina and Shembeleva [98] and Souffriau and Vansteenwegen [102] provide a more detailed overview of the models and functionalities of this type of systems.

## 6.3 The outdoor activity tour suggestion problem

The BTP model suits the requirements of company's tour suggestion module best. The objective function takes into account both edge and vertex attractiveness, whereas tour time (or length) resides under the constraints. However, the arc attractiveness $a_r$ has been optimized for point-to-point wayfinding in the company's labs. In this setting, $a_r$ is composed as a linear combination of a set of parameters referring to various aspects of attractiveness. During a SP analysis, the scalars of this linear combination are fine-tuned such that paths of minimal $\sum \frac{l_r}{a_r}$ hold the optimal balance between length and attractiveness. This process is very similar to the determination of the optimal buffer radius settling a trade-off between scenic sections and detour, by Hochmair and Navratil [49]. It implies that (sub)paths of optimal attractiveness below a specified length, as produced by the BTP, often lack this trade-off. This is illustrated for a sample Manhattan graph in Figure 6.1 at the left side. Therefore we introduce a problem model for which a feasible tour suggestion consists of a small number of concatenated paths of minimal $l_r/a_r$, aiming at optimal edge and vertex attractiveness. The target path length required by the tour suggestion module gives rise to a distance window constraint. A sample optimal path for this model is shown at the right side of Figure 6.1.




(a) (b)

Figure 6.1: Optimal paths for (a) the BTP with a distance window constraint and (b) the OATSP with $\varphi = 1$, in a sample Manhattan graph. The total path length must be in the range of $[14, 18]$. In this graph any arc has a reverse arc of equal $a_r$ and any $l_r = 1$. At the left, the values indicate the arcs' $a_r$; at the right they indicate $l_r/a_r$. The circle indicates the starting vertex $v_s$. Any vertex prize $p_v = 0$ except for the prize of the squared vertex where $p_v = 1$. The arrows point to the vertices between which subpaths of minimal $l_r/a_r$ are constructed.

**Definitions.** $X_i$ denotes the element at the $i$-th position in any series $X$. In a directed graph $G = (V, A)$

- any arc $r \in A$ is an ordered pair $(v_1, v_2) \in V^2$,
- any arc $r \in A$ has an associated attractiveness $0 < a_r \leq 1$ and length $l_r$,
- reverse arcs have equal attractiveness and length,
- each vertex $v$ has an associated prize $0 \leq p_v \leq 1$, and,
- each vertex has a latitudinal coordinate $y(v)$ and a longitudinal coordinate $x(v)$ in an authalic map projection.

A closed path $C$ is a series of arcs, which are circularly subsequent in $G$. $C^V$ is the corresponding series of vertices visited by $C$. The sets $A_C \subseteq A$ and $V_C \subseteq V$ consist of the arcs and vertices visited by the closed path $C$. The set $A'_C$ contains any arc $C_i = (v_1, v_2)$ of $C$ that is not preceded by its reverse arc $C_j = (v_2, v_1)$ with $j < i$. The shortest path $sp(v_a, v_b)$ refers to the path $S$ from $v_a$ to $v_b$ for which $\sum_{r \in A_S} \frac{l_r}{a_r}$ is minimal.

**Problem specification.** Given a prefered path length $l_p$, length tolerance $t$, *small* maximal sequence length $k$ and starting vertex $v_s$, a solution to the OATSP is a series of intermediate vertices $I$ with $|I| \leq k$, such that the closed path $C$ in $G(V, A)$, arising from the concatenation of $sp(v_s, I_1)$, $sp(I_1, I_2)$, ..., $sp(I_n, v_s)$ satisfies

1. Each element of $A$ appears at most once in $C$.
2. $(1-t) \cdot l_p \leq \sum_{r \in A_C} l_r \leq (1+t) \cdot l_p$
3. $$\frac{2\pi}{\left(\sum_{r \in A_C} l_r\right)^2} \cdot \sum_{\substack{i=1\ldots|C^V| \\ j=(i \bmod |C^V|)+1}} (x(C_j^V) - x(C_i^V)) \cdot (y(C_j^V) + y(C_i^V)) > \sigma$$
4. $$\varphi \cdot \frac{\sum_{r \in A'_C} (a_r \cdot l_r)}{\sum_{r \in A_C} l_r} + \sum_{v \in V_C} p_v \text{ is maximal}$$

This model suits the requirements of the tour suggestion module. It is able to suggest closed paths of a target length and optimal arc and POI attractiveness, by setting the vertex prizes to the degree of membership to the specified categories.

The first constraint prevents recurring arcs in a tour. Since recurrent visits to reverse arcs and nodes are penalized by the objective function (item 4), solutions tend to avoid U-turns and recurring reverse subpaths. The model does allow multiple vertex and reverse arc visits since the starting vertex or any of the POIs may be located on a subgraph with low connectivity.

The second constraint specifies a length window around target $l_p$ of width $2 \cdot t$. The inequality under item 3 specifies a lower bound $\sigma$ for the clockwise area measure. This measure evaluates the linestring resulting from $C^V$, which may be self-intersecting or self-tangent. Substracting the portions of enclosed area surrounded in counter-clockwise direction from those surrounded in clockwise direction, it indicates the *clockwiseness* of a tour. In countries with left-hand traffic, the $>$ inequality should be changed to $<$, since counter-clockwise tours result in negative clockwise area values. Moreover, this measure evaluates the

linestring's normalized area-perimeter ratio, which approaches 1 in case of a clockwise circle.
The objective function in item 4 takes into account the average attractiveness along the tour and the prizes of the vertices in $C^V$. The parameter $\varphi$ determines the relative importance of arc to node attractiveness.

A multiple suggestion extension to the model requires a distance function between the linestrings of two closed paths.

## 6.4 Approach

A multiple tour suggestion module has been designed for the web application of the company. It enables generating a set of $m$ heuristic solutions to the OATSP within a low computational time, in an environment without precalculated paths nor distances available. The main algorithm starts by determining a *feasibility window (FW)*. This square area discerns the arcs and nodes reachable in a round trip through $v_s$ given the path length window constraint. The algorithm assumes that, within the FW,

- only a few (ranging from none to the order of tens) vertices $v$ have $p_v > 0$,
- there is a diversity of attractiveness $a_r$ amongst the arcs $r$, and,
- vertices with $p_v > 0$ have a high probability to be located along trajectories of higher attractiveness than vertices in their local neighbourhoods.

These conditions often hold for the tour suggestion module presented, where the user selects few POI categories and relatively low values for $l_p$. Moreover, it assumes low values of $\varphi$.

Figure 6.2 shows the state diagram of the main algorithm. The underlined numbers in the diagram description below have been experimentally determined. In state (1), the FW is generated in the geographic coordinate plane, such that $v_s$ is in the center and the projected width and height measured through $v_s$, equals $l_p/\sqrt{2}$. If it contains more than 5 feasible POIs ($p_v > 0$), the algorithm continues with triangle search. Otherwise, the window is enriched by highly attractive vertices (HAVs), in state (2). HAVs are auxiliary POIs representing the most attractive arcs in the FW. This phase looks for the maximal value $a_{min}$ for which there exist at least 20 arcs $r$ with $a_r \geq a_{min}$. Any vertex that is connected by an arc $r$ with $a_r \geq a_{min}$, is an HAV. The maximal attractiveness of the connecting arcs of an HAV $v$ is denoted $a_v$. Next, all HAVs are promoted as POIs, with $p_v$ set to $\varphi a_v / \#HAVs$. If the total number of POIs does not

