SPATIAL DATABASES

TECHNOLOGIES, TECHNIQUES AND TRENDS

Yannis Manolopoulos,
Apostolos N. Papadopoulos
& Michael Gr. Vassilakopoulos

Spatial Databases:
Technologies, Techniques and Trends

Yannis Manolopoulos
Aristotle University of Thessaloniki, Greece

Apostolos N. Papadopoulos
Aristotle University of Thessaloniki, Greece

Michael Gr. Vassilakopoulos
Technological Educational Institute of Thessaloniki, Greece

IDEA GROUP PUBLISHING
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Mehdi Khosrow-Pour
Senior Managing Editor: Jan Travers
Managing Editor: Amanda Appicello
Development Editor: Michele Rossi
Copy Editor: Julie LeBlanc
Typesetter: Rachel Shepherd
Cover Design: Lisa Tosheff
Printed at: Integrated Book Technology

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033-1240
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: http://www.irm-press.com

and in the United Kingdom by
IRM Press (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 3313
Web site: http://www.eurospan.co.uk

Library of Congress Cataloging-in-Publication Data

Spatial databases : technologies, techniques and trends / Yannis Manolopoulos, Apostolos N. Papadopoulos and Michael Gr. Vassilakopoulos, Editors.
p. cm.
Includes bibliographical references and index.
ISBN 1-59140-387-1 (h/c) -- ISBN 1-59140-388-X (s/c) -- ISBN 1-59140-389-8 (ebook)
1. Database management. 2. Geographic information systems. I. Manolopoulos, Yannis, 1957- II. Papadopoulos, Apostolos N. III. Vassilakopoulos, Michael Gr.
QA76.9.D3S683 2004
005.74--dc22 2004021989

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Spatial Databases: Technologies, Techniques and Trends

Table of Contents

Section II: Indexing Techniques

Section III: Query Processing and Optimization

Section IV: Moving Objects

Section V: Data Mining

Preface

Spatial database systems has been an active area of research over the past 20 years. A large number of research efforts have appeared in literature aimed at effective modelling of spatial data and efficient processing of spatial queries. This book investigates several aspects of a spatial database system, and includes recent research efforts in this field. More specifically, some of the topics covered are: spatial data modelling; indexing of spatial and spatio-temporal objects; data mining and knowledge discovery in spatial and spatio-temporal databases; management issues; and query processing for moving objects. Therefore, the reader will be able to get in touch with several important issues that the research community is dealing with. Moreover, each chapter is self-contained, and it is easy for the non-specialist to grasp the main issues.

The authors of the book's chapters are well-known researchers in spatial databases, and have offered significant contributions to spatial database literature. The chapters of this book provide an in-depth study of current technologies, techniques and trends in spatial and spatio-temporal database systems research. Each chapter has been carefully prepared by the contributing authors, in order to conform with the book's requirements.

Intended Audience

This book can be used by students, researchers and professionals interested in the state-of-the-art in spatial and spatio-temporal database systems. More specifically, the book will be a valuable companion for postgraduate students studying spatial database issues, and for instructors who can use the book as a refer-

ence for advanced topics in spatial databases. Researchers in several related areas will find this book useful, since it covers many important research directions.

Prerequisites

Each chapter of the book is self-contained, to help the reader focus on the corresponding issue. Moreover, the division of the chapters into sections is very convenient for those focusing on different research issues. However, at least a basic knowledge of indexing, query processing and optimization in traditional database systems will be very helpful in more easily understanding the issues covered by each chapter.

Overview of Spatial Database Issues

Spatial database management systems aim at supporting queries that involve the space characteristics of the underlying data. For example, a spatial database may contain polygons that represent building footprints from a satellite image, or the representation of lakes, rivers and other natural objects. It is important to be able to query the database by using predicates related to the spatial and geometric characteristics of the objects.

To handle such queries, a spatial database system is enhanced by special tools. These tools include new data types, sophisticated indexing mechanisms and algorithms for efficient query processing that differ from their counterparts in a conservative alphanumeric database. The contribution of the research community over the past 20 years includes a plethora of significant research results toward this goal.

An important research direction in spatial databases is the representation and support of the time dimension. In many cases, objects change their locations and shape. In order to query past or future characteristics, effective representation and query processing techniques are required. A spatial database enhanced by tools to incorporate time information is called a *spatio-temporal* database system. The applications of spatio-temporal databases are very significant, since such systems can be used in location-aware services, traffic monitoring, logistics, analysis and prediction. Indexing techniques for pure spatial datasets cannot be directly applied in a spatio-temporal dataset, because time must be supported efficiently.

Apart from supporting queries involving space and time characteristics of the underlying dataset, similarity of object movement has also been studied in literature. The target is to determine similar object movement by considering the trajectories of the moving objects. The similarity between two object trajectories is a very important tool that can help reveal similar behavior and define clusters of objects with similar motion patterns.

The research area of data mining studies efficient methods for extracting knowledge from a set of objects, such as association rules, clustering and prediction. The application of data mining techniques in spatial data yielded the interesting research field of spatial data mining. Recently, spatio-temporal data mining has emerged, to take into consideration the time dimension in the knowledge extraction process.

Several of the aforementioned research issues in spatial databases are covered by this book.

Book Organization

The book is composed of 13 chapters, organized in five major sections according to the research issue covered:

I) Modelling and Systems
II) Indexing Techniques
III) Query Processing and Optimization
IV) Moving Objects
V) Data Mining

In the sequel we describe briefly the topics covered in each section, giving the major issues studied in each chapter.

Section I focuses on modelling and system issues in spatial databases.

Chapter I identifies properties that a spatial data model, dedicated to support spatial data for cartography, topography, cadastral and relevant applications, should satisfy. The properties concern the data types, data structures and spatial operations of the model. A survey of various approaches investigates mainly the satisfaction of these properties. An evaluation of each approach against these properties also is included.

In Chapter II the authors study the impact of the Web to Geographic Information Systems (GIS). With the phenomenal growth of the Web, rich data sources

on many subjects have become available online. Some of these sources store daily facts that often involve textual geographic descriptions. These descriptions can be perceived as indirectly georeferenced data – e.g., addresses, telephone numbers, zip codes and place names. This chapter's focus is on using the Web as an important source of urban geographic information. Additionally, proposals to enhance urban GIS using indirectly georeferenced data extracted from the Web are included. An environment is described that allows the extraction of geospatial data from Web pages, converts them to XML format and uploads the converted data into spatial databases for later use in urban GIS. The effectiveness of this approach is demonstrated by a real urban GIS application that uses street addresses as the basis for integrating data from different Web sources, combining the data with high-resolution imagery.

Section II contains three chapters that study efficient methods for indexing spatial and spatio-temporal datasets.

Chapter III studies object-relational indexing as an efficient solution to enable spatial indexing in a database system. Although available extensible indexing frameworks provide a gateway for seamless integration of spatial access methods into the standard process of query optimization and execution, they do not facilitate the actual implementation of the spatial access method. An internal enhancement of the database kernel is usually not an option for database developers. The embedding of a custom block-oriented index structure into concurrency control, recovery services and buffer management would cause extensive implementation efforts and maintenance cost, at the risk of weakening the reliability of the entire system. The authors present the paradigm of object-relational spatial access methods that perfectly fits with the common relational data model and is highly compatible with the extensible indexing frameworks of existing object-relational database systems, allowing the user to define application-specific access methods.

Chapter IV contains a survey of quadtree uses in the image domain, from image representation to image storage and content-based retrieval. A quadtree is a spatial data structure built by a recursive decomposition of space into quadrants. Applied to images, it allows representing image content, compacting or compressing image information, and querying images. For 13 years, numerous image-based approaches have used this structure. In this chapter, the authors underline the contribution of quadtree in image applications.

With the abundance of low-cost storage devices, a plethora of applications that store and manage very large multi-dimensional trajectory (or time-series) datasets have emerged recently. Examples include traffic supervision systems, video surveillance applications, meteorology and more. Thus, it is becoming essential to provide a robust trajectory indexing framework designed especially for performing similarity queries in such applications. In this regard, Chapter V presents an indexing scheme that can support a wide variety of (user-

customizable) distance measures, while at the same time guaranteeing retrieval of similar trajectories with accuracy and efficiency.

Section III studies approximate computation of distanced-based queries and algorithms, cost models and optimization for spatial joins.

Chapter VI studies the problem of approximate query processing for distance-based queries. In spatial database applications, the similarity or dissimilarity of complex objects is examined by performing distance-based queries (DBQs) on data of high dimensionality (a generalization of spatial data). The R-tree and its variations are commonly cited as multidimensional access methods that can be used for answering such queries. Although the related algorithms work well for low-dimensional data spaces, their performance degrades as the number of dimensions increases (dimensionality curse). To obtain acceptable response time in high-dimensional data spaces, algorithms that obtain approximate solutions can be used. This chapter reviews the most important approximation techniques for reporting sufficiently good results quickly. The authors focus on the design choices of efficient approximate DBQ algorithms that minimize response time and the number of I/O operations over tree-like structures. The chapter concludes with possible future research trends in the approximate computation of DBQs.

Chapter VII describes algorithms, cost models and optimization techniques for spatial joins. Joins are among the most common queries in Spatial Database Management Systems. Due to their importance and high processing cost, a number of algorithms have been proposed covering all possible cases of indexed and non-indexed inputs. The authors first describe some popular methods for processing binary spatial joins, and provide models for selectivity and cost estimation. Then, they study the evaluation of multiway spatial joins by integrating binary algorithms and synchronous tree traversal. Going one step further, the authors show how analytical models can be used to combine the various join operators in optimal evaluation plans.

Section IV deals with moving objects databases, and studies efficient algorithms, management issues and applications.

Chapter VIII presents the applications of Moving Objects Databases (MODs) and their functionality. Miniaturization of computing devices and advances in wireless communication and sensor technology are some of the forces propagating computing from the stationary desktop to the mobile outdoors. Some important classes of new applications that will be enabled by this revolutionary development include location-based services, tourist services, mobile electronic commerce and digital battlefield. Some existing application classes that will benefit from the development include transportation and air traffic control, weather forecasting, emergency response, mobile resource management and mobile workforce. Location management, i.e., the management of transient location information, is an enabling technology for all these applications. Loca-

tion management also is a fundamental component of other technologies, such as fly-through visualization, context awareness, augmented reality, cellular communication and dynamic resource discovery. MODs store and manage the location as well as other dynamic information about moving objects.

Chapter IX presents several important aspects toward simple and incremental nearestneighbor searches for spatio-temporal databases. More specifically, the authors describe the algorithms that already have been proposed for simple and incremental nearest-neighbor queries, and present a new algorithm. Finally, the chapter studies the problem of keeping a query consistent in the presence of insertions, deletions and updates of moving objects. Applications of MODs have rapidly increased, because mobile computing and wireless technologies nowadays are ubiquitous.

Chapter X deals with important issues pertaining to the management of moving objects datasets in databases. The design of representative benchmarks is closely related to the formal characterization of the properties (i.e., distribution, speed, nature of movement) of these datasets; uncertainty is another important aspect that conditions the accuracy of the representation and therefore the confidence in query results. Finally, efficient *index* structures, along with their compatibility with existing software, is a crucial requirement for spatio-temporal databases, as it is for any other kind of data.

Section V, the final section of the book, contains two chapters that study the application of data mining techniques to spatio-temporal databases.

Recent interest in spatio-temporal applications has been fueled by the need to discover and predict complex patterns that occur when we observe the behavior of objects in the three-dimensional space of time and spatial coordinates. Althoughcomplex and intrinsic relationships among the spatio-temporal data limit the usefulness of conventional data mining techniques to discover the patterns in the spatio-temporal databases, they also lead to opportunities for mining new classes of patterns. Chapter XI provides a survey of the work done for mining patterns in spatial databases and temporal databases, and the preliminary work for mining patterns in spatio-temporal databases. The authors highlight the unique challenges of mining interesting patterns in spatio-temporal databases. Two special types of spatio-temporal patterns are described: location-sensitive sequence patterns and geographical features for location-based service patterns.

The spatio-temporal prediction problem requires that one or more future values be predicted for time series input data obtained from sensors at multiple physical locations. Examples of this type of problem include weather prediction, flood prediction, network traffic flow, etc. Chapter XII provides an overview of this problem, highlighting the principles and issues that come into play in spatio-temporal prediction problems. The authors describe recent work in the area of flood prediction to illustrate the use of sophisticated data mining techniques that

have been examined as possible solutions. The authors argue the need for further data mining research to attack this difficult problem.

In Chapter XIII, the authors review similarity learning in spatial databases. Traditional exact-match queries do not conform to the exploratory nature of GIS datasets. Non-adaptable query methods fail to capture the highly diverse needs, expertise and understanding of users querying for spatial datasets. Similarity-learning algorithms provide support for user preference and therefore should be a vital part in the communication process of geospatial information. More specifically, the authors address machine learning as applied in the optimization of query similarity. Appropriate definitions of similarity are reviewed. Moreover, the authors position similarity learning within data mining and machine-learning tasks. Furthermore, prerequisites for similarity-learning techniques based on the unique characteristics of the GIS domain are discussed.

How to Read This Book

The organization of the book has been carefully selected to help the reader. However, it is not mandatory to study the topics in their order of appearance. If the reader wishes to perform an in-depth study of a particular subject then he/she could focus on the corresponding section.

What Makes This Book Different

The reader of this book will get in touch with significant research directions in the area of spatial databases. The broad field of topics covered by important researchers is an important benefit. In addition to pure spatial concepts, spatio-temporal issues also are covered, allowing the reader to make his/her comparisons with respect to the similarities and differences of the two domains (i.e., spatial and spatio-temporal databases). Each chapter covers the corresponding topic to a sufficient degree, giving the reader necessary background knowledge for further reading.

The book covers important research issues in the field of spatial database systems. Since each book chapter is self-contained, it is not difficult for the non-expert to understand the topics covered. Although the book is not a textbook, it can be used in a graduate or a postgraduate course for advanced database issues.

A Closing Remark

The authors have made significant efforts to provide high-quality chapters, despite space restrictions. These authors are well-known researchers in the area of spatial and spatio-temporal databases, and they have offered significant contributions to the literature. We hope that the reader will gain the most out of this effort.

Yannis Manolopoulos, PhD
Apostolos N. Papadopoulos, PhD
Michael Vassilakopoulos, PhD
Thessaloniki, Greece
2004

Acknowledgments

The editors are grateful to everyone who helped in the preparation of this book. First, we would like to thank the chapter authors for their excellent contributions and their collaboration during the editing process. We also would like to thank the reviewers, whose comments and suggestions were valuable in improving the quality and presentation of the chapters. Moreover, we are grateful to Michele Rossi from Idea Group Publishing for her help in completing this project. Finally, we would like to thank all our colleagues for their comments regarding the issues covered in this book.

Yannis Manolopoulos, PhD
Michael Vassilakopoulos, PhD
Apostolos N. Papadopoulos, PhD
Thessaloniki, Greece
May 2004

Section I

Modelling and Systems

Chapter I

Survey on Spatial Data Modelling Approaches

Jose R. Rios Viqueira, University of A Coruña, Spain

Nikos A. Lorentzos, Agricultural University of Athens, Greece

Nieves R. Brisaboa, University of A Coruña, Spain

Abstract

The chapter identifies properties that a spatial data model, dedicated to support spatial data for cartography, topography, cadastral and relevant applications, should satisfy. The properties concern the data types, data structures and spatial operations of the model. A survey of various approaches investigates mainly the satisfaction of these properties. An evaluation of each approach against these properties also is included.

Introduction

A lot of research has been undertaken in recent years for the management of spatial data. Initial approaches in the area of GIS exhausted their efforts in the precise geometric representation of spatial data and in the implementation of operations between spatial objects. Subsequently, only primitive effort was made on the association of spatial data with conventional data. As a consequence, the management of geographic data had to be split into two distinct types of processing, one for the spatial data and another for the attributes of conventional data and their association with spatial data. Effort to define a formal and expressive language for the easy formulation of queries was almost missing and, therefore, too much programming was required. Finally, even the processing of spatial data lacked an underlying formalism. On the other hand, efficient processing of conventional data can only be achieved from within a Database Management System (DBMS). Besides, due to its complexity, the management of spatial data is not possible from within a conventional DBMS.