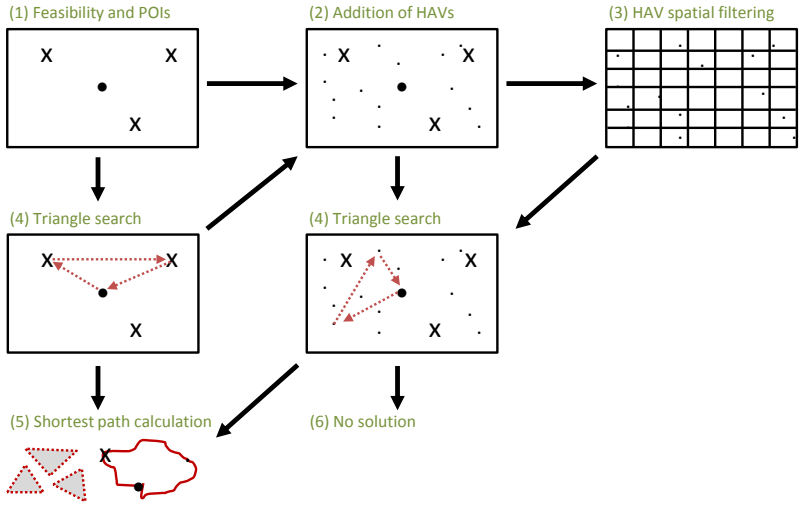



Figure 6.2: State diagram of the fast heuristic algorithm for the OATSP.

exceed 60, the algorithm continues with triangle search. Otherwise, the HAVs are subjected to spatial filtering (state (3)). This involves that all points are categorized into one of the boxes of the $g_1 \times g_1$ grid constructed over the window. Only one auxiliary POI per box is kept, absorbing the sum of $p_v$-values of the other points in the box.

Triangle search in state (4) involves the brute-force evaluation of any directed triangle made up by $v_s$ and any other two POIs within the window. This evaluation consists of two steps. First, a triangle undergoes a fast feasibility check. A triangle is feasible if its direction is clockwise (in case of right-hand traffic) and if its perimeter is between $(1 \pm t) \cdot l_p/1.6$. Next, an evaluation function computes a score for a feasible triangle in this simplified window representation. This function returns a weighted sum of the prizes $p_v$ for any of the POIs involved. Prizes of POIs located at one of the triangle vertices are given double weights. POIs located on one of the triangle edges generate $p_v$, which decreases as the elliptical distance to the closest triangle edge increases. If all angles are greater than $40°$, a global bonus ($*1.5$) is granted.

When no feasible triangle is found in a window containing auxiliary POIs, the algorithm does not return any solution (state (6)). Otherwise, the triangle of highest score is passed to the SP calculation state. Given a triangle $abc$, this fifth state entails finding the concatenation of paths of lowest cost between $(a, b)$, $(b, c)$ and $(c, a)$ in a graph with arc weights equal to $l_r/a_r$. In order to avoid U-turns and recurring subpaths, the weights of both the forward and

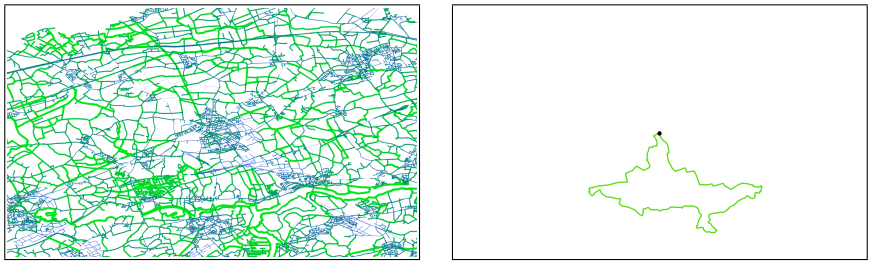
available reverse arcs of the resulting subpath are drastically increased, after each subrouting. If the concatenated path infringes the recurring arc, the length window or the clockwise area constraint, the paths between $(a, b)$, $(b, c)$ and $(c, a)$ are calculated in a changed order and again concatenated. If it still does not meet the constraints, it is rejected after which the second best triangle is processed, and so on.

The multiple suggestion extension is realized by the integration of a simple *competitive learning* algorithm in the triangle search. It relies on a triangle distance function which is defined as follows. The two variable vertices of a triangle are categorized into one of the boxes of the $g_2 \times g_2$ grid constructed over the window. The *distance* between two (clockwise) triangles is the sum of the Manhattan distance between the first vertices and the Manhattan distance between the second vertices in the grid. Two triangles are called *resemblant* if one of the Manhattan distances is lower than 2. During the brute-force triangle evaluation, the algorithm manages a store of maximally $m$ prototype solutions. Suppose the current solution does not resemble any prototype solution. If there are less than $m$ prototype solutions, the current solution enters the store as a new prototype. If the store is full and the current solution is better than the worst prototype, it replaces the prototype. Suppose the current solution resembles one or more prototype solutions. If it is better than the closest prototype, it replaces the prototype. In the end, the $m$ prototype solutions are passed to the SP calculation state.

## 6.5 Results

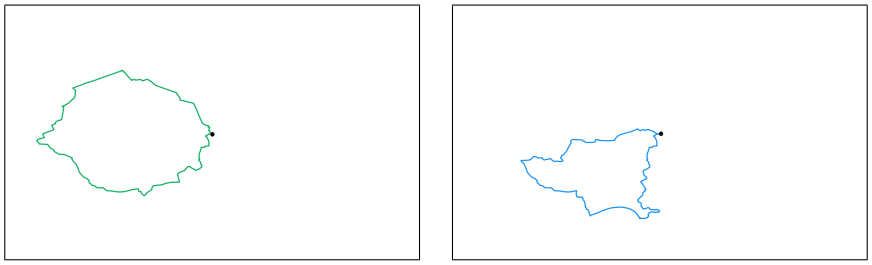
The approach introduced in the previous section has been tested for the outdoor activity mode 'attractive cycling'. The transportation network graph for this mode contains both paved and unpaved roads. Further settings are $\sigma = 0.3$, $\varphi = 1$, $g_1 = g_2 = 10$, $m = 10$ and $t = 0.30$.

A first experiment, called experiment SN-1, assigns the centre of a medium-sized city in Belgium as starting vertex $v_s$ and sets $l_p = 30km$. No POI category of interest is specified, so initially any vertex prize $p_v = 0$. Figure 6.3 shows the arc attractiveness map within the FW and the 3 out of 10 tours of highest scores in the solution store. Most of the arcs within the city are substantially less attractive than the arcs in the neighbourhood. The arcs in the south of the window, either along the rivers or within a woody region, have the highest attractiveness. Each of the top-3 tours visits this region. Any of the 10 tours in the store were assessed by amateur cyclists as attractive tours. They valued the solution diversity in the store highly.



(a) Arc attractiveness

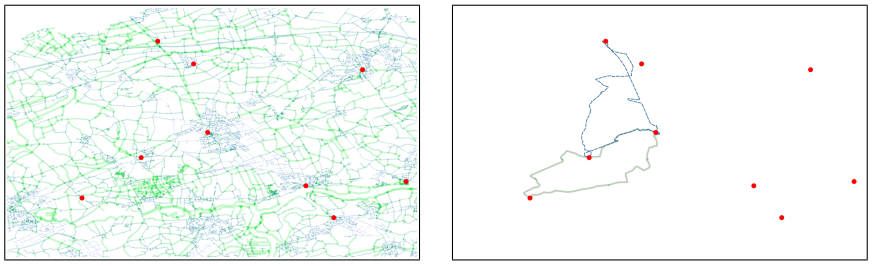
(b) Best solution



(c) Second-best solution

(d) Third-best solution

Figure 6.3: Arc attractiveness map and the three best tours in the solution store for experiment SN-1. Each subfigure is shown in the FW, centered around $v_s$. Highly attractive arcs are depicted by thick green lines, and less attractive arcs by thin blue lines.



(a) Arc attractiveness

(b) Best and second-best solution

Figure 6.4: Arc and node attractiveness map and the two best tours in the solution store for experiment SN-2. Each subfigure is shown in the FW, centered around $v_s$. The red dots indicate the POIs of the type 'interesting church'. The best solution is indicated by a solid line.