Because of this, a new research effort was undertaken in the area of *spatial databases*. Such effort covered various sectors, such as the design of efficient physical data structures and access methods, the investigation of query processing and optimization techniques, visual interfaces and so forth. All these approaches inevitably addressed spatial data modelling issues in an *indirect* way, in that spatial data modelling was not their primary objective. However, a *direct* way can also be identified, in that research has also been undertaken dedicated solely to the definition of data models.

This chapter surveys and evaluates spatial data modelling approaches in either of these types. Wherever applicable, the restriction of spatio-temporal models to the management of spatial data is also reviewed. In particular, properties concerning the data types considered, the data structures used and the operations supported by a data model for the management of cartography, topography, cadastral and relevant applications, are identified in the background section. A relevant review and evaluation of spatial data modelling approaches, *GIS-centric* and *DBMS-centric*, follow in the next two sections. Future trends are discussed in the fifth section, and conclusions are drawn in the last section.

Background

Traditional cartography, topography, cadastral and relevant applications require the processing of data that can geometrically be represented on a 2-d plane as

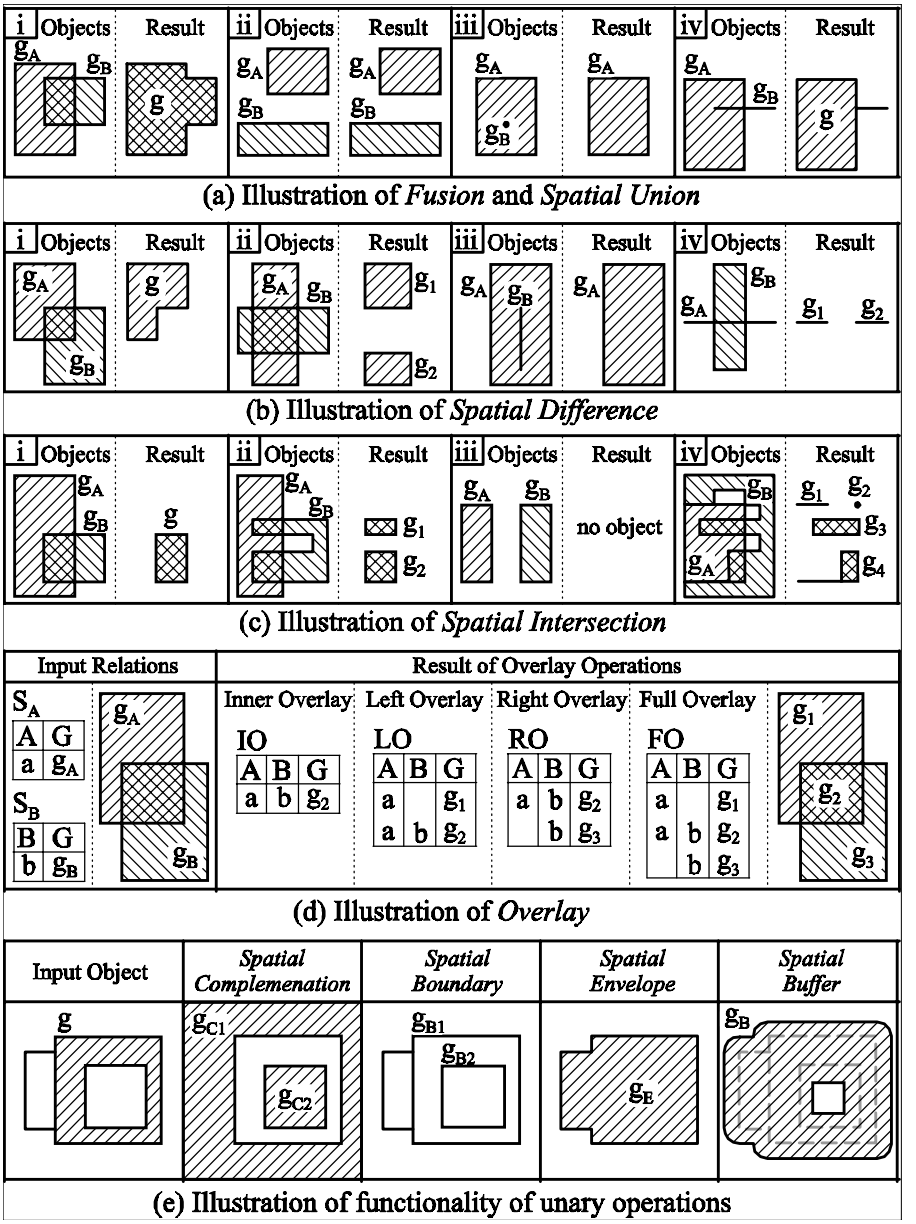
a point, line or surface. For the objectives of this chapter, every such piece of data, and any set of them as well, is termed *spatial data* or (*spatial*) *object*. This data is distinguished from *conventional data*, such as a name (for example, of a city, river, lake), a number (population of a city, supply of a river, depth of a lake), a date, and so forth. Data modelling requires specifying at minimum data types, data structures and operations.

The same is true for spatial data. However, spatial data have much individuality. To provide a few examples, consider spatial data of the three distinct common types: *point*, *line* and *surface*. Consider also Figure 1, which depicts some commonly used operations on spatial data (termed in this chapter *spatial operations*). It is then noted that the result of an operation between two spatial objects does not necessarily yield only one such object, but it may consist of two (Figure 1(a) case (ii), Figure 1(b) cases (ii) and (iv), Figure 1(c) case (ii)), more than two (Figure 1(c) case (iv)) and perhaps none (Figure 1(c) case (iii)). Also, the data type of the result objects may not necessarily match that of the input object(s) (Figure 1(a) case (iv) and Figure 1(c) case (iv)). Finally, the result of an operation may also contain objects that are combinations of surfaces with lines termed, for the objectives of this chapter, *hybrid surfaces* (Figure 1(a) case (iv), 1(c) case (iv)).

To face this individuality and at the same time define closed spatial operations, many distinct spatial data modelling approaches have been proposed. Many of them have the following characteristics: (i) They adopt *set-based* data types, such as *set of points*, *set of surfaces*, and so forth. (ii) They use either complex data structures to record spatial data or two types of such structures, one to record spatial and another to record conventional data. (iii) They define operations that apply to spatial data of one specific type; for example, *Overlay* only between surfaces. Other operations discard part of the result; for example, the point and line parts produced by the *spatial intersection* of two surfaces (Figure 1(c) case (iv)). However, a data model should be simple, and enable a most accurate mapping of the real world (Tsichritzis & Lochovsky, 1982). As opposed to the above observations, it is estimated that a spatial model should satisfy the following properties:

- **Spatial Data Types:** It should support the *point*, *line* and *surface* types, since in daily practice people are familiar with the use of these objects.
- **Data Structures:** They should be simple. As opposed to the First Normal Form (1NF) relational model, for example, it is noticed that a nested model, though more powerful, is more difficult to both implement and use. Similarly, it is penalizing for the user to process two distinct data structures.
- **Spatial Operations:** They should apply to structures containing any type of spatial data. Two examples: It is practical to (i) apply *Overlay* to lines,

Figure 1. Illustration of operations on spatial data



and (ii) apply an operation to two spatial objects of a different type, such as to compute the intersection of a surface with a line. Finally, pieces of data should not be discarded from the result of an operation.

Relevant to the operations that should be supported, it is estimated that for topographic, cartographic, cadastral and relevant applications, with which this chapter is mainly concerned, a spatial data model should support at least those in Figure 1. Indeed, many researchers have proposed the operations in Figure 1(a)-(d), which also match actual user requirements. Fewer researchers have proposed the remaining operations, but the authors estimate that they have general practical interest. Some explanations on these operations are the following: As opposed to *Spatial Union*, *Fusion* (Figure 1(a)) returns the results indicated only in the case that the pieces of conventional data, with which spatial data are associated, are identical. The *subtraction* of a point or line from a surface should return the surface itself (Figure 1(b) case (iii)). Indeed, it does not make sense to consider surfaces with missing points or lines. A similar remark applies to the subtraction of points from lines. Tables are used in the four *Overlay* operations to show the association of spatial with conventional data. Finally, the illustration of *Spatial Buffer* (Figure 1(e)) considers a distance of d = 1.

A brief review of various approaches for the management of spatial data, which follows, focuses mainly on the spatial data types considered, data structures used and support of the spatial operations shown in Figure 1. Wherever estimated to be necessary, more operations of a data model are presented. An evaluation of each approach also is given in Figure 2. The evaluation is based on the following criteria: (i) Support of *point*, *line* and *surface* types. (ii) Use of simple data structures, as opposed to the use of complex or more than one type of structure. (iii) Application of an operation to all types of spatial data, without discarding any part of the result. In Figure 2, a 'Y', 'N' or 'P' denotes, respectively, that a property is *satisfied*, *not satisfied* or satisfied *partially*. 'N/A' denotes that the property does not apply to the approach under consideration. Finally, '?' denotes that satisfaction of the property is not clear from the literature. Note that the evaluation was a hard task, due to the lack of formalism. To ease discussion, the approaches have been divided into two major classes, *GIS-centric* and *DBMS-centric* (IBM, 1998), and are reviewed separately in the next two sections.

	Tomlin 1990	Erwig & Schneider 1997	d'Onofrio & Pourabbas 2001	Hadzilacos & Tryfona 1996	ESRI 2003	Intergraph Corp. 2002	Bentley Systems 2001	GRASS 2002.	Kemp & Kowalczyk 1994	MapInfo Corp. 2002	Güting & Schneider 1995	Güting et al. 2000	Worboys 1994	Larue et al. 1993	Egenhofer 1994	Roussopoulos et al. 1988	Scholl & Voisard 1992	Böhlen et al. 1998	Chen & Zaniolo 2000	Gargano et al. 1991	Svensson & Huang 1991	Güting 1988	Chan & Zhu 1996	Grumbach et al. 1998	Kuper et al. 1998	van Roessel 1994	Scholl & Voisard 1989	Yeh & de Cambray 1995	ISO/IEC 2002	Oracle Corp. 2000	IBM 2001b	PostgreSQL 2001	Cheng & Gadia 1994	OpenGIS 1999	IBM 2001a	Yjlbrief& vanOosterom 1992	Park et al. 1998	Voigtmann 1997
Point	N	N	N	Y	Y	N	N	N	Y	N	N	Y	N	N	?	?	N	?	N	N	N	Y	Y	N	N	N	N	Y	N	Y	N	Y	N	Y	Y	N	N	Y
Line	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	?	?	N	?	N	N	P	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Surface	N	Y	Y	Y	N	N	N	Y	N	N	N	N	N	N	?	?	N	?	N	N	P	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Simple Structures	N/A	N/A	N/A	N	Y	Y	N	N/A	N	N	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Fusion	P	P	P	P	P	Y	N	N/A	N	P	P	P	N	N	N	N	N	Y	P	P	P	N	Y	Y	Y	P	Y	Y	Y	Y	P	P	P	P	P	P	P	P
Spatial Union	N/A	N/A	N	P	P	Y	N	N/A	N	P	P	P	N	Y	N	N	N	P	N	P	P	N	N	Y	Y	P	Y	Y	Y	Y	P	P	P	P	P	P	P	P
Spatial Difference	N/A	N/A	N	P	N	N	N	N/A	N	P	P	P	N	Y	N	N	N	P	N	P	P	N	N	Y	Y	P	Y	Y	Y	Y	P	P	P	P	P	P	P	P
Spatial Intersect	N/A	N/A	N	P	N	Y	P	N/A	N	N	Y	Y	P	N	N	P	P	P	Y	P	P	P	Y	Y	Y	P	Y	Y	Y	Y	P	P	P	P	P	P	P	P
Inner Overlay	Y	P	N	P	P	Y	P	P	N	N	Y	Y	P	Y	N	P	P	P	Y	P	P	P	Y	Y	Y	P	Y	Y	Y	Y	P	P	P	P	P	P	P	P
Left Overlay	Y	P	N	P	P	Y	P	P	N	N	P	P	N	Y	N	N	N	P	Y	P	P	N	N	Y	Y	P	Y	N	N	N	P	P	N	N	N	N	N	N
Right Overlay	Y	P	N	P	P	Y	P	P	N	N	P	P	N	Y	N	N	N	P	Y	P	P	N	N	Y	Y	P	Y	N	N	N	P	P	N	N	N	N	N	N
Full Overlay	Y	P	N	P	P	Y	P	P	N	N	P	P	N	Y	N	N	N	P	Y	P	P	N	N	Y	Y	P	Y	N	N	N	P	P	N	N	N	N	N	N
Complementation	N/A	N/A	N	?	N	N	N	N/A	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Boundary	N/A	N/A	N	?	?	N	N	N/A	N	N	Y	Y	?	N	P	N	Y	N	N	N	P	N	Y	Y	Y	N	P	N	P	P	N	N	P	P	P	P	P	P
Envelope	N	N	N	N	Y	Y	N	N	N	N	Y	N	N	N	N	N	N	N	N	Y	N	Y	Y	N	Y	P	N	N	Y	Y	N	N	Y	Y	N	N	N	N
Buffer	Y	N	N	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	Y	P	Y	N	Y	Y	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	N	Y

Figure 2. Evaluation of spatial approaches

GIS-Centric Approaches

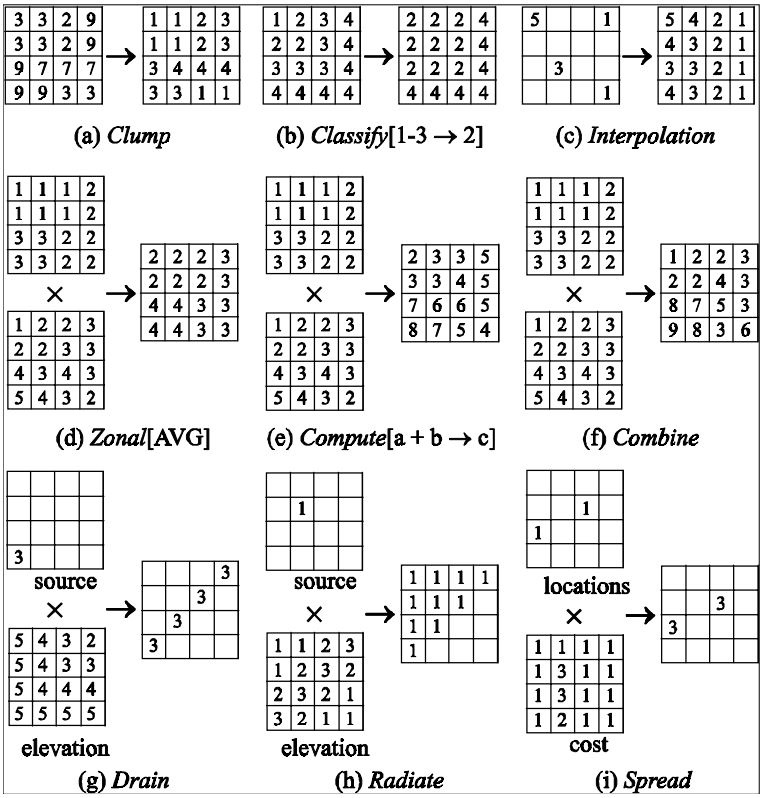
GIS-centric approaches are dedicated solely to the management of spatial data (IBM, 1998). *Specialized* data structures also are used to associate spatial with conventional data, but the handling of these structures takes place outside the GIS.

In one informal approach (Tomlin, 1990), *map layers* (termed simply *maps*) and operations on them are described at a conceptual level. A *map m* can be seen as a set of pairs (p, v), where p is a *location* (a 2-d point in the plane) and *v* is a number assigned to *p*, which indicates a property of *p*. Distinct maps are used to record distinct properties of locations, such as height, degree of pollution, and so forth. The approach enables recording properties of areas that change gradually from one location to another, termed *continuous changes*. Spatial data types are not defined. A *zone* of *m* is a set of pairs $Z = \{(p_1, v), (p_2, v), \ldots, (p_k, v)\}$ (adjacent or not) with identical values on the second coordinate. An open-ended set of operations is proposed. They all apply to maps and produce a new map. The approach classifies operations into four categories: (i) *Local*: The value of each *location p* depends on the value of the same *location p* in one or more input *maps*. (ii) *Zonal*: The result value of each *location p* depends on the values of the *locations* contained in the *zone* of *p* in one or more input *maps*. (iii) *Focal*: The result value of each *location p* depends on the values of the *locations* contained in the *neighbourhood* of *p* in one or more input *maps*. (iv) *Incremental*: They extend the set of *Focal* operations by taking into account the *type* of zone at each *location*. One of the local operations resembles *Full Overlay*.

Implementations based on Tomlin (1990) are Grass (2002), Keigan Systems (2002), Lorup (2000), McCoy and Johnston (2001), and Red Hen Systems (2001). A map is now modelled as a 2-d raster *grid* data structure, which represents a partition of a given rectangular area into a matrix of a finite set of squares, called *cells* or *pixels*. Each cell represents one of Tomlin's locations (Figure 3). All these approaches consider only surfaces. Examples of operations on grids are shown in Figure 3. Note that the functionality of operation *Combine* (Figure 3(f)) resembles that of *Full Overlay* on surfaces.

In Erwig and Schneider (1997), a map (called *spatial partition*) of a given area is defined as a set of non-overlapping, adjacent surfaces. Each such surface is associated with a tuple of conventional data. Surfaces associated with the same conventional data merge automatically into a single surface. *Point* and *Line* types are not defined. Three primitive operations are defined and, based on them, a representative functionality for map management is achieved (Figure 4), as proposed earlier in Scholl and Voisard (1989). One operation is *Full Overlay* (Figure 4(a)). A similar approach is the restriction to spatial data management

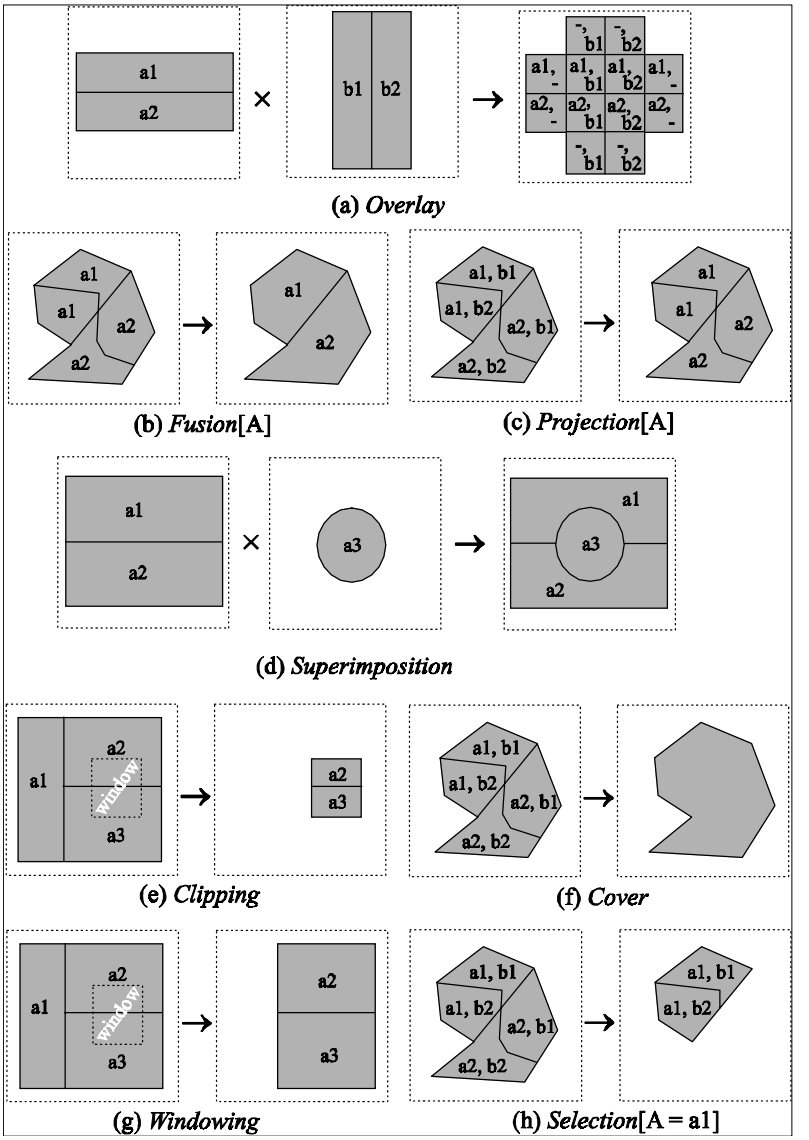
Figure 3. Examples of operations on raster grids



of the spatio-temporal model (d'Onofrio & Pourabbas, 2001). It considers maps of surfaces or lines, but it does not achieve the functionality of all the operations in Figure 4.

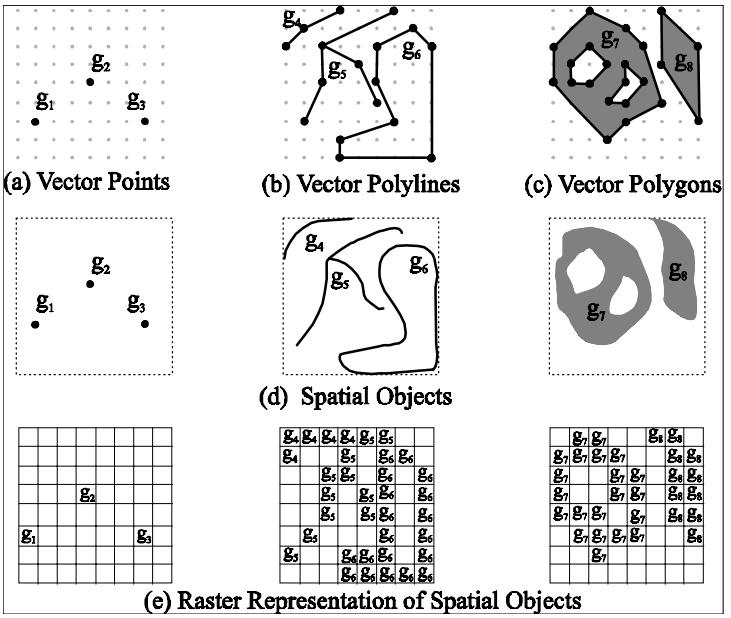
Point, *simple polyline* and *polygon* data types (Figure 5(a-c)) are proposed in Hadzilacos and Tryfona (1996). A map (called *layer*) M is defined as a mapping from a set of spatial values G to the Cartesian product of a set of conventional attributes (M: G $\rightarrow C_1,C_2,...,C_n$). Hence, a map can be seen as a relation with just one spatial attribute G. Operations on maps also are defined. Operation *Attribute derivation* (*Spatial computation*) enables the application of conventional (spatial) functions and predicates. Operation *Reclassification* merges into one all those tuples of a layer that have identical values in a given attribute and also are associated to adjacent spatial objects (Figure 4(b)). It can apply only to layers of type *simple polyline* or *polygon*. Operation *Overlay* (Figure 4(a)) or *Full Overlay* (Figure 1(d)) applies to two maps L_1 and L_2 of any data type. Its result is the union of three sets, (i) I, consisting of the pieces of spatial objects

Figure 4. Representative operations on maps



both in L_1 and L_2, (ii) L, consisting of the pieces of spatial objects in L_1 that are not inside the spatial objects in L_2, and (iii) R, consisting of the pieces of spatial objects in L_2 that are not inside the spatial objects in L_1. A similar approach is the restriction to spatial data management of the spatio-temporal model (Kemp & Kowalczyk, 1994).

Figure 5. Representation of spatial objects in various approaches



There are some other approaches, similar to the previous one. Differences are the following: The ESRI (2003) approach considers data types of the form *point*, *set of points*, *set of lines* and *set of surfaces* (Figure 5(a-c)) and a large set of operations. To illustrate operation *Overlay*, consider layer L_1, with objects of any type; layer L_2, consisting of only surfaces; and the sets I, L and R of the previous paragraph. Then each of the *Overlay* operations is associated with one of the result sets I, I ∪ L, I ∪ R, I ∪ L ∪ R. Operation *Erase* yields a new map with the pieces of the spatial objects in L_1 that are outside all the surfaces in L_2. *Update* yields the *Superimposition* (Figure 4(d)) of two *compatible* maps. Other functionalities are *Buffer* (Figure 1(c)); *Clipping* (Figure 4(e)); *Cover* (Figure 4(f)), one that yields the Voronoi diagram of a set of points; and operation *Reclassification*. In place of the ESRI (2003) *point* data type, the commercial GIS described in Intergraph Corp. (2002) supports a type of the form *set of spatial objects*. Finally, this is the only one supported in MapInfo Corp. (2002) and Bentley Systems (2001).

DMBS-Centric Approaches

DBMS-centric approaches form the third generation of spatial management systems (IBM 1998), in which spatial data is simply another data type within a DBMS environment. The approaches consider the data structures of some underlying data model (relational, object-oriented, and so forth) that usually incorporates spatial data types. This way, they enable the association of spatial with conventional data and take full advantage of the database technology. At the same time, they lack the flexibility of GIS-centric approaches for the management of maps. Operations are usually defined on either spatial objects or data structures.

The approach in Güting and Schneider (1995) is actually independent of a specific underlying data model. Hence, it restricts only to the definition of spatial data types and to operations on them. It considers a vector-based spatial representation and defines three spatial data types, of the form *set of points*, *set of lines* and *set of surfaces* (Figure 5(a-c)). Operation *Union* (*Minus*) yields only that part of the spatial union (Figure 1(a)) (spatial difference, Figure 1(b)) of two objects whose type matches that of the input objects. Operations *Intersection* and *Common_border* yield specific parts of the spatial intersection (Figure 1(c)) of two objects. *Contour* applies to an element of type *set of surfaces* and returns its boundary of a *set of lines* type (see operation *Boundary*, Figure 1(e)). Assuming the existence of an underlying conventional data model, the following operations apply to data structures that associate spatial with conventional data: *Decompose* decomposes a non-connected spatial object into its connected components. *Fusion* computes the spatial union of all the spatial objects that share identical conventional values, and yields a result similar to that of *Fusion* in Figure 1(a). Finally, *Overlay* computes the spatial intersection of every element of type *set of surfaces* in one data structure with every such element in another, and yields a result similar to that of relation IO, that is, of the result of *Inner Overlay* that is depicted in Figure 1(d).

Similar approaches are the restriction to spatial data management of the spatio-temporal approaches in Güting, Böhlen, Erwig, Jensen, Lorentzos, Schneider and Vazirgiannis, (2000) and Worboys (1994). Spatial data types and operations similar to Güting and Schneider (1995) are also defined in Güting et al. (2000), except that now an infinite spatial representation is considered (Figure 5(d)). A *point* data type is also supported. Finally, the model defined in Worboys (1994) considers only one spatial data type whose elements are collections of points, non-overlapping straight-line segments and non-overlapping triangles. Set operations *Union*, *Difference* and *Intersection* can be applied to spatial objects, obtaining, respectively, their spatial union, difference and intersection. Finally, operation *Boundary* is presented informally.

Spatial data are recorded in relations that satisfy 1NF in Larue, Pastre and Viémont (1993), Roussopoulos, Faloutsos and Sellis (1988), Egenhofer (1994), Scholl and Voisard (1992), Gargano, Nardelli and Talamo (1991), Chen and Zaniolo (2000), and Böhlen, Jensen and Skjellaug (1998). They either define a relational algebra or they extend SQL by functions and relational operations. They are outlined below.

Only one spatial type, GEOMETRY, is supported in Larue et al. (1993). An element of this type is a set of spatial objects, either points, polylines or polygons (Figure 5(a-c)). Functions compute the spatial union, difference and intersection (Figures 1(a-c)). An aggregate function yields the spatial union of a set of spatial objects. Although Roussopoulos et al. (1988) and Egenhofer (1994) do not address spatial data modelling issues, they consider *point*, *line* and *surface* data types and relational SQL extensions. *Solid* and *spatial object* types are also considered in Egenhofer (1994) types. Two functions are also defined in it, called *Complementation* and *Boundary*. The boundary of a line is a set of points. That of a point is the empty set. Types of the form *set of points*, *set of lines* and *set of surfaces* are considered in Scholl and Voisard (1992). Four functions enable computing specific parts of the spatial intersection of two spatial objects of specific data types. Another function returns the element of type *set of lines* that forms the boundary of an object of type *set of surfaces*. A raster-based spatial representation is considered in Gargano et al. (1991). If S is the set of all raster cells (pixels) in a grid, an element of a single data type, GEOMETRY(S), is defined as a set of sets of elements in S (Figure 5(e)). The empty set and non-connected surfaces are valid spatial objects. Operation *G-Compose* merges the spatial objects in some attribute of a relation R, provided that they are in tuples with identical values in some other attribute of R. Operation *G-Decompose* decomposes each spatial object to so many tuples as the number of cells it consists of. A last operation is similar to *G-Compose*, but it also enables applying aggregate functions to non-spatial attributes.

In Chen and Zaniolo (2000), a spatio-temporal SQL extension is proposed, whose restriction to spatial data management enables evaluating an SQL statement for each of the *triangles* a spatial object is composed of. Similarly, in the restriction to spatial data management of the spatio-temporal SQL extension (Böhlen et al., 1998), two types of spatial attributes, *explicit* and *implicit*, are considered, which enable evaluating an SQL statement for each *point* of a spatial object.

Data types of the form *point*, *simple polyline* and *polygon without holes* are considered (Figure 5(a-c)) and *many-sorted algebras* are defined in Güting (1988) and Svensson and Huang (1991). In Güting (1988), a data type AREA is defined as a *polygon without holes* with one additional restriction – that the intersection of two polygons, recorded in the same column, may not be another polygon. Operation *Intersection* enables obtaining *part* of the result of the

spatial intersection of pairs of spatial objects that are recorded in different relations. If the input relations contain only areas, then the operation is called *Overlay* and the result contains only areas. In Svensson and Huang (1991), every operation on 1NF structures is implicitly followed by the application of operation *Unnest*, thus always resulting in a 1NF relation. Operations *Union*, *Difference* and *Intersection* yield, respectively, specific parts of the spatial union, difference and intersection of two spatial objects of the same data type, either *simple polyline* or *polygon without holes*. Operation *Boundary* yields the boundary lines of elements of a polygon type. Further functionality includes the *buffer* area of a spatial object (Figure 1(e)), the *split* of a polygon with respect to a line, the split of a line with respect to a point and the *Voronoi diagram* of a set of points.

Relational approaches with either set-valued or relation-valued attributes are Chan and Zhu (1996); Grumbach, Rigaux and Segoufin (1998); and Kuper, Ramaswamy, Shim and Su (1998). Thus, spatial predicates and functions can be applied to relations, on these attributes. In Chan and Zhu (1996), data types of the form *point* (Figure 5(a)), *simple polyline* (g_4 in Figure 5(b)), *polyline* (Figure 5(b)), *polygon without holes* (g_7 in Figure 5(c)) and *polygon* (Figure 5(c)) are considered. Further, an element of type LINE* is either a point or a polyline, and an element of type REGION* is either a polygon or a LINE*. Sets of elements of these types are also valid types. Many primitive operations are defined. *Fusion* computes the spatial union of a set of spatial objects of any data type (Figure 1(a)). The result is a set of spatial objects of the same data type. Operation *Intersection* computes the spatial intersection of two spatial objects of any type (Figure 1(c)). In the general case, the result is a set of spatial objects of type REGION*. Additional functionality includes *Envelope* (Figure 1(e)), *Buffer* (Figure 1(e)), *Split* (Svensson & Huang, 1991), *Voronoi diagram*, the set of *paths* that link two points in a network of lines, the *holes* of surfaces, and so forth.