Experiment SN-2 adds the POI category 'interesting church' to the specifications of experiment SN-1. Each closest vertex to a POI of this category gets $p_v = 1$. Figure 6.4 shows these POIs, and the 2 out of 5 tours of highest scores in the solution store. One of the POIs is located very close to and to the west of $v_s$. This gives tours in the west containing this POI, priority over the other tours.

| | SN-1 | SN-2 |
|---|---|---|
| # POIs | 0 | 9 |
| # HAVs | 764 | 0 |
| # HAVs after reduction | 84 | 0 |
| # feasible triangles | 1256 | 13 |
| # comput.time of triangle search | 752 ms | 2 ms |
| # solutions in store | 10 | 5 |

Table 6.1: Experiment SN-1 and SN-2 statistics.

Table 6.1 indicates that in experiment SN-1, 764 HAVs were added to the FW and reduced to a set of 84. It took 752 ms to evaluate $84 \cdot 83$ triangles[1]. Experiment SN-2 only went through states (1), (4) and (5), so no HAVs were added. It results in a very constrained triangle set. Only 5 out of 13 feasible triangles were considered sufficiently diverse by the competitive learning algorithm. The

| | Triangle eval. fct. | Tour length | OATSP obj. fct. |
|---|---|---|---|
| SN-1 # 1 | 0.4477 | 37.6 km | 0.8126 |
| SN-1 # 2 | 0.3937 | 34.9 km | 0.8316 |
| SN-1 # 3 | 0.2998 | 31.8 km | 0.7800 |
| SN-1 # 4 | 0.2619 | 28.6 km | 0.7297 |
| SN-1 # 5 | 0.2280 | 30.4 km | 0.6826 |
| SN-2 # 1 | 4.9709 | 25.1 km | 3.7618 |
| SN-2 # 2 | 4.9073 | 30.1 km | 3.7265 |
| SN-2 # 3 | 4.3079 | 34.1 km | 2.7078 |
| SN-2 # 4 | 4.0000 | 32.3 km | 2.7393 |
| SN-2 # 5 | 4.0000 | 25.6 km | 2.7660 |

Table 6.2: Top-5 result characteristics for experiments SN-1 and SN-2.

triangle evaluation scores, tour lengths and OATSP objective function scores of the 5 best SN-1 and SN-2 results in the solution store are given in Table 6.2. Specifically in SN-1, the OATSP objective function equals the attractiveness average of arcs visited by the tour. The triangle evaluation score correlates with the objective function. Only the solution of rank 1 has been overestimated. The

[1]The experiments have been executed using PHP (Command Line Interface) 5.3.6 in Ubuntu 11.10, with an Intel Core i7-920 Processor.

average arc attractiveness along the $(a, b)$ subpath of this solution is remarkably lower than the average along the other subpaths. In the case of SN-2, the OATSP objective function takes into account both arc and node attractiveness. Only POIs were assigned to the triangle vertices, and therefore each tour scores at least two. Since no HAVs were used to calculate the triangle evaluation score, this function is only effective in predicting the number of POIs contained by a tour.

## 6.6 Conclusion

The OATSP, introduced in the present chapter, involves finding attractive closed paths in a transportation network graph, tailored for a specific outdoor activity mode. The objective function is based on both the arc attractiveness along the path and the prizes of the visited vertices. A window constraint restricts the total path length.

An experiment showed that the presented algorithm is able to generate a set of heuristic solutions to the OATSP, satisfying the constraints. The core of this algorithm is a brute-force triangle evaluation in the FW, containing the POIs with $p_v > 0$ and a set of auxiliary POIS, constrained to the number of $g_1^2$. Although a quadratic number of triangles is evaluated, the evaluation runs in low computational time. The triangle that receives the best evaluation is passed to an SP calculation module, prioritizing paths along attractive arcs. The triangle evaluation function has been found an efficient heuristic for the OATSP objective value of the resulting path. A simple competitive learning algorithm enables generating $m$ tours of high objective value, which are spatially different. This algorithm of low computational impact showed to be effective.

The algorithm has been integrated in a web application. Further work comprises an evaluation of the objective function scores and computational time of the results generated by this algorithm, an exact method and a local search method. Moreover there are opportunities to improve the introduced algorithm. If precalculated subpaths are introduced in the triangle evaluation, the OATSP objective function can be evaluated for each triangle during triangle search. Through this, the real OATSP objective scores instead of heuristic triangle scores will determine which triangle is selected. The replacement of the brute-force triangle enumeration technique by a heuristic selection of triangles will decrease the computation time drastically. Obviously, the triangle approach will fail to produce a heuristic solution to the OATSP when the feasibility window exhibits many POIs or many regional variance in attractiveness. In these cases,

the evaluation of polygons could generate better results, at the expense of the search space size.

This work in close collaboration with industry resides under the third theme of the present dissertation, about the discovery of structures related to attractiveness in spatial graphs. Tour suggestion for outdoor activities entailed a novel type of service in spatial networks where the edges have both a length and attractiveness score.

# Chapter 7

# Conclusion

## 7.1 Contribution

In the present dissertation various applications of high importance to GIS practice have been introduced and validated. The quality of large amounts of dynamic geographical data originating from multiple sources can be maintained effectively by tools and methods based on relational datamining. Furthermore, three service components for spatial networks have been presented and validated, dealing with efficiency in environments with a limited amount of resources. Both scientific and economic contributions are clarified below for each of the three themes contributing to the present thesis.

**Geographical data quality.** DILP was identified as a suitable class of techniques to identify anomalies in geographical data. An experiment showed the practical value of considering *almost complete* rules, i.e. regularities that hold over all the geographical data apart from a small number of exceptions, and looking up these exceptions afterwards. Exceptions indicate either errors or correct but exceptional situations in the data. The experiment showed the restrictions of both the input data size and the number of geographical relations and feature and attribute types involved in a single experiment. It revealed the need for a more generic infrastructure, supporting an evolving data model and facilitating the automated geographical data extraction and preparation and pattern modelling and evaluation. This dissertation further addresses this concern by the introduction of a generic tool to mine for relational regularities and corresponding outliers in geographical data. It is adaptive to data model changes since it integrates a geographical metamodel in the knowledge discovery

process. Its value is supported by two experiments and a sanity check. Data engineers of Tele Atlas identified previously unknown and valuable patterns in the experiment results, and the sanity check showed that the tool is able to identify three out of four rules that are known by the company's quality maintenance system. The missing rule was not found because there were too few instances in the sample data to which it applies.

The feasibility study of applying descriptive ILP to large geographic databases, now an integral part of the present dissertation, was at the time the first instance where relational datamining is applied for outlier detection. The approach is more broadly applicable for outlier detection and quality maintenance in data-rich intelligent systems.
For the field of data quality analysis in GIS, traditionally relying on techniques prospecting for statistical deviation in geographical data, this work entailed the first application of relational data mining techniques. The resulting tool was adopted by the quality maintenance team of Tele Atlas in December 2008 for further integration in the quality infrastructure. This infrastructure comprises a geographical rule language, operational tools for the creation and management of quality rules, and, autonomous processes for the tracing and reporting of outliers. The tool has received a lot of positive feedback both from the data engineers and the engineering management of the company. The underlying concepts of the tool can be applied seamlessly to any GIS.

**Shortest path (cost) approximation.** A multi-tier architecture was depicted where database operations heavily burden the performance of the shortest path calculation process. Moreover, both storage costs and calculation time of preprocessing procedure are restricted. An analysis showed that the class of (often approximate) hierarchical shortest path algorithms operating on graphs divided into predefined levels suits these requirements best. Prior experiments showed that the *heuristic node promotion algorithm* exhibits the best trade-off between calculation time and cost of the approximate path. As this algorithm originally was defined for two hierarchic levels, it was redefined such that it is effective for any number of levels. Since the original algorithm requires a problematic irreversible graph transformation, two alternative geographical data processing steps and two algorithmic adaptations were proposed restoring the algorithm's effectiveness. An experimental evaluation showed the effectiveness of each of the proposed adaptations. The combination of the resulting algorithm and a rectangular pruning mechanism generates paths of which the cost approximates the path of lowest cost and achieves low average number of visited nodes.