Particular cases of nested-relational approaches are the *Constraint-Based Models* proposed in Grumbach et al. (1998) and Kuper et al. (1998). At a *conceptual level of abstraction*, a spatial object is represented by a (possibly infinite) relation with attributes that are interpreted as the dimensions of an n-d space. At a lower level of abstraction, however, such a relation is represented by a finite set of constraints. Spatial union, difference and intersection are achieved by the relational operations *Union*, *Except* and *Intersect*. Operation *Unionnest* applies the relational operation *Union* to all the relations of a relation-valued attribute, provided that these relations belong to tuples whose values for another set of attributes match. The behaviour of operation *Internest* is similar to that of *Unionnest*, except that *Intersection* is now applied instead of *Union*. Further functionality in both of these approaches includes the *Boundary* of surfaces and spatial *Complementation*.

Data structures, which are more complex than those of a nested relation, are used in van Roessel (1994); Scholl and Voisard (1989); and Yeh and de Cambray (1995). Generally, these structures are defined recursively and spatial operations are applied to them. The approach in van Roessel (1994) is close to that of Gargano et al. (1991), discussed earlier. Differences are as follows: *Points* and *infinite subsets of R^2 points* are valid data types. Specifically, two distinct *set of point* spatial data types are defined, one for connected and another for non-connected subsets of R^2. Operations *Fold* and *Unfold*, borrowed from research on temporal databases (Lorentzos & Johnson, 1988), resemble, respectively, *G-Compose* and *G-Decompose* in Gargano et al. (1991). Based on those and the four types of Codd's outer natural join, four types of *Overlay* operations are defined whose functionality is similar to those in ESRI (2003). In Scholl and Voisard (1989), an *elementary region* is defined as a subset of R^2. A *region* is either elementary or a set of elementary regions. Functions to compute the spatial union, difference and intersection of two regions are defined in terms of the respective set operations. A *map* is defined as a relation with at least one attribute of some region data type. Based on predicates, functions and primitive operations, it is shown how representative operations between maps can be achieved (Figure 4). Note however that contrary to Figure 4(f), operation *Overlay* is supported only between maps of the same *cover*. Finally, the characteristics of the restriction to spatial data management of the spatio-temporal model defined in Yeh and de Cambray (1995) match those of Larue et al. (1993) discussed above.

Object-relational models inherit the characteristics of the 1NF model but, at the same time, they incorporate object-oriented capabilities (ISO/IEC, 2002; OpenGIS, 1999; Oracle Corp., 2000; IBM, 2001b,2001a; PostgreSQL, 2001; Vijlbrief & van Oosterom, 1992; Park et al., 1998; Cheng & Gadia, 1994). They consider spatial data types and possibly complex data structures and methods. For the management of various types of complex data, a set of class libraries of the SQL 1999 object types are considered in the SQL Multimedia and Application Packages (SQL/MM) (ISO/IEC, 2002). The part for spatial data management includes a hierarchy of classes that enables the manipulation of 2-d spatial objects. Data type ST_POINT (ST_LINESTRING, ST_POLYGON) consists of vector points (polylines, polygons) (Figure 5(a-c)). Type ST_MULTIPOINT (ST_MULTILINESTRING, ST_MULTIPOLYGON, ST_GEOMCOLLECTION) consists of collections of points (polylines, polygons, spatial objects of any type). An element of type ST_GEOMETRY is an element of any of these types. The *boundary* of an ST_POLYGON is a set of ST_POLYLINES, and the boundary of an ST_POLYLINE is the (possibly empty) set of its end points. The boundary of an ST_POINT element is the empty set. Some of the many methods it considers compute the spatial union, difference and intersection of objects (Figures 1(a-c)). In the general case, the result is a possibly empty element of

type ST_GEOMCOLLECTION. Additional functionality includes the *buffer* of a spatial object (Figure 1(e)). The Simple Feature Specification for SQL (OpenGIS, 1999), proposed by the OpenGIS consortium, is a similar approach. Extensions of commercial DBMS, implementing to some extent the previous standards, are provided in Oracle (Oracle Corp., 2000), Informix (IBM, 2001b) and DB2 (IBM, 2001a). Only one spatial data type of the form *set of spatial objects* is supported in Oracle Corp. (2000). Heterogeneous collections of primitive spatial objects are not supported as spatial objects in IBM (2001a, 2001b). This leads to limitations of the functionality of spatial operations. In IBM (2001b), an aggregate function, *st_dissolve*, enables computing the spatial union of a group of spatial objects whose primitive atomic elements are of the same data type. An Open-Source *Object Relational* DBMS, whose design has been based on Postgres, is PostgreSQL (2001). Its primitive data types are of the form *point*, *infinite straight line*, *line segment*, *rectangle*, *simple polyline* (g_4 in Figure 5(b)) and *polygon without holes*. Many functions and predicates are supported as methods, but their functionality is very primitive. Some example functions return the intersection point of two line segments, the intersection box of two boxes and a *path* defining the boundary of a polygon. Approaches with similar characteristics are Vijlbrief and van Oosterom (1992) and Park, Lee, Lee, Ahn, Lee and Kim (1998). Finally, a query language for spatio-temporal databases is proposed in Cheng and Gadia (1994), in the context of an *object relational* model (ORParaDB). In its restriction to spatial data management, the set R of all spatial objects is a parametric element. Such elements are closed under the set operations *Union*, *Difference*, *Intersection* and *Complementation*. Relational operations *Union*, *Except* and *Intersection* are applied to the parametric elements of tuples.

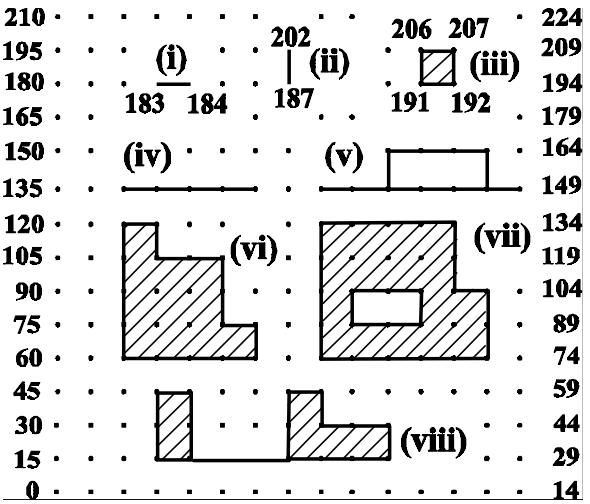
In object-oriented approaches, data structures and methods are combined in the definition of classes. A hierarchy of classes is provided as a general tool for the design of spatial applications. A spatial data structure is incorporated in the model as the data structure of a class. Spatial operations (Figure 1) are incorporated as methods of classes. Application-specific classes, with spatial capabilities, are defined as subclasses of the hierarchy provided by the system. One of these approaches is the restriction to spatial data management of the spatio-temporal approach (Voigtmann, 1997). One hierarchy of classes supports the representation and management of 2-d and 3-d spatial data. *Elementary features* are vector or raster objects. Vector classes represent points (Figure 5(a)), simple polylines (Figure 5(b)) and polygons (Figure 5(c)). Class *Solid* represents 3-d polyhedra. Raster elements are represented by classes *Profile*, *Grid* and *Lattice*. A *Feature* is elementary or a collection of features. A *GeoObject* combines a set of non-spatial properties with a collection of at least one feature. User classes with spatial functionality are defined as subclasses of *GeoObject*. *SpatialObject* is a feature or a *GeoObject*. Functions and predi-

cates for the manipulation of spatial data are defined as methods of classes in this hierarchy. Operations *Touch*, *Overlap* and *Cross* enable obtaining those parts of the spatial intersection of two objects for which a relevant predicate evaluates to true. Their functionality is similar to that of *Spatial Intersection* (Figure 1(c)). The *boundary* of a point is the empty set. The *boundary* of a polyline is the set of its end-points. The *boundary* of a polygon is a set of polylines. Further spatial functionality includes *Buffer* (Figure 1(e)). Many other object-oriented models have also been proposed, some of which are Balovnev, Breunig, and Cremers (1997), Clementini and Di Felice (1993), Ehrich, Lohmann, Neumann and Ramm (1998), Günther and Riekert (1993), and Manola and Orenstein (1986).

Based on the fact that none of the previous approaches satisfies all the requirements for spatial data management that were outlined in the Background section, another spatial data model has been formalised in Viqueira (2003), that satisfies them all. Its fundamental characteristics are outlined briefly below, and relevant examples are shown in Figure 6.

A finite grid of points is initially considered as an underlying discrete 2-d space. A *quantum point*, then, is defined as a set of just one point of the grid (any point in Figure 6). A *pure quantum line* can be *horizontal* (object (i)) or *vertical* (object (ii)). It is the smallest line segment that can be considered on the underlying finite grid. Similarly, a *pure quantum surface* is the smallest surface that can be considered (object (iii)). A *quantum line* is a pure quantum line or a quantum point. A *quantum surface* is a pure quantum surface or a quantum line.

Figure 6. Spatial quanta and spatial data types



Based on spatial quanta, five spatial data types are formalized. Each of them consists of all the R^2 points of the union of *connected* spatial quanta. POINT type consists of all the quantum points (any point in Figure 6). An element of PLINE (*pure line*) type is composed of one or more connected pure quantum lines (objects (i), (ii), (iv) and (v)). An element of PSURFACE (*pure surface*) type is composed of one or more connected pure quantum surfaces (objects (iii), (vi), (vii) and (viii)). Object (vii) is a surface with a *hole* and object (viii) is a hybrid surface. An element of LINE (*line*) type consists of all the elements of either a POINT or PLINE type. Finally, an element of SURFACE (*surface*) type consists of all the elements of either LINE or PSURFACE type. Hence, the model supports directly the *point*, *line* and *surface* data types. All these types are *set-theoretically closed*, that is, lines with missing points and surfaces with missing lines or points are not valid objects. Hybrid surfaces are also valid objects. This property enables the modelling of spatial data of practical interest. The empty set is not a valid spatial object.

The model considers 1NF relations. A relation may have one or more attributes of a spatial data type. Under such an attribute of a SURFACE type it is possible to record spatial data whose geometric representation can be either that of a point or a line or a surface. Codd's relational algebra has been extended by two operations, namely, *Unfold* and *Fold*. Based on all these operations and on some spatial predicates, some more relational algebra operations have been defined that achieve the functionality of the operations in Figure 1. In other words, spatial operations actually reduce to operations on relations. Subsequently, a map can be seen as one or more relations that contain spatial data. All the operations are closed. They apply to any type of spatial data and to any combination of such types. Finally, every operation yields *all* the spatial objects and no part of such an object is missing. For example, spatial intersection yields all the spatial objects in Figure 1(c) case (iv). Overall, the model provides a clear understanding of the management of spatial data.

Although the model has been defined for the management of 2-d spatial data, its extension to n-d data is straightforward. Further, the indication is that it can also be used for the management of continuous spatial changes (Viqueira, Lorentzos, & Brisaboa, 2003). The model also enables the uniform management of any type of data. Indeed, an SQL:1999 extension (Viqueira, 2003), enables the management of conventional, temporal, spatial and spatio-temporal data by the same set of operations. The pseudo-code developed in Viqueira (2003) shows that the model can be implemented. However, a DBMS should provide *data independence*. Due to this, an efficient implementation may consider a vector-based approach at the physical level, in spite of the fact that the model is closer to raster-based approaches.

Future Trends

As already reported, object-relational models inherit characteristics of the 1NF model. At the same time, they incorporate object-oriented capabilities in that spatial data types are defined as abstract data types, which integrate (possibly complex) data structures and methods. Standards for these models are available today (ISO/IEC, 2002; OpenGIS, 1999), but they are restricted to the management of 2-d spatial data. Hence, standards for at least 3-d spatial data have to be developed, preceded by relevant research.

Relevant to continuous changes in space, it is noticed that, so far, there are only informal approaches and implementations. Hence, research work is still required in the formalization of such a model. The estimation is that the same is also true for spatio-temporal data models, despite the many models that have been proposed. It is also noticed that many applications are concerned with the management of spatial networks. Perhaps it is worth investigating this management from within a DBMS.

Finally, the management of spatial data is not yet satisfactorily simple for such end-users as cartographers and others. Hence, the anticipation is that friendly graphical user interfaces will have to be developed on top of DBMS that handle spatial data.

Conclusions

Properties were identified concerning the data types considered, the data structures used and the operations supported by a spatial data model that is intended to support spatial data for cartography, topography, cadastral and relevant applications. A survey of various approaches investigated mainly the satisfaction of these properties. Each approach was also evaluated against these properties.

Acknowledgment

This work has partially been supported by the European Union, TMR Project CHOROCHRONOS (FMRX-CT96-0056).

References

Balovnev, O., Breunig, M., & Cremers, A.B. (1997, July 15-18). From GeoStore to GeoToolKit: The second step. *Proceedings of the 5th International Symposium on Large Spatial Databases (SSD'97)*, Berlin, Germany.

Bentley Systems Inc. (2001). *MicroStation GeoGraphics user's guide*. Version 7.2. Found at *http://docs.bentley.com*

Böhlen, M.H., Jensen, C.S., & Skjellaug, B. (1998, February 27-March 1). Spatio-temporal database support for legacy applications. *Proceedings of the 1998 ACM symposium on Applied Computing (SAC'98)*, Atlanta, Georgia.

Chan, E.P.F., & Zhu, R. (1996). QL/G: A query language for geometric data bases. *Proceedings of the First International Conference on GIS, Urban Regional and Environmental Planning*, April 19-21. Samos, Greece.

Chen, C.X., & Zaniolo, C. (2000, October 9-12). SQL^{ST}: A spatio-temporal data model and query language. *Proceedings of the 19th International Conference on Conceptual Modeling (ER-2000)*, Salt Lake City, Utah.

Cheng, T.S., & Gadia, S.K. (1994, November 29-December 2). A pattern matching language for spatio-temporal databases. *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, Gaithersburg, Maryland.

Clementini, E., & Di Felice, P. (1993, February 14-16). An object calculus for geographic databases. *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice (SAC'93)*, Indianapolis, Indiana.

d'Onofrio, A., & Pourabbas, E. (2001, November 9-10). Formalization of temporal thematic map contents. *Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems (GIS 2001)*. Atlanta, Georgia.

Egenhofer, M.J. (1994). Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, *6*(1), 86-95.

Ehrich, H.D., Lohmann, F., Neumann, K. & Ramm, I. (1988). A database language for scientific map data. In R. Vinken (Ed.), *Construction and display of geosientific maps derived from databases* (pp. 139-152). Hanover.

Environmental Systems Research Institute Inc. (2003). ArcInfo OnLine.Retrieved January 2003 from *www.esri.com/software/arcgis/arcinfo/index.html*

Erwig, M., & Schneider, M. (1997, October 15-18). Partition and conquer. *Proceedings of the International Conference on Spatial Information Theory (COSIT'97)*, Laurel Highlands, Pennsylvania.

Gargano, M., Nardelli, E., & Talamo, M. (1991). Abstract data types for the logical modeling of complex data. *Information Systems*, *16*(6), 565-583.

GRASS. (2002). Geographic resources analysis support system homepage. Retrieved November 2002 from *www.geog.uni-hannover.de/grass/index2.html*

Grumbach, S., Rigaux, P., & Segoufin, L. (1998, June 2-4). The DEDALE system for complex spatial queries. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, Washington.

Günther, O., & Riekert, W. (1993). The design of GODOT: An object-oriented geographic information system. *Data Engineering Bulletin*, *16*(3), 4-9.

Güting, R.H. (1988, March 14-18). Geo-relational algebra: A model and query language for geometric database systems. *Proceedings of the International Conference on Extending Database Technology (EDBT 1988)*. Venice, Italy.

Güting, R.H., & Schneider, M. (1995). Realm-based spatial data types: The ROSE algebra. *VLDB Journal, 4*, 100-143.

Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems, 25*(1), 1-42.

Hadzilacos, T., & Tryfona, N. (1996). Logical data modeling for geographical applications. *International Journal in Geographical Information Systems*, *10*(2), 179-203.

Intergraph Corp. (2002). Working with geomedia proffesional. Available at *http://imgssupport.intergraph.com*

International Business Machines Corp. (1998). *IBM's DB2 Spatial Extender: Managing geo-spatial information within the DBMS.* Technical Report. J.R. Davis.

International Business Machines Corp. (2001). *IBM DB2 Spatial Extender user's guide and reference*. Version 7.

International Business Machines Corp. (2001). *IBM Informix Spatial DataBlade Module user's guide*. Version 8.11.

ISO/IEC JTC 1/SC 32/WG 4: VIE-008. (2002). *SQL Multimedia and Application Packages (SQL/MM) Part 3: Spatial*. M. Ashworth (Ed.), ISO/IEC Committee Draft.

Keigan Systems. (2002). MFWorks OnLine. Retrieved November 2002 from *www.keigansystems.com/tech.html*

Kemp, Z., & Kowalczyk, A. (1994). Incorporating the temporal dimension in a GIS. In M. Worboys (Ed.), *Innovations in GIS* (pp. 89-102). London: Taylor & Francis.

Kuper, G.M., Ramaswamy, S., Shim, K., & Su, J. (1998, November 6-7). A constraint-based spatial extension to SQL. *Proceedings of the 6th International Symposium on Advances in Geographic Information Systems (GIS 1998)*. Washington, DC.

Larue, T., Pastre, D., & Viémont, Y. (1993, June 23-25). Strong integration of spatial domains and operators in a relational database system. *Proceedings of the Third International Symposium on Large Spatial Databases (SSD 1993)*. Singapore.

Lorentzos, N.A., & Johnson, R.G. (1988, May 13-15). TRA: A model for a temporal relational algebra. *Proceedings of the IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems*. Sophia-Antipolis, France.

Lorup, E. J. (2000). IDRISI Tutorial online. Retrieved November 2002 from *www.sbg.ac.at/geo/idrisi/wwwtutor/tuthome.htm*

Manola, F., & Orenstein, J.A. (1986, August 25-28). Toward a general spatial data model for an object-oriented DBMS. *Proceedings of the 12th International Conference on Very Large Data Bases (VLDB 1986)*. Kyoto, Japan.

MapInfo Corp. (2002). *MapInfo professional user's guide.* Version 7.0.

McCoy, J., & Johnston, K. (2001). *Using ArcGis spatial analyst.* Environmental Systems Research Institute, CA.

Open GIS Consortium Inc. (1999). The OpenGIS Implementation Specification: Simple Features Specification for SQL. Revision 1.1, OpenGIS Project Document 99-049. Available at *http://www.opengis.org*

Oracle Corp. (2000). *Oracle Spatial: Users guide and reference.* Release 8.1.7.

Park, K., Lee, J., Lee, K., Ahn, K., Lee, J., & Kim, J. (1998). The development of GEUS: A spatial DBMS tightly integrated with an object-relational database engine. *Proceedings of the Annual Conference of Urban & Regional Information Systems Association (URISA 1998)*, July 18-22. Charlotte, North Carolina.

PostgreSQL Global Development Group. (2001). *PostgreSQL User's Guide*. Version 7.2.

Red Hen Systems Inc. (2001). *MapCalc User's Guide*.

Roessel, J.W. van (1994). An Integrated Point-Attribute Model for Four Types of Areal GIS Features. *Proceedings of the Sixth International Sympo-*

sium on Spatial Data Handling (SDH 1994). Edinburg, Scotland, UK. Vol. 1, 127-144.

Roussopoulos, N., Faloutsos, C., & Sellis, T.K. (1988). An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering, 14*(5), 639-650.

Scholl, M., & Voisard, A. (1989, July 17-18). Thematic map modeling. *Proceedings of the First International Symposium on Large Spatial Databases (SSD 1989)*. Santa Barbara, California.

Scholl, M., & Voisard, A. (1992). Geographic applications: An experince with 02. In F. Bancilhon, C. Delobel, & P.C. Kanellakis (Eds.), *Building an object-oriented database system: The story of 02* (pp. 585-618). San Francisco: Morgan Kaufmann.

Svensson, P. & Huang Z. (1991). Geo-SAL: A query language for spatial data analysis. *Proceedings of the Second International Symposium on Large Spatial Databases (SSD 1991),* August 28-30. Zürich, Switzerland.

Tomlin, C.D. (1990). *Geographic information systems and cartographic modeling.* Englewoods Cliffs, NJ: Prentice Hall.

Tsichritzis, D.C., & Lochovsky, F.H. (1982). *Data models.* Englewoods Cliffs, NJ: Prentice Hall.

Vijlbrief, T., & van Oosterom, P. (1992). The GEO++ system: An extensible GIS. *Proceedings of the Fifth International Symposium on Spatial Data Handling (SDH 1992),* August 3-7, Charleston, South Carolina.

Viqueira, J.R.R. (2003). *Formal extension of the relational model for the management of spatial and spatio-temporal data.* Doctoral Dissertation. Computer Science Department, University of Athens, Corunia. Spain.

Viqueira, J.R.R., Lorentzos, N.A., & Brisaboa, N.R. (2003). Management of continuous spatial changes. *Proceedings of the Ninth Panhellenic Conference on Informatics,* November 21-23. Salonica, Greece.

Voigtmann, A. (1997). *An object-oriented database kernel for spatio-temporal geo-applications.* Doctoral Dissertation. Westfalischen Wilhelms-Universität, Münster, Germany.

Worboys, M.F. (1994). A unified model for spatial and temporal information. *The Computer Journal, 37*(1), 34-36.

Yeh, T.-S., & de Cambray, B. (1995). Modeling highly variable spatio-temporal data. *Proceedings of the Sixth Australasian Database Conference (ADC 1995),* January, Glenelg/Adelaide, South Australia.

Chapter II

Integrating Web Data and Geographic Knowledge into Spatial Databases

Alberto H.F. Laender, Federal University of Minas Gerais, Brazil

Karla A.V. Borges,
Federal University of Minas Gerais, Brazil & PRODABEL, Brazil

Joyce C.P. Carvalho, Federal University of Minas Gerais, Brazil

Claudia B. Medeiros, University of Campinas, Brazil

Altigran S. da Silva, Federal University of Amazonas, Brazil

Clodoveu A. Davis Jr.,
PRODABEL, Brazil & Catholic University of Minas Gerais, Brazil

Abstract

With the phenomenal growth of the World Wide Web, rich data sources on many different subjects have become available online. Some of these sources store daily facts that often involve textual geographic descriptions. These descriptions can be perceived as indirectly georeferenced data – for example, addresses, telephone numbers, zip codes and place names. In this chapter we focus on using the Web as an important source of urban geographic information and propose to enhance urban Geographic Information Systems (GIS) using indirectly georeferenced data extracted

from the Web. We describe an environment that allows the extraction of geospatial data from Web pages, converts them to XML format and uploads the converted data into spatial databases for later use in urban GIS. The effectiveness of our approach is demonstrated by a real urban GIS application that uses street addresses as the basis for integrating data from different Web sources, combining the data with high-resolution imagery.

Introduction

With the popularization of the Web, a huge amount of information has been made available to a large audience (Abiteboul, Buneman, & Suciu, 2000). In some cases, the information available in a Web site, such as pages containing information on restaurants, theaters, movies and shops, concern mostly communities that dwell in a specific neighborhood (Buyukkokten, Cho, Garcia-Molina, Gravano, & Shivakumar, 1999). Furthermore, these sites often provide indirectly georeferenced data such as addresses, telephone numbers, zip codes, place names and other textual geographic descriptions. By *indirectly georeferenced data* we mean spatial data with no associated coordinate (x,y) data. Nevertheless, this kind of data can be converted to positional data, using, for example, address matching functions (Arikawa, Sagara, & Okamura, 2000). Indeed, it can be observed that indirectly georeferenced data abound on the Web. Thus, under this perspective, the Web can be seen as a large geospatial database that often provides up-to-date, regionally relevant information.

In spite of being publicly and readily available, Web data can hardly be properly queried or manipulated as, for instance, data available in traditional and spatial databases (Florescu, Levy, & Mendelzon, 1998). Almost all Web data are unstructured or semistructured (Abiteboul et al., 2000), and cannot be manipulated using traditional database techniques. Web sources are usually constructed as HTML documents in which data of interest (for example, public facilities) is implicit. The structure of these documents can only be detected by visual inspection and is not declared explicitly. In most cases, such data are mixed with markup tags, other strings and in-line code; the structure of most data on the Web is only suggested by presentation features. Besides, when looking for specific information on the Web, users are generally faced with the problem of having to access various distinct and independent Web sites to obtain scattered complementary pieces of information. Typically, this occurs in situations where the required information cannot be found in a single Web source. For example, suppose many different Web sites provide information about restaurants in a city, each site with its own informational content. Someone who wants to get

complete information on all restaurants must access these various sites many times. This leads to a great need for information integration tools and techniques. Methods to allow a correct match of related data contents from several distinct sites are particularly required.

In order to overcome this problem, a possible strategy is to extract data from Web sources to populate databases for further handling; for instance, by using special programs called *wrappers* that identify data of interest and map them to some suitable format (Florescu et al., 1998; Laender, Ribeiro-Neto, Silva, & Teixeira, 2002b). As shown in this chapter, an analogous strategy can be applied to extract geographic context from Web pages to populate spatial databases, thus providing means for supporting new location-based Web services.

Information about cities is currently being accumulated as online digital content both in urban GIS and in local Web pages. Urban GIS stores physical, political and socio-economic information about cities, such as the street network, neighborhood boundaries and demographic information; Web pages store daily life information that can be relevant to local Web users (Hiramatsu & Ishida, 2001). We argue that it is possible to enhance urban GIS using indirectly georeferenced data and information extracted from the Web. The resulting data can be used to build new GIS applications or to update spatial databases.

Data acquisition and updating in urban environments are costly and time-consuming. In developed countries, especially in the United States, there are usually governmental nationwide efforts to generate and to maintain information infrastructure elements, such as address databases. The US Census Bureau, for instance, maintains and distributes at a very low cost its *Topologically Integrated Geographic Encoding and Referencing (TIGER*) files (USCB, 2000), in which street addresses are coded as a set of attributes for segments of street centerlines. The result is a considerable amount of freely available structured geospatial data that can be used to locate points of interest in a very straightforward manner. In emergent countries, on the other hand, the situation is quite the opposite, because of the associated costs and the lack of policies that enforce the updating and integrity of geographic databases.

In this chapter, we present a software environment that allows the extraction of geospatial data from Web pages. The solution also converts extracted data to a suitable format (in our implementation, XML), and uploads it into spatial databases for later use in urban GIS. This is achieved by *geocoding* the data using addresses and other elements in the urban GIS as locators for each data item. Before the geocoding process, data extracted from multiple Web sources must be *integrated*, eliminating duplicate references to the same urban entities or phenomena. There are various techniques that look for identities among objects in order to promote integration of various sources (Cohen, 2000; Garcia-Molina, Quass, Papakonstantinou, Rajaraman, Savig, Ullman, & Widom, 1997;

Guha, Jagadish, Koudas, Srivastava, & Yu, 2002; Tejada, Knoblock, & Minton, 2001). We propose an identification approach based on information retrieval techniques (Baeza-Yates & Ribeiro-Neto, 1999). Our approach focuses specifically on the integration of data from distinct Web sources, urban GIS and high-resolution imagery, using street addresses as the basis for integration.

The remainder of this chapter is organized as follows: First we characterize the background of our research and recent related work. Next we describe our approach to extract address data from Web sources and to integrate data from multiple sources. This is followed by an implemented case study, which integrates Web data and GIS in a real application for the city of Belo Horizonte, Brazil. Furthermore, we describe a new Web site that provides geographic information on restaurants, hotels and other locations of community interest, and that has been created using data extracted from different sources. Finally, we present our conclusions and future work.

Background

Recently, many research efforts have been conducted on the recognition and use of geospatial information from Web sites. Kambayashi, Cheng, and Lee (2001) divide these efforts into three categories. The first category uses maps as a user-friendly interface for the Web, thus making it possible to handle geographic data through usual Web browsers. This approach is called *map-enhanced Web applications* (Hiramatsu & Ishida, 2001; Jones, Purves, Ruas, Sanderson, Sester, van Kreveld, & Weibel, 2002; McCurley, 2001). The second category exploits geographic location information found on Web pages. This information consists of place names, latitude/longitude pairs, postal addresses and so on, used to classify and to index Web pages (Arikawa et al., 2000; Buyukkokten et al., 1999; Ding, Gravano, & Shivakumar, 2000; Jones et al., 2002). The third category focuses on the integration of Web information and geographic knowledge (Kambayashi et al., 2001).

Some approaches belong to more than one category due to their comprehensiveness. Buyukokkten et al. (1999) studied the use of several possible geographic keys for the purpose of assigning site-level geographic context. They analyzed records from WHOIS services, combined them with the hyperlink structure of the Web and were able to infer the geography of the Web at the site level. Ding et al. (2000) extended this approach by introducing techniques for automatically computing the geographical scope of Web sources based on the textual content of the resources, as well as on the geographical distribution of hyperlinks to them. They have implemented a geographically-aware search engine that downloads

and indexes the full contents of 436 online newspapers based in the US. In the Spatial Media Fusion project (Arikawa et al., 2000), map data is used as a basis for integrating various kinds of multimedia data on the Web using *spatial keys*, defined as location data or place names. They use spatial dictionaries to connect and convert spatial data and to make links between various multimedia contents.

The work of Hiramatsu and Ishida (2001) proposes an augmented Web space that integrates GIS facilities and Web services to support everyday life in a metropolitan area. This augmented Web space consists of home pages, hyperlinks and generic links that represent geographical relations between home pages. McCurley (2001) investigated different approaches to discovering geographic context for Web pages, and described a navigational tool for browsing Web resources by geographic proximity. To use the Web as a geographic knowledge base for advanced GIS, Kambayashi et al. (2001) focused on two kinds of factors, namely *geowords* (place names) and *non-geowords*, as found on a Web page. On the basis of these two word domains, they examined co-existence and association rules by applying data mining methods.

The Building Finder Web service is a demonstration application that has been developed using a .NET framework as part of a project on composing Web sources (Thakkar, Knoblock, Ambite, & Shahabi, 2002). It allows a user to input an area of interest, then provides a satellite image of the area overlapped by polygons and graphical texts depicting houses and streets. However, its integration plan is pre-defined and very limited.

These experiments show that discovery and exploitation of geographic information in Web pages is quite feasible, and provides a useful new paradigm for the navigation and retrieval of Web information. *SPatially-aware Information Retrieval on the InTernet(SPIRIT*) (Jones et al., 2002) is a European project that will develop a spatially-aware Web search engine with a geographically-intelligent user interface. The main goal of this project is to create tools and techniques to help people find information that relates to specific geographical locations.

In addition to the initiatives described in this section, there are some commercial search tools that have recently started offering a geographic search capability, where it is possible to locate places of interest in the vicinity of a given address and to navigate on the selected Web sites. Examples includeDotGeo (www.dotgeo.org); InfoSpace (www.infospace.com); Northern Light GeoSearch (www.northernlight.com); WhereOnEarth (www.whereonearth.com); and Yahoo (www.yp.yahoo.com). However, these search tools have been built to locate business Web pages that are already stored in their database.

Since Web sources may overlap or contain replicated data, adequate integration of such sources after the data extraction process contributes to reducing the geocoding effort. Object identification is a problem of central interest in multiple

data sources integration, since it is necessary to identify those objects that should be matched together. Information integration systems like TSIMMIS (Garcia-Molina et al., 1997), Infomaster (Genesereth, Keller, & Duschk, 1997), Information Manifold (Levy, Rajaraman, & Ordille, 1996) and WebView (Arantes, Laender, Golgher, Silva, 2001) manage objects from multiple information sources, but do not offer a general method to determine semantically similar objects. A possible approach to solve this problem in such systems is by using the so-called Skolem function that creates "logical identifiers" for the objects involved (Abiteboul et al., 2000).