This approach illustrated the applicability of geographical data processing in order to increase the efficiency of an algorithm operating on this data. The cell classification method presented in one of the additional geographical

data processing steps, can be more generally applied for functional road class prediction for transportation networks in GIS. The approach has been operational in the multi-tier environment of the company RouteYou for approximately 3.5 years.

An evaluation framework for approximate distance oracles was introduced, based on the relative or absolute RMSE of its network distance approximations in a graph. An advanced oracle for distance approximation minimizing the absolute RMSE in spatial graphs was presented. It is based on partitioning graph vertices into clusters of vertices sharing an LCA only a few edges away for the greater part of paths of lowest cost. The approximation accuracy of this oracle and a unit size oracle were evaluated for a set of 10000 query samples. The first oracle reduced 3.5 times the RMSE and 8.5 times the average absolute error of the latter oracle.

The framework introduced shifts the emphasis for ADO evaluation in general from guaranteed upper and lower bounds towards overall accuracy performance, and allows both absolute and relative errors. The RMSE of the presented oracles of unit size can be used as a *reference value* for ADO evaluation. Various design concepts behind the advanced ADO, such as shortest path tree sampling, association graph construction and determination of a minimal set of transit nodes, can be reused in related application domains e.g. shortest path calculation. In as far as it has been possible to backtrack publications on the subject, it was the first time that an ADO designed for spatial networks was validated on a time-weighted transportation network.
This work resulted in a compact component, which can be integrated in any application requiring shortest path cost approximation, such as solvers of combinatorial optimization problems.

**Discovery of structures related to attractiveness in spatial graphs.** The tour suggestion problem for outdoor activities in graphs was introduced. This problem model aims at optimizing the arc and vertex attractiveness of a closed path in a transportation network graph, satisfying a set of constraints: no recurring arcs are allowed in the closed path; the total path length is restricted by a window constraint, and, only (counter-)clockwise paths of a high area-perimeter ratio are allowed. An algorithm of low computational impact generating heuristic solutions to this problem was presented. An extension of this algorithm based on competitive learning collects multiple tours of high attractiveness, visiting different spatial regions. It was shown that this algorithm produces valuable solutions satisfying the constraints.

The application domain of this type of discovery models is not limited to tourism and leisure. They generally apply to problems where attractiveness corresponds to inverse cost or a degree of urgency or demand.

The presented algorithm is of added value to the company RouteYou. The tour suggestion algorithm was integrated in the company's operational multi-tier architecture, in order to plan closed paths which are attractive with regard to any of the outdoor activity modes available.

The results of the present thesis have been achieved by a multi-disciplinary approach combining the fields of relational data mining and heuristic algorithms with techniques from software engineering, geographical data processing, complex data modelling and computational experimentation.

## 7.2 Further directions

The work presented in this thesis offers many opportunities to further improve and validate (1) quality maintenance in geographical data and (2) components delivering services regarding spatial networks. Directions for further research and for extending the above contributions are clarified below by theme.

**Geographical data quality**

- The two-fold process of discovering regularities and corresponding violations has proven its efficiency in providing the data engineers with complete information. This finding clears the path for the further formalization of relational outlier detection and the design of corresponding algorithms for direct detection. The formalization and algorithm design for related models was treated by Angiulli et al. [4] and Angiulli and Fassetti [3].

- It was shown that rules expressed in (a subset of) first-order logic are useful to model regularities in GIS. Relational regularities of different structures e.g. decision trees are latently present in the geographical data, although to a lesser extent. The integration of the corresponding mining algorithms in the generic tool introduced in Chapter 3 is a potential complementary approach.

- The tool enables detection experiments for outliers relating to a limited number of types and relations in a limited amount of geographical data. The integration of these experiments in the global quality process leaves a number of open questions. Which combinations of geographical features, attributes and spatial relations are worthy of being considered in one experiment? Is there any heuristic to design fruitful combinations e.g. themed maps? The large scale application of experiments will produce many redundant equivalent rules: what are the appropriate measures for

detection and prevention? How should the rule repository deal with a set of inequivalent rules producing the same outliers? Should the company maintain a repository of rules or a repository of outliers? Can we apply the data mining process over multiple partitions of geographical data in a distributed and asynchronous manner?

- A similar tool for OpenStreetMap data would enable individuals to trace anomalies and hence improve the quality of this collaboratively produced data. Moreover, it would contribute to the formalization of the project's map feature specification, which is collaboratively edited.

**Shortest path (cost) approximation**

- In Chapter 4, the top-down, bottom-up and stitching algorithmic strategies were identified to integrate well in the multi-tier architecture. An adapted version of the top-down strategy was evaluated, but the integration, modification and evaluation of the two other approaches were not tackled.
- The advanced ADO's design presented in Chapter 5 is still open to improvement. Currently, the minimal set of transit nodes, for a specified coverage ratio, is determined by a mixed integer programming approach. Instead, an approximation of the minimal set for this ratio will streamline the time performance of the oracle's construction. The integration of a hierarchical approach, implying graph partitions of multiple coarsenesses, offers perspectives for enhancing the ADO's scalability.
- The introduced graph partitioning concepts can be applied to exact shortest path algorithms. The very effective transit node approach by Bast et al. [7], for instance, minimizes the number of transit nodes per cluster, but does not consider the graph partitioning as such. The introduction of an LCA-based partitioning method would enable further reduction of the global number of transit nodes.

**Discovery of structures related to attractiveness in spatial graphs**

- Further research implies the generation and user-oriented evaluation of solutions that are optimal to the OATSP. It would be interesting to compare the objective function scores of the optimal solution and the solutions generated by the algorithm presented in Chapter 6. Alternative (meta-heuristic) approaches generating good solutions to the problems introduced, such as local search and genetic algorithms are left to investigate.

- The discovery of connected subgraphs of high attractiveness is an interesting subject of equal contribution to the last theme. For the company RouteYou, this type of subnetwork serves (1) as a data structure enhancing route computation performance (as in the second theme), (2) as a visual map overlay aiding web platform users to plan highly attractive routes, and, (3) as a reference network of a hardcopy themed recreation guide. This subject of high industrial relevance is closely related to the latent natural hierarchy in graphs. An example concept revealing this hierarchy is the *reach* metric introduced by Gutman [44]. This metric encodes for each vertex the maximum length of the paths of lowest cost on which it lies, and enables the exactness of reach-based shortest path algorithms of low computation time. It is currently unclear whether this concept can be used to harvest a *discrete* hierarchy guaranteeing exactness for classical hierarchical shortest path algorithms.
- The application of closed path and connected subgraph discovery algorithms to other domains is left to investigate. The discovery of closed paths of high attractiveness can be applied to schedule tours for a mobile team in a transportation network e.g. police surveillance scheme, city cleaning tour. In this case, arc attractiveness represents a certain *urgency* of a task or a degree of demand along the arcs. The discovery of connected subgraphs of high attractiveness has also applications in the domain of urban planning. Here, attractiveness models the inverse (infrastructural) costs and the demand for promoting an arc to for instance a safe cycling network or a public transportation network.

As two important evolutions in digital map-making and offering online maps were introduced in Chapter 1, a third more recent evolution involves the widespread use of smartphones and the breakthrough of mobile internet and social networking. Since smartphone devices are equipped by a set of sensors such as accelerometers and positioning components, this trend paves the path for *crowdsourcing* of geo-referenced data, implying the collection and compilation of this type of data originating from online communities. Crowdsourcing can be based on various types of contributions, ranging from active data mediation and provision e.g. collaborative mapping, to the implicit retrieval of data resulting from the actions of a user actively performing other tasks. An example of the latter form is a navigation app on a smartphone collecting traffic information.

Crowdsourcing offers a gamut of opportunities both for quality maintenance and service enhancement regarding geographical data. Most of the current navigation applications can be extended by a component enabling individuals in actively reporting anomalies encountered in the geographical data. Another way of data quality improvement involves detection of inconsistencies between

traffic information and the current map data. Implicitly retrieved geo-referenced sensor data can be used to verify certain attributes of a geographic object e.g. a way's surface type. Inconsistencies between suggested paths and tours and the actual traversed trajectory - in combination with user feedback - are caused by inconsistencies in the data or by inadequacies in the path suggestion module. Collaborative mapping can contribute to the fine-tuning of a themed reference network produced by the connected subgraph discovery algorithm.

This dissertation supports further developments in various research areas such as relational datamining, shortest path algorithms and *operations research*. For the application domains of quality maintenance in GIS and routing in resource-constrained environments, the continuation of this work offers perspectives for further enhancement of the quality of both geographical data and the services built upon this data.

# Appendix A

# Extraction of a directed weighted spatial graph from OpenStreetMap data

The following rules describe a mapping of OpenStreetMap (OSM) data to a car transportation network, in the form of a directed weighted spatial graph $G(V, E)$.

An OSM *way* instance is **traversable** if

- it does not have a *visible*-tag of value *false*,
- it does have a *highway*-tag, and,
- the value of this tag is different from *bridleway*, *bus_guideway*, *construction*, *cycleway*, *footway*, *path*, *pedestrian*, *proposed*, *raceway*, *service* or *steps*.

If a *way* instance has a *oneway*-tag, this *way* is traversable

- in the **forward direction** if this tag's value equals *yes*, *true* or *1*,
- in the **backward direction** if this tag's value equals *-1*,
- in **both directions** if this tag's value equals *no*, *false* or *0*.

Otherwise, if this *way* instance has a *junction*-tag, this *way* is traversable in the **forward direction** if this tag's value equals *roundabout*.
Otherwise, if the *highway*-tag's value of this *way* equals *motorway_link* this *way* is traversable in the **forward direction**.
In any other case this way is traversable in **both directions**.

If a way instance has a *maxspeed*-tag, its **speed** in km/h is defined as:

- 130 if this tag's value equals *none*,
- 50 if this tag's value equals *signals*,
- this tag's value if it is a number,
- this tag's value multiplied by 1.609344 if it is a number followed by *mph*.

Otherwise, when the way instance has a *maxspeed*-tag and its value has the *countrycode:waytype* pattern, the country code and way type yield a **speed** according to the following mapping.
*(((BE, motorway), 120), ((BE, trunk), 90), ((BE, primary), 90), ((BE, secondary), 90), ((BE, tertiary), 90), ((BE, residential), 30), ((BE, living_street), 20),*
*((NL, motorway), 120), ((NL, trunk), 100), ((NL, primary), 80), ((NL, secondary), 80), ((NL, tertiary), 80), ((NL, living_street), 15),*
*((ES, motorway), 120), ((ES, trunk), 100), ((ES, primary), 90), ((ES, secondary), 90), ((ES, tertiary), 90), ((ES, residential), 30), ((ES, living_street), 20))*
Otherwise, the same mapping applies to the country code derived from the data extract's metadata and the value of the *highway*-tag.
When the mapping failed, its **speed** is 50.

We define the **reduced node reference list** of a traversable *way* instance as the list of node references appearing as the first or the last element in the *node reference list* of any traversable *way* instance. The **waylength** between the nodes $n_i$ and $n_j$ appearing in a full *node reference list* of a *way* instance equals the sum of the spherical distances in metres between any two consecutive nodes $n_k$ and $n_{k+1}$ in the full *node reference list*, where $i \leq k < j$.

Any OSM *node* instance appearing in the reduced node reference list of a traversable *way* instance maps to a vertex of $V$. The values of its *lon*-tag and *lat*-tag map to the spatial position in $\mathbb{R}^2$ of the vertex. $d_S(u, v)$ indicates the spherical distance in metres between the vertices $u$ and $v$.
Any two consecutive nodes of the reduced node reference list of a *way* traversable

in the forward or in both directions maps to a directed edge of $E$ starting in the first corresponding vertex and ending in the second corresponding vertex. Any two consecutive nodes of the reduced node reference list of a *way* traversable in the backward or in both directions maps to a directed edge of $E$ starting in the second corresponding vertex and ending in the first corresponding vertex. The weight of an edge $e$ from vertex $u$ to $v$ corresponds to $\frac{waylength(u,v)\cdot 0.06}{speed(e)} + 0.167$, where $speed(e)$ refers to the speed of the *way* instance from which $e$ is generated.

# Bibliography


[1] Adler, J. L. A best neighbor heuristic search for finding minimum paths in transportation networks. *Transportation research record 1651* (1998), 49–53.

[2] Aggarwal, C. C., and Yu, P. S. Outlier detection for high dimensional data. In *SIGMOD Conference* (2001).

[3] Angiulli, F., and Fassetti, F. Exploiting domain knowledge to detect outliers. *Data Mining and Knowledge Discovery* (2013), 1–50.

[4] Angiulli, F., Greco, G., and Palopoli, L. Outlier detection by logic programming. *ACM Trans. Comput. Logic 9*, 1 (2007), 7.

[5] Appice, A., Ceci, M., Lanza, A., Lisi, F. A., and Malerba, D. Discovery of spatial association rules in geo-referenced census data: A relational mining approach. *Intelligent Data Analysis 7* (2003), 541–566.

[6] Bander, J. L., and White III, C. C. A heuristic search algorithm for path determination with learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 28*, 1 (1998), 131–134.

[7] Bast, H., Funke, S., Sanders, P., and Schultes, D. Fast routing in road networks with transit nodes. *Science 316*, 5824 (2007), 566.

[8] Berzal, F., Cubero, J.-C., and Marín, N. Anomalous association rules. In *IEEE ICDM Workshop Alternative Techniques for Data Mining and Knowledge Discovery* (2004).

[9] Blockeel, H., De Raedt, L., Jacobs, N., and Demoen, B. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery 3*, 1 (1999), 59–93.

[10] 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 16* (2002), 135–166.

[11] Blockeel, H., Dehaspe, L., Ramon, J., Struyfand, J., Assche, A. V., Vens, C., and Fierens, D. *The ACE Data Mining System, User's Manual*. DTAI, K.U.Leuven, March 2009.

[12] Brandes, U., Gaertler, M., and Wagner, D. Experiments on graph clustering algorithms. In *Algorithms - ESA 2003*, G. Battista and U. Zwick, Eds., vol. 2832 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 568–579.

[13] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA* (2000), W. Chen, J. F. Naughton, and P. A. Bernstein, Eds., ACM, pp. 93–104.

[14] Car, A., and Frank, A. General principles of hierarchical spatial reasoning: The case of wayfinding. In *Proceedings of the 6th International Symposium on Spatial Data Handling* (September 1994), Taylor and Francis.

[15] Caruso, C., and Malerba, D. A data mining methodology for anomaly detection in network data. In *KES '07: Knowledge-Based Intelligent Information and Engineering Systems and the XVII Italian Workshop on Neural Networks on Proceedings of the 11th International Conference* (Berlin, Heidelberg, 2007), Springer-Verlag, pp. 109–116.

[16] Ceci, M., and Appice, A. Spatial associative classification: propositional vs structural approach. *J. Intell. Inf. Syst. 27*, 3 (2006), 191–213.

[17] Chan, T. M. More algorithms for all-pairs shortest paths in weighted graphs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* (New York, NY, USA, 2007), STOC '07, ACM, pp. 590–598.

[18] Cheverst, K., Davies, N., Mitchell, K., Friday, A., and Efstratiou, C. Developing a context-aware electronic tourist guide: some issues and experiences. In *Proceedings of the Conference on Human Factors in Computing Systems* (2000), ACM, pp. 17–24.

[19] Cho, H.-J., and Lan, C.-L. Hybrid shortest path algorithm for vehicle navigation. *J. Supercomput. 49*, 2 (2009), 234–247.

[20] Chou, Y.-L., Romeijn, H. E., and Smith, R. L. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS Journal on Computing 10*, 2 (1998), 163–179.

[21] Clare, A., and King, R. D. Data mining the yeast genome in a lazy functional language. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages* (London, UK, 2003), Springer-Verlag, pp. 19–36.