Information retrieval (IR) techniques have also been used to find objects that correspond to different descriptions of the same real world object (similar identities). The vector space model (Baeza-Yates & Ribeiro-Neto, 1999) is the most used technique, because it has a good performance when we consider generic object collections. WHIRL (Cohen, 2000) is a database management system that supports similarity joins among database relations with free text attribute values. It uses the vector space model to determine the similarity among text attributes. Active Atlas (Tejada et al., 2001) is an object identification system that also uses the vector space model to establish the mapping between objects of two sources. It learns to tailor mapping rules, through limited user input, to a specific application domain.

There are other approaches that look for similarities among objects (Gravano, Ipeirotis, Jagadish, Koudas, Muthukrishnan, & Srivastava, 2001; Lujn-Mora & Palomar, 2001). Golgher, Silva, Laender and Ribeiro-Neto (2001) describe a strategy that uses a pre-existing repository containing data on a given domain to automatically generate examples for extracting data from Web sources on the same domain. This approach uses a number of heuristics to recognize the intersection among data in the source repository and the target sites. Embley, Jackman and Xu (2001) present a framework for discovering direct matches between sets of source and target attributes in an integration process. The information about potential matches from various facets of metadata is combined to generate and place confidence values on potential attribute matches. Guha et al. (2002) study the problem of integrating XML data sources through correlations realized as join operations. This work shows how edit metrics (Navarro, 2001) and other metrics that quantify distance between trees can be incorporated in a join framework.

The integration approach described in this chapter also adopts similarity techniques taken from the IR field (Carvalho & Silva, 2003). However, while the approaches proposed by Cohen (2000) and Tejada et al. (2001) can be applied only to objects with a flat structure, ours provides a more general framework for object identification that can be applied to more complex objects.

Another problem of great interest in multiple data sources integration is schema matching. It consists of establishing a mapping between object attributes from different sources. A survey on automatic schema matching is presented by Rahm and Bernstein (2001). Our approach does not address the matching schema problem. We assume that the mapping between object attributes from different sources is previously given.

Regarding the recognition and use of geospatial information: Our approach differs from the ones just mentioned because we use Web data as a source for the improvement of geographic knowledge on a city, which includes being able to populate and enrich urban GIS databases using information extracted directly from the Web. Our main motivation is to take advantage of Web data, a rich source of local information, as well as to offer an alternative for collecting and processing new data, since data acquisition costs are a very important issue (Laurini, 2001).

Obtaining Spatial Information from Web Sources

Challenges and Procedure

To create an environment for integrating Web pages to spatial location information, we had to meet several challenges. The first one is to extract indirectly georeferenced data in textual form (such as postal addresses) from the contents of Web pages. We stress that such information, when available, is implicit and occurs as any other ordinary string mixed with HTML markup tags. In the GIS literature, the process of recognizing geographic context is referred to as *geoparsing*, and the process of assigning geographic coordinates is referred to as *geocoding* (McCurley, 2001). This section discusses the efforts to geoparse and to geocode Web pages. The extracted addresses act as keys to the geocoder, which in turn must resort to a previously available data set corresponding to the city's addressing database in order to obtain coordinates.

Another challenge involves the identification of similar objects from different Web sources, since objects can exist in different formats and structures. Establishing ways for transforming the extracted spatial location information – which is found in the form they are generally understood by the public – to the form they are stored in a typical GIS is another challenge to be achieved.

The basic procedure in our approach is to (1) crawl Web sites to collect pages containing data of interest, (2) geoparse the collected pages to extract geo-

graphic indication and relevant data, (3) make the data available in a suitable format (in our case, XML), (4) integrate the Web data that belong to the same domain, (5) geocode the addresses, thus obtaining coordinate data, (6) update the GIS database entity data using these coordinates, and (7) integrate information within the GIS from several geospatial data sources.

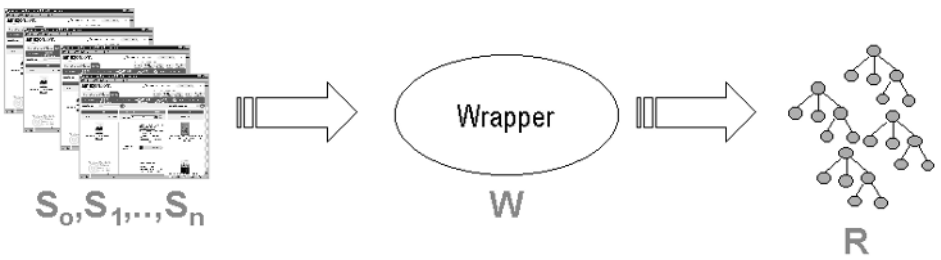
The next section describes how steps 2 and 3 can be accomplished by deploying the Data Extraction By Example (DEByE) approach to automatically extract semistructured data (Laender, Ribeiro-Neto, & Silva, 2002a). This approach is more convenient for our application because it lets the user specify a target structure for the data to be extracted. Furthermore, the user might be interested in only a subset of the information encoded in the page. Moreover, DEByE does not require the user to describe the inherent structure of a whole page. In the following we not only describe steps 2 and 3, but also steps 4 and 5.

The DEByE Tool

DEByE is a tool that has been developed by the UFMG Database Group to generate wrappers for extracting data from Web pages. The problem of generating a wrapper for Web data extraction can be stated as follows: Given a Web page S_0 containing a set of implicit objects, determine a mapping W that populates a data repository R with the objects in S_0. The mapping W must also be capable of recognizing and extracting data from any other page S_i similar to S_0. We use the term *similar* in a very empirical sense, meaning pages provided by a same site or Web service. In this context, a wrapper is a program that executes the mapping W (Figure 1).

DEByE is based on a visual paradigm that allows the user to specify a set of examples for the objects to be extracted. Example objects are taken from a sample page of the same Web source from which other objects will be extracted. By examining the structure of the Web page and the HTML text surrounding the

Figure 1. Wrapper overview



example data, the tool derives an *Object Extraction Pattern* (OEP), a set of regular expressions that includes information on the structure of the objects to be extracted and also the textual context in which the data appear in the Web pages. The OEP is then passed to a general-purpose wrapper that uses it to extract data from new pages in the same Web source, provided that they have structure and contents similar to the sample page (Laender et al., 2002a).

DEByE provides a user interface based on the paradigm of nested tables used, which is simple, intuitive and yet powerful enough to describe hierarchical structures that are very common in data available on the Web. This interface comprises two main windows (see Figures 2 and 3): the upper window (also called the *source window)*, where the sample pages are displayed, and the lower window (also called the *table window)*, which is used to assemble example objects. The user can select pieces of data of interest from the source window and "paste" them on the respective columns of the table window. After specifying the example objects, the user can select the "Generate Wrapper" button to generate the corresponding OEP, which encompasses structural and textual information on the objects present in the sample page. Once generated, this OEP is used by an extractor module that performs the actual data extraction of new objects and then will output them using an XML-based representation. DEByE is also capable of dealing with more complex objects by using a *bottom-up* assembly strategy, explained in Laender et al.(2002a). Figure 2 shows a snapshot of a user session. The source window contains data of a pub (Café com Letras), its name, business hour, address and so on. The corresponding example object appears in the table window. Figure 3 shows the extracted objects presented in HTML format in the upper window. The pub Café com Letras is the

Figure 2. Snapshot of an example specification session with the DEByE interface

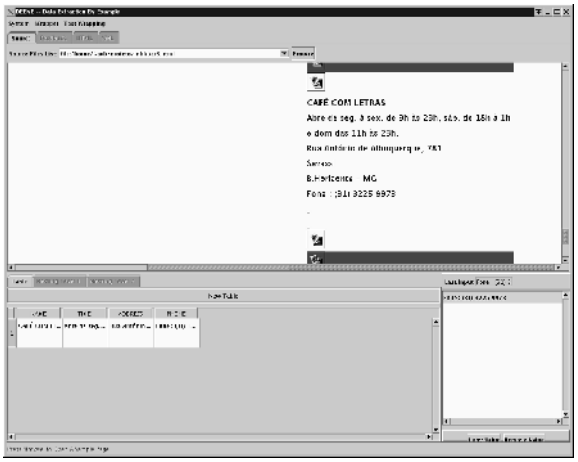
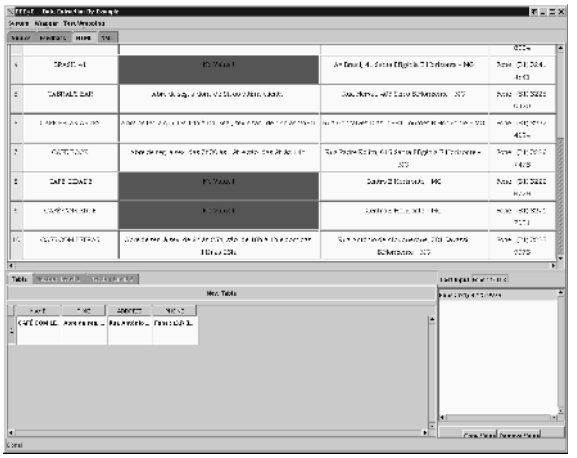


Figure 3. Snaphot of the extracted objects shown in HTML format



last one in this window. Figure 4 shows a fragment of a resulting XML document for the same pub and Figure 5 presents an overview of the DEByE approach.

Data Integration Process

Let us consider without loss of generality the integration of two data sources S_p and S_q. Each source S_i contains a set of objects of type $t_i = (A_{i1}, A_{i2}, ..., A_{in})$, where A_{ij}, $1 \leq j \leq n$, is either an atomic attribute or a list of atomic attributes A defined as $l\{A\}$. For instance, let *Pub_inbh* be a data source extracted from the InBH Web site (www.inbh.com.br) that contains a set of objects of type *source1 = (name, business_hours, address, phone)*, and *Pub_terra* be a data source extracted from the Terra Web site (www.terra.com.br) containing objects of type *source2* = (*name, type, address, phone, price*). Data sources such as these can be naturally represented as XML documents. Figure 4 shows a fragment of an XML document that corresponds to the data source *Pub_inbh*. We also consider that, for each data source S_p and S_q, there is a subset of their attributes, say $S_p' \subseteq S_p$ and $S_q' \subseteq S_q$, respectively, that are semantically equivalent.

Definition 1. Two attribute sets are *semantically equivalent* if they carry the same semantic meaning and if they share similar values from compatible domains.

Figure 4. Example of extracted objects described with XML

```
<Table_XML>
<Table>
........
    <city>
      B.Horizonte - MG
    </city>
        <phone>
        Fone : (31) 3271 7031
    </phone>
</Table>
<Table>
    <name>
      CAFÉ COM LETRAS
    </name>
    <business_hours>
        Abre de seg. à sex. de 9h às 23h, sáb. de 18h à 1h e dom das 11h às 23h
    </business_hours>
    <address>
        Rua Antônio de Albuquerque, 781    Savassi
    </address>
    <city>
        B.Horizonte - MG
    </city>
    <phone>
        Fone : (31) 3225 9973
    </phone>
</Table>
</Table_XML>
```

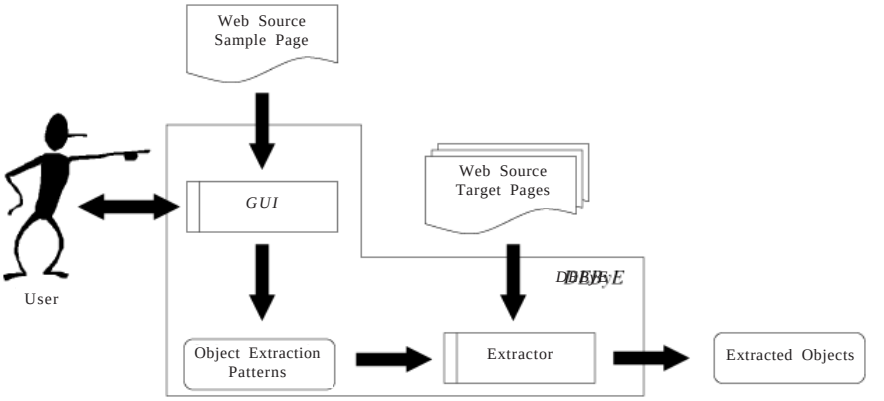
For instance, for the above defined data sources, the attribute sets (name, address, phone) of *Pub_inbh* and (name, address, phone) of *Pub_terra* are semantically equivalent. Thus, to integrate these data sources we need some strategy to establish the identity of the objects involved, possibly considering their semantically equivalent attributes. In this example, these attributes have the same names only for clarity, but in practice this might not be the case.

To find similar identities among objects from two distinct data sources S_p and S_q, we use a similarity-based approach that works like the *join* operation in relational databases. In our approach, a semantic similarity condition takes the place of the equality condition in the join operation.

Definition 2. Two objects $o_p \in S_p$ and $o_q \in S_q$ are *semantically related* if their similarity degree is greater than a parameter r, called the *similarity factor*, that depends on the application domain.

The similarity degree between objects o_p and o_q can be calculated using IR methods. The most-used IR method to calculate similarity degree is the vector space model (Baeza-Yates & Ribeiro-Neto, 1999). In this method objects are

Figure 5. Modules of the DEByE tool and their role in the data extraction process



represented by vectors in a space of $|T|$ dimensions, where T is a set that contains all terms of an object collection C. The similarity degree of two objects p and q is quantified by the cosine of the angle between the vectors that represent them, that is:

$$sim(p,q) = \frac{\vec{p} \bullet \vec{q}}{|\vec{p}| \times |\vec{q}|} \tag{1}$$

$$sim(p,q) = \frac{\sum_{i=1}^{t} w_{i,p} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,p}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{2}$$

where $w_{i,p} \geq 0$ and $w_{i,q} \geq 0$ are term weights that can be calculated by the Term Frequency, Inverse Document Frequency (TF-IDF) weighting scheme (Baeza-Yates & Ribeiro-Neto, 1999). Thus:

$$w_{i,p} = f_{i,p} \cdot idf_i$$

where $f_{i,p}$ is the number of times the term *i* appears in the object *p* and

$$idf = \log \frac{N}{n_i},$$

where *N* is the total number of objects in the collection *C* and n_i is the number of objects in which the term *i* appears. The term $w_{i,q}$ is calculated analogously.

To determine the similarity degree between the extracted objects, we assume that the objects $o_p \in S_p$ and $o_q \in S_q$ can be properly represented as vectors in a multidimensional space. We also assume that the semantically equivalent attribute sets are previously determined, for example, by a schema matching approach (Rahm & Bernstein, 2001).

Let S_p' be an attribute subset of S_p, and S_q' an attribute subset of S_q. We consider that S_p' and S_q' are semantically equivalent, according to Definition 1. For each attribute $A_{pj} \in S_p$', a vector $\vec{o}_{pj}$ is constructed considering a space $\Re^{|T_{pj}|}$, where T_{pj} is the set of all values of A_{pj}. The vector $\vec{o}_{qk}$ is constructed using the same $\Re^{|T_{pj}|}$ space. The similarity degree is given by:

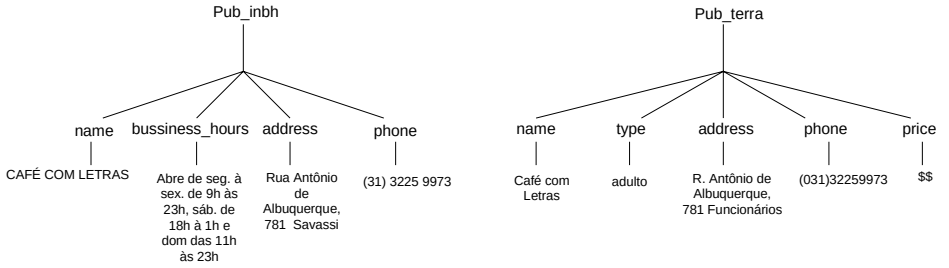
$$sim(o_p, o_q) = \sum_{j \in S_p', k \in S_q'} \theta_p f_{sim}(\vec{o}_{pj}, \vec{o}_{qk})$$

where $f_{sim}(\vec{o}_{pj}, \vec{o}_{qk})$ is calculated according to Equation 2 and $\theta_p \geq 0$ reflects the importance of the corresponding attributes in determining the identity of the objects (Carvalho & Silva, 2003).