[22] Dantzig, G. On the shortest route through a network. *Management Science 6* (1960), 187–190.

[23] Das, G., and Narasimhan, G. A fast algorithm for constructing sparse euclidean spanners. *Int. J. Comput. Geometry Appl. 7*, 4 (1997), 297–315.

[24] De Raedt, L., Blockeel, H., Dehaspe, L., and Van Laer, W. Three companions for data mining in first order logic. In *Relational Data Mining*, S. Džeroski and N. Lavrač, Eds. Springer-Verlag, 2001, pp. 105–139.

[25] De Raedt, L., and Dehaspe, L. Clausal discovery. *Mach. Learn. 26*, 2-3 (1997), 99–146.

[26] Dehaspe, L. *Frequent Pattern Discovery in First-Order Logic.* PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 1998.

[27] Dehaspe, L., and Toivonen, H. Discovery of frequent datalog patterns. *Data Min. Knowl. Discov. 3*, 1 (1999), 7–36.

[28] Deitch, R., and Ladany, S. The one-period bus touring problem: Solved by an effective heuristic for the orienteering tour problem and improvement algorithm. *European Journal of Operational Research 127*, 1 (2000), 69–77.

[29] Delafontaine, M., Nolf, G., van de Weghe, N., Antrop, M., and de Maeyer, P. Assessment of sliver polygons in geographical vector data. *Int. J. Geogr. Inf. Sci. 23*, 6 (2009), 719–735.

[30] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[31] Džeroski, S. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl. 5*, 1 (2003), 1–16.

[32] Ester, M., Frommelt, E., peter Kriegel, H., and Sander, J. Algorithms for characterization and trend detection in spatial databases. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD* (1998), pp. 44–50.

[33] Flinsenberg, I. I., van der Horst, M. M., Lukkien, J. J., and Verriet, J. J. Creating graph partitions for fast optimum route planning. *WSEAS Transactions on Computers 3*, 3 (2004), 569 – 574.

[34] Frank, R., Ester, M., and Knobbe, A. A multi-relational approach to spatial classification. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2009), ACM, pp. 309–318.

[35] Frank, R., Jin, W., and Ester, M. Efficiently mining regional outliers in spatial data. In *Advances in Spatial and Temporal Databases, 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16-18, 2007, Proceedings* (2007), D. Papadias, D. Zhang, and G. Kollios, Eds., vol. 4605 of *Lecture Notes in Computer Science*, Springer, pp. 112–129.

[36] Fu, L. *Real-time vehicle routing and scheduling in dynamic and stochastic traffic networks*. PhD thesis, University of Alberta, Edmonton, Alberta, 1996.

[37] Fu, L., Sun, D., and Rilett, L. R. Heuristic shortest path algorithms for transportation applications: state of the art. *Computers and Operations Research 33*, 11 (2006), 3324–3343.

[38] Gao, J., Jin, R., Zhou, J., Yu, J. X., Jiang, X., and Wang, T. Relational approach for shortest path discovery over large graphs. *PVLDB 5*, 4 (2011), 358–369.

[39] Geisberger, R., Sanders, P., Schultes, D., and Delling, D. Contraction hierarchies: faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th international conference on Experimental algorithms* (Berlin, Heidelberg, 2008), WEA'08, Springer-Verlag, pp. 319–333.

[40] Godart, J. Combinatorial optimisation based decision support system for trip planning. In *Information and Communication Technologies in Tourism 1999* (1999), D. Buhalis and W. Schertler, Eds., Springer, pp. 318–327.

[41] Goldberg, A. V., and Harrelson, C. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2005), SODA '05, Society for Industrial and Applied Mathematics, pp. 156–165.

[42] Goldberg, A. V., Kaplan, H., and Werneck, R. F. Reach for A*: Efficient point-to-point shortest path algorithms. In *Proceedings of the eighth Workshop on Algorithms Engineering and Experiments* (2006), vol. MSR-TR-200, Society for Industrial and Applied Mathematics, pp. 129–143.

[43] Gudmundsson, J., Levcopoulos, C., Narasimhan, G., and Smid, M. Approximate distance oracles for geometric spanners. *ACM Trans. Algorithms 4*, 1 (Mar. 2008), 10:1–10:34.

[44] Gutman, R. J. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004* (2004), L. Arge, G. F. Italiano, and R. Sedgewick, Eds., SIAM, pp. 100–111.

[45] Haglin, D. J., and Manning, A. M. On minimal infrequent itemset mining. In *Proceedings of the 2007 International Conference on Data Mining, DMIN 2007, June 25-28, 2007, Las Vegas, Nevada, USA* (2007), R. Stahlbock, S. F. Crone, and S. Lessmann, Eds., CSREA Press, pp. 141–147.

[46] Hahne, F., Nowak, C., and Ambrosi, K. Acceleration of the a*-algorithm for the shortest path problem in digital road maps. In *OR* (2007), J. Kalcsics and S. Nickel, Eds., Springer, pp. 455–460.

[47] Han, J., Koperski, K., and Stefanovic, N. Geominer: a system prototype for spatial data mining. *SIGMOD Rec. 26*, 2 (1997), 553–556.

[48] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions On Systems Science And Cybernetics 4*, 2 (1968), 100–107.

[49] Hochmair, H. H., and Navratil, G. Computation of scenic routes in street networks. In *Proceedings of the Geoinformatics Forum* (Salzburg, Austria, 2008), A. Car, G. Griesebner, and J. Strobl, Eds., Heidelberg: Wichmann Verlag, pp. 124–133.

[50] Huang, Y.-W., Jing, N., and Rundensteiner, E. A. Optimizing path query performance: graph clustering strategies. *Transportation Research Part C: Emerging Technologies 8*, 1–6 (2000), 381 – 408.

[51] Jacob, R., Marathe, M., and Nagel, K. A computational study of routing algorithms for realistic transportation networks. *J. Exp. Algorithmics 4* (1999), 6.

[52] Jagadeesh, G. R., Srikanthan, T., and Quek, K. H. Heuristic techniques for accelerating hierarchical routing on road networks. *IEEE Transactions on Intelligent Transportation Systems 3*, 4 (2002), 301–309.

[53] Jung, S., and Pramanik, S. An efficient path computation model for hierarchically structured topographical road maps. *Knowledge and Data Engineering, IEEE Transactions on 14*, 5 (2002), 1029–1046.

[54] Kannan, R., Vempala, S., and Veta, A. On clusterings-good, bad and spectral. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on* (2000), pp. 367–377.

[55] Karimi, H. A. Real-time optimal route computation: a heuristic approach. *ITS Journal 3*, 2 (1996), 111–27.

[56] Karimi, H. A., Sutovsky, P., and Durcik, M. Accuracy and performance assessment of a window-based heuristic algorithm for real-time routing in map-based mobile applications. In *Map-based Mobile Services*, L. Meng, A. Zipf, and S. Winter, Eds., Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 2008, pp. 248–266.

[57] Knorr, E. M., Ng, R. T., and Tucakov, V. Distance-based outliers: Algorithms and applications. *VLDB Journal: Very Large Data Bases 8*, 3–4 (2000), 237–253.

[58] Koh, Y. S., and Rountree, N. Finding sporadic rules using apriori-inverse. In *PAKDD* (2005), T. B. Ho, D. W.-L. Cheung, and H. Liu, Eds., vol. 3518 of *Lecture Notes in Computer Science*, Springer, pp. 97–106.

[59] Koh, Y. S., Rountree, N., and O'Keefe, R. A. Mining interesting imperfectly sporadic rules. *Knowl. Inf. Syst. 14*, 2 (2008), 179–196.

[60] Koperski, K., and Han, J. Discovery of spatial association rules in geographic information databases. In *Proc. 4th Int. Symp. Advances in Spatial Databases, SSD* (6–9 1995), M. J. Egenhofer and J. R. Herring, Eds., vol. 951, Springer-Verlag, pp. 47–66.