To illustrate our approach, let us consider two objects o_1 and o_2 that are shown in Figure 6. We assume that attribute sets *(name, address)* of o_1 and *(name, address)* of o_2 are semantically equivalent. The similarity degree is given by:

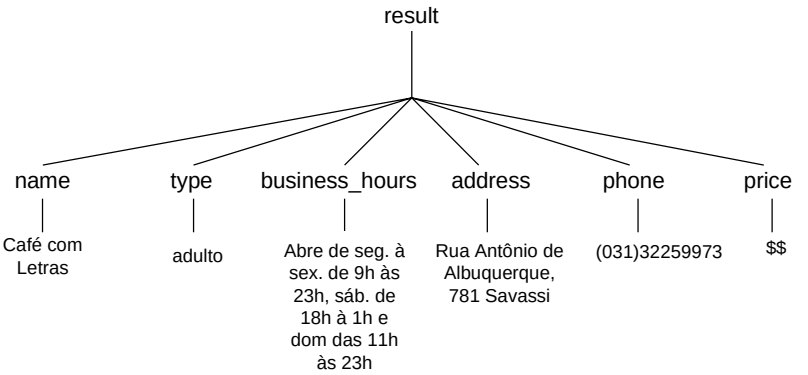
$$sim(o_1, o_2) = \theta_1 \frac{\vec{o}_{11} \bullet \vec{o}_{21}}{|\vec{o}_{11}| \times |\vec{o}_{21}|} + \theta_2 \frac{\vec{o}_{12} \bullet \vec{o}_{22}}{|\vec{o}_{12}| \times |\vec{o}_{22}|}$$

Figure 6. Object examples



where $\vec{o}_{11}$ = {café, letras}, $\vec{o}_{12}$ = {café, letras}, $\vec{o}_{21}$ = {rua, Antonio, Albuquerque, 781, Savassi}, and $\vec{o}_{22}$ = {r., Antonio, Albuquerque, 781, Funcionários} and θ_1 and θ_2 reflect, respectively, the importance of each semantically equivalent attribute in determining the identification of the two objects. For instance, suppose we consider that pub names provide a stronger evidence than pub addresses for the identification of the objects. We can capture this knowledge by making θ_1 greater than θ_2. The object obtained by the integration process is shown in Figure 7. Experimental results presented in Carvalho and Silva (2003) demonstrate that similarity functions based on the vector space model are effective to provide the means to integrate objects from distinct data sources.

Figure 7. Resulting object



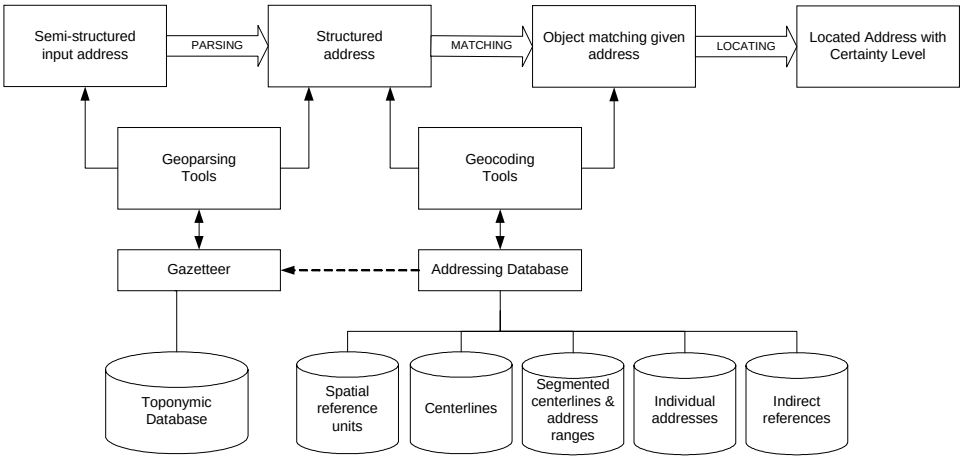
Geocoding Process Using Addresses

One obvious source of geospatial information is the postal address, which is universally used to facilitate the delivery of physical mail to a specific location around the world (Davis Jr., Fonseca, & Borges, 2003). Though recognition of addresses is a fairly studied problem, it is complicated by the fact that formatting standards vary considerably from one country to another (McCurley, 2001). Furthermore, in the same country it is common to have variations for the encoding of the same address. The postal address may or may not have fields for each addressing component, such as thoroughfare type (street, avenue, plaza and so on), thoroughfare name, building number, neighborhood, city, state, country, zip code and possibly others.

Addresses in Web pages are typically segmented into comma-delimited fields or line breaks; sometimes information items, such as the country name, are omitted. This broad variation in abbreviations, punctuation, line breaks and other features used to express the same address makes the parsing process quite complex (Arikawa et al., 2000; McCurley, 2001). Even though our approach takes advantage of user-provided examples to recognize and to extract addresses, it is not easy to separate the fields that compose the address correctly. Therefore, our strategy is to extract the address without necessarily dividing it into fields. Thus, we postpone the parsing problems that normally arise until we get to the geocoding step. The most frequent problems include misspellings, format variations, different names used for the same location and coincidental names for different thoroughfares. The geocoding process includes three phases: (1) treatment of the semi-structured alphanumeric addresses that have been extracted from the Web (*parsing*), (2) establishment of a correspondence between the structured address and the addressing database (*matching* phase), and (3) actual assignment of coordinates to the identified object (*locating* phase) (Figure 8).

Starting from structured addresses, actual geocoding can be performed in several ways, depending on the available addressing information. In order to perform the parsing, matching and locating tasks, the geocoding process needs to have access to a database in which information about the addressing system is stored. There are two basic categories of information in such a database. The first is comprised of the actual addressing infrastructure, with objects such as point-georeferenced individual addresses and street centerlines with address ranges. The second includes any additional information items that can be used to resolve ambiguities or serve as a rough geographic reference in case the address, for any reason, cannot be located in the first category. This includes elements such as all sorts of spatial reference units (area objects that correspond

Figure 8. General geocoding schema



to artificial borders, such as neighborhood limits, districts, ZIP areas, municipal divisions and so on), along with a catalog of what we call "reference places," that is, popularly known spots in a city that can be referenced by name only, that are so easily recognized by the population that their location does not require a formal address. Of course, the addressing database can be rather incomplete, depending on the available data about a given city or location. Our goal is to accommodate this by trying to geocode at the most precise level first and, if that is not possible, successively resorting to less precise geocoding methods until some location can be established.

Since we do not assume any particular structuring in the incoming address, we must be able to determine the structure by analyzing the corresponding string of text. The objective of this process is to create a structured tuple containing every significant piece of information from the original address string. If necessary, addressing elements found in the string are normalized or adjusted before becoming fields in the tuple.

The algorithms that can be used in the parsing of the address are very similar to the ones used in programming languages in order to assess the syntax of a language construct. Initially, the string gets divided into tokens, considering whitespace characters (blanks, commas, points, hyphens and so on) as delimiters. The resulting set of tokens is then analyzed sequentially in an attempt to determine the function of each one. The analysis of each token uses the addressing database in order to establish hypotheses as to the function of each term (token) in the original address. The token functions we look for are: (1)

thoroughfare type: street, avenue, plaza, boulevard and so on, along with their usual abbreviations; (2) thoroughfare name: the name popularly associated with the thoroughfare; thoroughfare names can also have shortened versions, popular nicknames and older names that need to be taken into consideration; (3) street number: number that is usually posted at the door of each building, to indicate a sequence within the thoroughfare; (4) neighborhood: name of any intra-municipal division that is used to identify distinct regions within the city's limits and (5) additional data, such as city name, state and postal code.

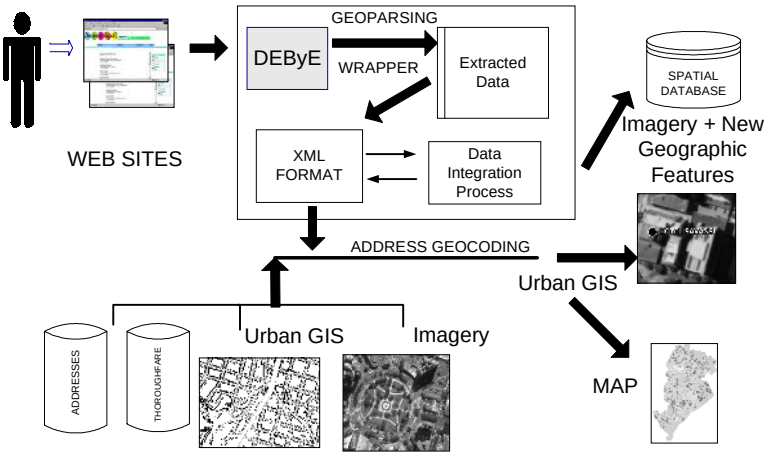
The result of the parsing is a set of fully structured addresses in which there are fields for each of the identified components. Once a postal address has been recognized and parsed it must be geocoded; that is, a coordinate pair must be found for it in the geographic space. The matching function requires a thoroughfare code for each address that has been extracted and parsed. All the problems previously mentioned must be solved in this phase. This is important because there can be redundant names; for example, more than one street can have the same name, possibly in different neighborhoods. Also, there is no guarantee that the thoroughfare names resulting from the parsing process are complete, correct or even existent. However, there can be situations in which the thoroughfare name alone cannot determine a single thoroughfare code, like in the case of homonymous streets. We must then be able to resort to additional information, such as a postal code or a neighborhood name, in order to establish the correct thoroughfare code association. We assume that geocoding is being performed in order to establish a location for real world object or phenomenon – even though this location can be approximate.

After the geocoding task, data objects extracted from Web sites can be stored in a spatial database. These objects represent entities in the real world, like restaurants, hotels and museums. Each object has a set of attributes (such as name, street, phone, URL) and a position in the geographic space, and thus can be handled and displayed by a GIS.

An Application Experience: The Case of Belo Horizonte

We chose to work with a spatial database from the local government of Belo Horizonte, Brazil's fourth largest city, with over two million inhabitants. Belo Horizonte was one of the first Brazilian cities to develop an urban GIS. The development and implementation of this GIS started in 1989 and has proceeded significantly to become what is known as the most complete experience of this kind in Brazil, covering application domains such as education, health, sanitation,

Figure 9. Proposed architecture for the application



urban planning and transportation, among others (Borges & Sahay, 2000). The city's extensive geographic database includes 400,000 individual addresses and 0.40-meter resolution images. These two factors, plus the vast amount of information about the city on the Internet, enabled us to develop a prototype application for Belo Horizonte to validate our proposal. Figure 9 shows the architecture for this prototype application.

The data integrated by our application comes from six different sources: Belo Horizonte's high-resolution imagery (available at www.belohorizonte.com.br), Belo Horizonte urban GIS data, and four distinct Web sites (www.passeiolegal.com.br, www.terra.com.br, www.inbh.com.br, www.comidadibuteco.com.br). The selected sites provide information about hotels, 11 categories of restaurants, pubs, museums and other cultural attractions, consulates, advertising agencies, movie theaters, theaters, clothing stores, libraries and hospitals. A subset of the pages available in each site was collected, totalling 65 pages and 540 urban objects. Some of these pages belong to the same domain. In this case, information on the same object was available from different sources, and thus the collection contained replicated or different data. The integration of these sources, after the data extraction process, contributed to decreasing the geocoding efforts, thus improving quality while enhancing performance.

We next used the DEByE tool to parse the collected pages, extracting the names of points of interest and their addresses (Figure 2). The set of extracted data was then coded in an XML-based format (Figure 4). Before the geocoding process,

extracted data were integrated. After integration, we created a new set of extracted data in XML format, in which each object incorporates all attributes obtained from the integrated versions. It consisted in establishing a mapping between object attributes from different sources. In order to geocode the extracted addresses, we had to transform them into the format in which they are stored in the Belo Horizonte's addressing database. The result is a set of fully structured addresses containing (1) thoroughfare type (street, avenue, plaza, boulevard and so on), (2) thoroughfare name, (3) building number and (4) neighborhood or other types of complementary information. Each postal address recognized and parsed was next geocoded using an address matching function. Naturally, the most precise method is the matching of the extracted address to an object from the Individual Address class in Belo Horizonte's GIS. If a matching individual address was not found, it was possible to resort to the numerically closest address in the same thoroughfare or to match the given address to a reference point (that is, to a member of the Reference Point class in the GIS). Finally, all data extracted from the Web pages in which these addresses were recognized were stored in a spatial database and assigned (x,y) coordinate points, enabling their cartographic visualization.

As a result, we extended the Belo Horizonte GIS database with 28 new tables, one for each urban object category we worked with. The attributes of these tables were: place name, thoroughfare type, thoroughfare name, street number, neighborhood and individual address-assigned code. Our results are summarized in Table 1, which includes columns that, respectively, indicate the total of pages collected, the number of found objects and the number of extracted objects. Subsequent columns show the number of objects for which an exact location has been obtained (*exact match*), the number of those for which the location has been obtained at the numerically closest address (*close match*) on the same street or at an approximate location based on a reference point, and the number of addresses that have not been located (*not found*). As we can see, 90% of the objects were placed at exact locations, 8% were placed at the numerically closest address or at a reference point, and only 2% were not located, mostly because their addresses were incomplete or nonexistent.

Table 1. Experimental results

Site	Pages	Found Objects	Extracted Objects	Exact Match	Close Match	Not Found
www.passeiolegal.com.br	12	122	122	120	---	2
www.terra.com.br	2	52	52	48	4	---
www.inbh.com.br	38	277	277	237	33	7
www.comidadibuteco.com.br	13	89	89	83	6	---

The new geographic features were superimposed on high-resolution imagery in the GIS, thus allowing for many kinds of maps to be produced by integrating points taken from the Web sites to the existing GIS data (Figure 10). This allows, for instance, browsing all facilities within the same area in an integrated manner. Alternatively, we can visualize all extracted data as a new geographic page, providing an opportunity to create a new information space for everyday life. Figure 11 illustrates a map published using Alov Map (Alov, 2003) that combines data extracted from the Web to data taken from the spatial database.

Future Trends

The development of the techniques presented in this chapter opens a number of interesting possibilities for future work on Web-based urban GIS applications. Among them, there are some initiatives already under development by our group. For instance, once we have associated Web pages with geographic information, a natural step would be to query the Web using such information. This would allow searching for pages within some geographic context (for example, on a street, near a beach, and so forth). Implementing such a feature requires aggregating specific data coming from GIS to traditional search engine data structures. Another example of interesting future development can be achieved by transforming implicit (textual) relationships among Web pages into explicit

Figure 10. Integration of urban information from Web pages with geographic databases and high-resolution imagery in a GIS environment

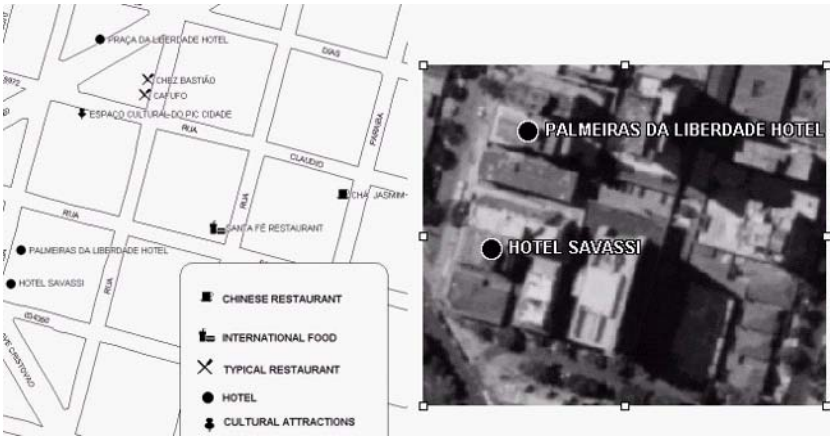
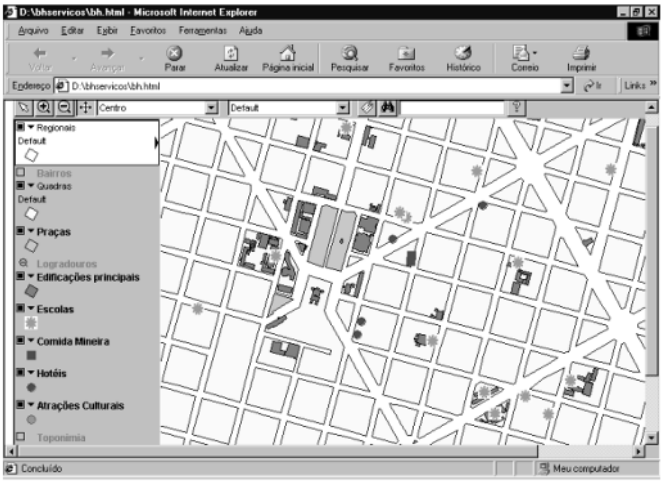


Figure 11. A map showing hotels, typical restaurants and cultural attractions extracted from Web sources, and schools extracted from the spatial database



(GIS) spatial relationships. Then, along with hyperlinks already available on Web pages, it would be possible to automatically generate "geographic" hyperlinks among Web pages, allowing navigation through them using paths not originally identified by page authors.