[61] Kuramochi, M., and Karypis, G. Finding frequent patterns in a large sparse graph. *Data Min. Knowl. Discov. 11*, 3 (2005), 243–271.

[62] Lansdowne, Z. F., and Robinson, D. W. Geographic decomposition of the shortest path problem, with an application to the traffic assignment problem. *Management Science 28*, 12 (1982), 1380–1390.

[63] Laporte, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research 59*, 3 (1992), 345–358.

[64] Laros, J. F. J. Unique factors in the human genome. Master's thesis, Leiden University, 2005.

[65] Lauther, U. An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. *IfGI prints, Institut für Geoinformatik, Universität Münster 22* (2004), 219–230.

[66] Lavrač, N., and Džeroski, S. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

[67] Lazarevic, A., Srivastava, J., Kumar, V., Banerjee, A., and Chandola, V. Data mining for anomaly detection (tutorial). In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (2008).

[68] Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., and hua Teng, S. On trip planning queries in spatial databases. In *Proc. of SSTD-05* (2005), Springer, pp. 273–290.

[69] Libura, M., van der Poort, E. S., Sierksma, G., and van der Veen, J. A. Stability aspects of the traveling salesman problem based on k-best solutions. *Discrete Applied Mathematics 87*, 1–3 (1998), 159 – 185.

[70] Lin, Y.-W. A qualitative enquiry into openstreetmap making. *New Rev. Hypermedia Multimedia 17*, 1 (Apr. 2011), 53–71.

[71] Lisi, F. A., and Malerba, D. Inducing multi-level association rules from multiple relations. *Machine Learning 55*, 2 (2004), 175–210.

[72] Liu, B. Route finding by using knowledge about the road network. *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans 27(4)* (1997), 436–448.

[73] Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W. *Geographic Information Systems and Science*. John Wiley & Sons, Apr. 2005.

[74] Maervoet, J. Rule induction for geographical databases. Master's thesis, Vrije Universiteit Brussel, 2007.

[75] Malaka, R., and Zipf, A. Deep map - challenging IT research in the framework of a tourist information system. *Information and Communication Technologies in Tourism* (2000), 15–27.

[76] Malerba, D., Esposito, F., Lanza, A., Lisi, F., and Appice, A. Empowering a gis with inductive learning capabilities: The case of ingens. *Journal of Computers, Environment and Urban Systems 27* (2003), 265–281.

[77] Malerba, D., Esposito, F., Lisi, F., and Appice, A. Mining spatial association rules in census data. *Research in Official Statistics 5*, 1 (2002), 19–44.

[78] Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., and Ito, M. A personal tourism navigation system to support traveling multiple destinations with time restrictions. In *Proceedings of AINA 2004* (2004), IEEE Computer Society, pp. 18–22.

[79] Maue, J., Sanders, P., and Matijevic, D. Goal-directed shortest-path queries using precomputed cluster distances. *J. Exp. Algorithmics 14* (Jan. 2010), 2:3.2–2:3.27.

[80] Monien, B., and Diekmann, R. A local graph partitioning heuristic meeting bisection bounds. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, PPSC 1997, March 14-17, 1997, Hyatt Regency Minneapolis on Nicollet Mall Hotel, Minneapolis, Minnesota, USA* (1997), SIAM.

[81] Münz, G., Li, S., and Carle, G. Traffic anomaly detection using k-means clustering. In *Proc. of Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 4. GI/ITG-Workshop MMBnet 2007* (Hamburg, Germany, Sept. 2007).

[82] Narasimhan, G., and Smid, M. Approximating the stretch factor of euclidean graphs. *SIAM J. Comput. 30*, 3 (May 2000), 978–989.

[83] Niaraki, A. S., and Kim, K. Ontology based personalized route planning system using a multi-criteria decision making approach. *ESWA 36*, 2 (2009), 2250–2259.

[84] Plantevit, M., Goutier, S., Guisnel, F., Laurent, A., and Teisseire, M. Mining unexpected multidimensional rules. In *DOLAP* (2007), I.-Y. Song and T. B. Pedersen, Eds., pp. 89–96.

[85] Pothen, A. Graph partitioning algorithms with applications to scientific computing. In *Parallel Numerical Algorithms* (1997), Kluwer Academic Press, pp. 323–368.

[86] Provost, F. J., and Kolluri, V. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery 3*, 2 (1999), 131–169.

[87] Qiao, M., Cheng, H., Chang, L., and Yu, J. X. Approximate shortest distance computing: A query-dependent local landmark scheme. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012* (2012), A. Kementsietsidis and M. A. V. Salles, Eds., IEEE Computer Society, pp. 462–473.

[88] Qiao, M., Cheng, H., and Yu, J. X. Querying shortest path distance with bounded errors in large graphs. In *Proceedings of the 23rd international conference on Scientific and statistical database management* (Berlin, Heidelberg, 2011), SSDBM'11, Springer-Verlag, pp. 255–273.

[89] Ramaswamy, S., Rastogi, R., and Shim, K. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec. 29*, 2 (2000), 427–438.

[90] Ramon, J. *Clustering and instance based learning in first order logic.* PhD thesis, K.U.Leuven, 2002.

[91] Rogers, S., and Langley, P. Personalized driving route recommendations. In *Proceedings of the AAAI Workshop on Recommender Systems* (1998), Madison, pp. 96–100.

[92] Sanders, P., and Schultes, D. Highway hierarchies hasten exact shortest path queries. In *Algorithms - ESA 2005*, G. S. Brodal and S. Leonardi, Eds., vol. 3669 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 568–579.

[93] Sanders, P., and Schultes, D. Engineering highway hierarchies. In *Proceedings of the 14th conference on Annual European Symposium - Volume 14* (London, UK, UK, 2006), ESA'06, Springer-Verlag, pp. 804–816.

[94] Sanders, P., and Schultes, D. Engineering fast route planning algorithms. In *Experimental Algorithms*, vol. 4525 of *LNCS*. Springer, 2007, pp. 23–36.

[95] Sankaranarayanan, J., and Samet, H. Distance oracles for spatial networks. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on* (2009), pp. 652–663.

[96] Schultes, D., and Sanders, P. Dynamic highway-node routing. In *Experimental Algorithms*, C. Demetrescu, Ed., vol. 4525 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 66–79.

[97] Sharifzadeh, M., and Shahabi, C. Processing optimal sequenced route queries using voronoi diagrams. *Geoinformatica 12*, 4 (2008), 411–433.

[98] Shcherbina, O., and Shembeleva, E. Modeling recreational systems using optimization techniques and information technologies. *Annals of OR* (2011), 1–21.

[99] Shekhar, S., Zhang, P., Huang, Y., and Vatsavai, R. R. *Data Mining: Next Generation Challenges and Future Directions*. AAAI/MIT Press, 2003, ch. Trends in Spatial Data Mining.

[100] Sommer, C., Verbin, E., and Yu, W. Distance oracles for sparse graphs. In *50th IEEE Symposium on Foundations of Computer Science (FOCS)* (2009), pp. 703–712.

[101] Souffriau, W., Maervoet, J., Vansteenwegen, P., Vanden Berghe, G., and Van Oudheusden, D. A mobile tourist decision support system for small footprint devices. In *Bio-Inspired Systems: Computational and Ambient Intelligence* (2009), vol. 5517 of *Lecture Notes in Computer Science*, Springer, pp. 1248–1255.

[102] Souffriau, W., and Vansteenwegen, P. Tourist trip planning functionalities: State-of-the-art and future. In *Current Trends in Web Engineering* (2010), F. Daniel and F. Facca, Eds., vol. 6385 of *LNCS*, Springer, pp. 474–485.

[103] Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., and Van Oudheusden, D. A personalised tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence 22*, 10 (2008), 964–985.

[104] Stolle, C., Karwath, A., and De Raedt, L. Classic'cl: An integrated ilp system. In *Proceedings of the 8th International Conference of Discovery Science* (2005), Springer-Verlag, pp. 354–362.

[105] Strauss, C. A note on hierarchical routing algorithms based on traverse-oriented road networks. *International Journal of Spatial Data Infrastructures Research 4* (2009), 239–264.

[106] Sun, Y., and Lee, L. Agent-based personalized tourist route advice system. In *SPRS Congress Istanbul 2004, Proceedings of Commission II* (2004), pp. 319–324.

[107] Suzuki, E. Undirected discovery of interesting exception rules. *IJPRAI 16*, 8 (2002), 1065–1086.

[108] Tarapata, Z. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *Int. J. Appl. Math. Comput. Sci. 17*, 2 (2007), 269–287.

[109] Thorup, M., and Zwick, U. Approximate distance oracles. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece* (2001), J. S. Vitter, P. G. Spirakis, and M. Yannakakis, Eds., ACM, pp. 183–192.

[110] van Dongen, S. M. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, The Netherlands, 2000.

[111] Vansteenwegen, P., Souffriau, W., and Van Oudheusden, D. The orienteering problem: A survey. *European Journal of Operational Research 209*, 1 (2011), 1 – 10.

[112] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., and Van Oudheusden, D. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research 196*, 1 (2009), 118–127.

[113] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., and Van Oudheusden, D. Iterated local search for the team orienteering problem with time windows. *Comput. Oper. Res. 36*, 12 (2009), 3281–3290.

[114] Vansteenwegen, P., and Van Oudheusden, D. The Mobile Tourist Guide: an OR Opportunity. *OR Insight 20*, 3 (2007), 21–27.

[115] Wagner, D., and Willhalm, T. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Proceedings of Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Lecture Notes in Computer Science* (2003), G. D. Battista and U. Zwick, Eds., vol. 2832, Springer, pp. 776–787.

[116] Zeitouni, K. *A survey of spatial data mining methods databases and statistics point of views.* IRM Press, Hershey, PA, United States, 2002, pp. 229–242.

[117] Zhan, B. F., and Noon, C. E. Shortest path algorithms: An evaluation using real road networks. *Transportation Science 32*, 1 (February 1998), 65–73.

[118] Zhu, C., Kitigawa, H., Papadimitriou, S., and Faloutsos, C. Outlier detection adaptive to users' intentions. In *Proceedings of the 15th IEICE Data Engineering Workshop* (2004).

# Curriculum vitae

Joris Maervoet obtained a master in Industrial Science Electronics/ICT (KdG, Antwerp) and started as an applied research associate at KaHo Sint-Lieven in 2002. From 2002 to 2007, he was involved in technology transfer projects on software development for mobile devices and mobile agents. In 2007 he graduated as a master in Computer Science at Vrije Universiteit Brussel. From 2007 to 2009 he worked on various technology transfer projects including relational data mining for quality maintenance in dynamic geographical databases (Tele Atlas, currently TomTom) and Java ME porting of a decision component for tourist trips (citytripplanner.com). In December 2009, he obtained a grant of more than 300k€ for a 4-year industrial PhD mandate at KU Leuven in close collaboration with the company RouteYou.com. This company manages a web 2.0 environment which enables users to create, share and use tourist routes in an interactive way.

Joris has a hands-on attitude of applying and valorising methods and concepts from academia in industry. His research interests include almost everything in the intersection of geoinformatics and artificial intelligence, as well as geographic information systems and relational data mining. He is also experienced in the organisation of conferences and seminars and in project-based teaching and the supervision of thesisses at the 'Faculty of Engineering Technology'. Joris is a member of the Combinatorial Optimisation and Decision Support (CODeS) research group at KU Leuven.

# List of publications

## Articles in internationally reviewed academic journals

Maervoet, J., Vens, C., Vanden Berghe, G., Blockeel, H., De Causmaecker, P. (2012). Outlier detection in relational data: a case study in geographical information systems. Expert Systems with Applications, 39 (5), art.nr. ESWA7039, 4718-4728.

## Technical reports

Maervoet, J., Christiaens, J., De Causmaecker, P., Vanden Berghe, G. Least squares approximate distance oracles for spatial networks.

Maervoet, J., Brackman, P., De Causmaecker, P., Verbeeck, K., Vanden Berghe, G. Wayfinding by multi-level heuristic node promotion in real road networks.

## Other academic books; as editor

Proceedings of BNAIC 2011. (De Causmaecker, P., Ed., Maervoet, J., Ed., Messelis, T., Ed., Verbeeck, K., Ed., Vermeulen, T., Ed.). Drongen: Nevelland.

## Papers at international scientific conferences and symposia, published in full in proceedings

Maervoet, J., Brackman, P., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G. (2013). Tour Suggestion for Outdoor Activities. In Liang, S. (Ed.), Wang, X. (Ed.), Claramunt, C. (Ed.), Lecture Notes in Computer Science: Vol. 7820. W2GIS 2013. Banff, AB, Canada, 4-5 April 2013 (pp. 54-63). Heidelberg, Germany: Springer.

Vermeulen, T., Vangheluwe, K., Maervoet, J., Verbeeck, K., Verhoeve, P., Stubbe, B. (2010). NuCiA - Nurse call simulation in agent environments. In Janssens, G. (Ed.), Ramaekers, K. (Ed.), Caris, A. (Ed.), Proceedings of the 2010 European Simulation and Modelling Conference. European Simulation and Modelling Conference. Hasselt, 25-27 October 2010 (pp. 276-279). Ostend, Belgium: Eurosis-eti.

Souffriau, W., Maervoet, J., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D. (2009). A mobile tourist decision support system for small footprint devices. In: Cabestany J., Sandoval F., Prieto A., Corchado J. (Eds.), Bio-inspired systems: computational and ambient intelligence. Berlin: Springer-verlag, 1248-1255.

Maervoet, J., De Causmaecker, P., Nowé, A., Vanden Berghe, G. (2008). Feasibility Study of Applying Descriptive ILP to Large Geographic Databases. Proceedings of the Workshop on Mining Multidimensional Data (MMD). European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD). Antwerp, 15-19 September 2008 (pp. 1-15).

## Demonstration papers at international scientific conferences and symposia, published in proceedings

Maervoet, J., Blomme, L., Verbeeck, K., Vanden Berghe, G., De Causmaecker, P. (2010). Road Network Hierarchy Generation by Distributed Agents for a Routing Application. Proceedings of BNAIC 2010. Benelux Conference on Artificial Intelligence. Luxembourg, 25-26 October 2010.

Maervoet, J., Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D. (2009). Tourist Decision Support for Mobile Navigation Systems: a Demonstration. In Calders, T. (Ed.), Tuyls, K. (Ed.), Pechenizkiy, M.

(Ed.), Proceedings of BNAIC 2009. Benelux Conference on Artificial Intelligence. Eindhoven, 29-30 October 2009 (pp. 393-394).

Maervoet, J., De Causmaecker, P., Vanden Berghe, G. (2008). A generic rule miner for geographic data (demonstration paper). In Nijholt, A. (Ed.), Pantic, M. (Ed.), Poel, M. (Ed.), Hondorp, H. (Ed.), Proceedings of BNAIC 2008. Belgian-Dutch Conference on Artificial Intelligence. Boekelo, 30-31 October 2008.

## Meeting abstracts, presented at other scientific conferences and symposia, published or not published in proceedings or journals

Maervoet, J., Baker, K., Vanden Berghe, G. (2013). Route Planning Enhancement through Collective Intelligence. LICT Scientific Symposium on Adaptivity in ICT. Heverlee, 11 September 2013.

Maervoet, J., De Causmaecker, P., Vanden Berghe, G. (2012). Structural heuristics for personalised routes. Doctoral Symposium 2012 (colocated with ECUMICT 2012). Gent, 23 March 2012.

Maervoet, J. (2008). A theoretical framework and algorithms for the detection of irregularities in relational data. Doctoral symposium - Onderzoek in de kijker bij KaHo Sint-Lieven en KHBO. Gent, 9 December 2008.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
COMBINATORIAL OPTIMISATION AND DECISION SUPPORT
Celestijnenlaan 200A box 2402
3001 LEUVEN, Belgium