Additionally, in order to improve the geocoding process, our future work includes the use of ontology-based tools to automatically recognize the indications of the urban geographical context of Web pages, including the recognition of addresses, ZIP codes, reference place names and popular names for urban locations, for which the user would not have to provide examples as shown here. Still another line of work involves proposing a way to assign geographic locations to local Web pages, in which the location of the page's subject is stored within its HTML code. This can provide means to index Web pages according to their geographical location(s). In this approach, coordinates or other forms of geographic reference can be retrieved from Web pages and included in a spatial index. This spatial index can be used to improve the retrieval process; users would be able to provide the usual keywords, along with place references in which their interest lies. Under this framework, place names can be thought of as being a special kind of keyword, associated with a different context, related to the geographic location. This can also have an important effect on the tools and resources that can be used to update spatial databases, using information available in the Web.

Conclusions

In this work we focus on using the Web as an important source of urban geographic information and propose to enhance urban GIS using indirectly georeferenced data extracted from the Web. We describe an environment that allows the extraction of geospatial data from text in Web pages, converts them to XML format, integrates the Web data that belong to the same domain and uploads them into spatial databases for later use in urban GIS. Our proposal is centered on the integration of urban information from local Web pages with geographic databases and high-resolution imagery in a GIS environment. All Web pages that refer to public spaces, including, for instance, restaurants, schools, hospitals, shopping centers and theaters, can be collected, their data extracted and associated with a city's map. Integration with existing GIS data will allow, for instance, urban planners to have a more realistic view of the city, with the actual distribution of its services. The effectiveness of our approach is demonstrated by a real urban GIS application that uses street addresses as the basis for integrating data from different Web sources, combining them with high-resolution imagery. Although still preliminary, the results obtained with our application prototype are encouraging.

Acknowledgments

This work was partially supported by Project GERINDO (MCT/CNPq/CT-INFO grant 552.087/02) and by the first author's individual grant from CNPq. Karla Borges acknowledges the institutional support provided by Prodabel for her doctoral program. Altigran Silva receives grants by CNPq (SiteFix project, MCT-CNPQ-CT-INFO 55.2197/02-5) and PHILIPS MDS Manaus. Claudia Medeiros is partially financed by grants from CNPq and FAPESP, and by the MCT-PRONEX SAI and the CNPq WebMaps projects. Clodoveu Davis' work is partially supported by CNPq, grant 300316/02-0.

References

Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web: From relations to semistructured data and XML*. CA: Morgan Kaufman Publishers.

Alov (2003). Alov map. Retrieved April 2003 from *http://alov.org/index.html*

Arantes, A.R., Laender, A.H.F., Golgher, P.B., & Silva, A.S. (2001). Managing Web data through views. *Proceedings of the Electronic Commerce and Web Technologies Second International Conference, EC-Web 2001*, Munich, Germany.

Arikawa, M., Sagara, T. & Okamura, K. (2000). Spatial Media Fusion Project. *Proceedings of the Kyoto International Conference on Digital Library: Research and Practice,* Kyoto, Japan.

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). *Modern information retrieval.* New York: Addison-Wesley.

Borges, K.A.V. & Sahay, S. (2000). GIS for the public sector: Experiences from the city of Belo Horizonte. *Information Infrastructure and Policy, 6*, 139-155.

Buyukkokten, O., Cho, J., Garcia-Molina, H., Gravano, L. & Shivakumar, N. (1999). Exploiting geographical location information of Web pages. *Proceedings of the Workshop on Web Databases (WebDB 1999) held in conjunction with ACM SIGMOD'99,* Philadelphia, Pennsylvania.

Carvalho, J.C.P. & Silva, A.S. (2003). Finding similar identities among objects from multiple Web sources. *Proceedings of the Fifth International Workshop on Web Information and Data Management, WIDM 2003*, New Orleans, Louisiana.

Cohen, W. (1998). Integration of heterogeneous databases without common domains using queries based on textual similarity. *Proceedings of the International ACM SIGMOD Conference on Management of Data*, Seattle, Washington.

Cohen, W. (1999). Reasoning about textual similarity in a Web-based information access system. *Autonomous Agents and Multi-Agent Systems*, *2*(1), 65-86.

Cohen, W. (2000). Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems, 18*(3), 288-321.

Davis Jr., C.A., Fonseca, F.T, & Borges, K.A.V. (2003). A flexible addressing system for approximate urban geocoding. *Proceedings of the Fifth Brazilian Symposium on GeoInformatics (GeoInfo 2003),* Campos do Jordão (SP), Brazil. Retrieved February 2004 from *www.geoinfo.info*

Ding, J., Gravano, L. & Shivakumar, N. (2000). Computing geographical scopes of Web resources. *Proceedings of the 26th International Conference on Very Large Databases*, Cairo, Egypt.

Embley, D.W., Jackman, D., & Xu, L. (2001). Multifaceted exploitation of metadata for attribute match discovery in information integration. *Proceedings of the Workshop on Information Integration on the Web 2001*, Rio de Janeiro, Brazil.

Florescu, D., Levy, A. & Mendelzon, A. (1998). Database techniques for the World Wide Web: A survey. *SIGMOD Record*, *27*(3), 59-74.

Garcia-Molina, H., Quass, D., Papakonstantinou, Y., Rajaraman, A., Savig, Y., Ullman, J.D. & Widom, J. (1997). The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information System, 8*(2), 117-132.

Genesereth, M.R., Keller, A.M. & Duschka, O.M. (1997). Infomaster: An information integration system. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona.

Golgher, P.B., Silva, A.S., Laender, A.H.F. & Ribeiro-Neto, B. (2001). Bootstrapping for example-based data extration. *Proceedings of the Tenth International Conference on Information and Knowledge Managment*, Atlanta, Georgia.

Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S. & Srivastava, D. (2001). Approximate string joins in a database (almost) for free. *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*. Orlando, FL: Morgan Kaufmann.

Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D. & Yu, T. (2002). Approximate XML joins. *Proceedings of the ACM SIGMOD Conference on Management of Data*, Madison, Wisconsin.

Hiramatsu, K. & Ishida, T. (2001). An augmented Web space for digital cities. *Proceedings of IEEE Symposium on Applications and the Internet, SAINT-2001*, San Diego, California.

Jones, C.B., Purves, R., Ruas, A., Sanderson, M., Sester, M., van Kreveld, M. & Weibel, R. (2002). Spatial information retrieval and geographical ontologies. An overview of the SPIRIT project. *Proceedings of SIGIR 2002: The 25th Annual International ACM SIGIR Conference on Research and Development Information Retrieval*, Tampere, Finland.

Kambayashi, Y., Cheng, K. & Lee, R. (2001). Database approach for improving Web efficiency and enhancing geographic information systems. *Proceedings of International Conference on Internet Information Retrieval, 2001 IRC*, Seoul, Korea.

Laender, A.H.F., Ribeiro-Neto, B.A. & Silva, A.S. (2002a). DEByE: Data extraction by example. *Data and Knowledge Engineering*, *40*(2), 121-154.

Laender, A.H.F., Ribeiro-Neto, B.A., Silva, A.S. & Teixeira, J.S. (2002b). A brief survey of Web data extraction tools. *SIGMOD Record*, *31*(2), 84-93.

Laurini, R. (2001). *Information systems for urban planning: A hypermedia co-operative approach*. New York: Taylor & Francis.

Levy, A.Y., Rajaraman, A. & Ordille, J.J. (1996). Query answering algorithms for information agents. *Proceedings of the AAAI 13th National Conference on Artificial Intelligence*, Portland, Oregon.

Lujn-Mora, S. & Palomar, M. (2001). Reducing inconsistency in integrating data from different sources. In M. Adiba, C. Collet & B.P. Desai (Eds.), *Proceedings of the International Database Engineering and Applications Symposium (IDEAS 2001), IEEE Computer Society*, Grenoble, France.

Makinouchi, A. (1977). A consideration on normal form of not-necessarily-normalized relation in the relational data model. *Proceedings of the Third International Conference on Very Large Database*, Tokyo, Japan.

McCurley, K.S. (2001). Geospatial mapping and navigation of the Web. *Proceedings of the 10th International World Wide Web*, Hong Kong, China.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, *33*(1), 31-88.

Rahm, E. & Bernstein, P.A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, *10*, 334-350.

Tejada, S., Knoblock, C.A. & Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, *26*(8), 607-635.

Thakkar, S., Knoblock, C.A., Ambite, J.L. & Shahabi, C. (2002). Dynamically composing Web services from on-line sources. *Proceedings of the 18th National Conference on Artificial Intelligence Workshop on Intelligent Service Integration*, Edmonton, Canada.

United States Census Bureau (2000). The 108th CD Census 2000 TIGER/Line Files Technical Documentation. Retrieved Febrary 2004 from *www.census.gov/geo/www/tiger/tgrcd108/tgr108cd.pdf*

Section II

Indexing Techniques

Chapter III

Object-Relational Spatial Indexing

Hans-Peter Kriegel, University of Munich, Germany

Martin Pfeifle, University of Munich, Germany

Marco Pötke, sd&m AG, Germany

Thomas Seidl, RWTH Aachen University, Germany

Jost Enderle, RWTH Aachen University, Germany

Abstract

In order to generate efficient execution plans for queries comprising spatial data types and predicates, the database system has to be equipped with appropriate index structures, query processing methods and optimization rules. Although available extensible indexing frameworks provide a gateway for seamless integration of spatial access methods into the standard process of query optimization and execution, they do not facilitate the actual implementation of the spatial access method. An internal enhancement of the database kernel is usually not an option for database developers. The embedding of a custom, block-oriented index structure into concurrency control, recovery services and buffer management would cause extensive implementation efforts and maintenance

cost, at the risk of weakening the reliability of the entire system. The server stability can be preserved by delegating index operations to an external process, but this approach induces severe performance bottlenecks due to context switches and inter-process communication. Therefore, we present the paradigm of object-relational spatial access methods that perfectly fits to the common relational data model, and is highly compatible with the extensible indexing frameworks of existing object-relational database systems, allowing the user to define application-specific access methods.

Introduction

Users of database systems want to manage data of very different types, depending on the particular application area. While office applications, for example, mainly perform simple access and update operations on records of simple data types, spatial data usually have a complex structure and demand specialized operations. It is not a choice for vendors of database management systems to provide data types and management functions for each conceivable domain. So the design of *extensible architectures* allowing users to adapt systems to their special needs represents an important area in database research.

Traditional relational database systems support very limited extensibility. All data have to be mapped on rows of flat tables consisting of attributes with such simple types as numbers, character strings or dates. For the retrieval and manipulation of data, there exist only generic operations for selecting, inserting, updating and deleting (parts of) rows within tables. Data of more complex types cannot be stored directly as a unit in the database but have to be split across several tables. To restore the data from the system, complex queries with many joins have to be performed. Alternatively, the data can be coded within a large object that prevents direct access to single components of the data using the database language. Operations on complex types have to be implemented within the application and cannot be used within the database language directly.

Object-oriented database management systems (OODBMS) seem to provide solutions for most of the cited problems of relational databases. An OODBMS has an extensible type system that allows the user to define new data types (by the nested application of type constructors) together with corresponding operations. The resulting *object types* then describe the structure as well as the behavior of the objects based on this type. Furthermore, subtypes (inheriting the properties of their supertypes) can be derived of existing object types.

To make object-oriented and extensibility features also available in relational systems, database researchers and manufacturers proposed and implemented corresponding enhancements for the relational model during the past few years. The resulting *object-relational database management systems* (ORDBMS) retain all features of the relational model, especially the storage of data within tables and the powerful declarative query processing with the relational database language SQL. Beyond that, the object-relational data model introduces abstract data types into relational database servers. Thereby, object-relational database systems may be used as a natural basis to design an integrated user-defined database solution. The ORDBMS already support major aspects of the declarative embedding of user-defined data types and predicates. In order to achieve a seamless integration of custom object types and predicates within the declarative *data definition language* (DDL) and *data manipulation language* (DML), ORDBMS provide the database developer with extensibility interfaces. They enable the declarative embedding of abstract data types within the built-in optimizer and query processor.

In the following, we categorize possible approaches to incorporate third-party spatial indexing structures into a relational database system by what we call *Relational Indexing*. After an introduction to ORDBMS and their extensible indexing facilities, we discuss three different implementations of spatial access methods, including the relational approach, and introduce basic concepts of object-relational spatial access methods. Then the design of the corresponding update and query operations is investigated. Afterwards, we identify two generic schemes for modeling object-relational spatial access methods, which are discussed with respect to their support of concurrent transactions and recovery. Finally, some concluding remarks are given.

Indexing Interfaces in Object-Relational Databases

Extensible frameworks are available for most object-relational database systems, including Oracle (Oracle, 1999a; Srinivasan, Murthy, Sundara, Agarwal, & DeFazio, 2000), IBM DB2 (IBM, 1999; Chen, Chow, Fuh, Grandbois, Jou, Mattos, Tran, Wang, 1999) or Informix IDS/UDO (Informix, 1998; Bliujute, Saltenis, Slivinskas, & Jensen, 1999). Custom server components using these built-in services are called *data cartridges*, *database extenders* and *data blades* in Oracle, DB2 and Informix, respectively. The open-source ORDBMS PostgreSQL (PostgreSQL, 2002; Stonebraker & Kemnitz, 1991) has the same

Figure 1. Object-relational DDL statements for polygon data

```
// Type declaration

CREATE TYPE POINT AS OBJECT (x NUMBER, y NUMBER);
CREATE TYPE POINT_TABLE AS TABLE OF POINT;
CREATE TYPE POLYGON AS OBJECT (
  points POINT_TABLE,
  MEMBER FUNCTION intersects (p POLYGON) RETURN BOOLEAN
);

// Type implementation
// …

// Functional predicate binding

CREATE OPERATOR INTERSECTS (a POLYGON, b POLYGON)
RETURN BOOLEAN
BEGIN RETURN a.intersects(b); END;

// Table definition

CREATE TABLE polygons (
  id NUMBER PRIMARY KEY,
  geom POLYGON
);
```

roots as the commercial database system of Informix and also provides similar extensibility features.

Declarative Integration

As an example, we create an object type *POLYGON* to encapsulate the data and semantics of two-dimensional polygons. Instances of this spatial object type are stored as elements of relational tuples. Figure 1 depicts some of the required object-relational DDL statements in pseudo SQL, thus abstracting from technical details that depend on the chosen product. By using the functional binding of the user-defined predicate *INTERSECTS*, object-relational queries can be expressed in the usual declarative fashion (Figure 2). Provided only with a functional implementation that evaluates the INTERSECTS predicate in a row-by-row manner, the built-in optimizer has to include a full-table scan into the execution plan to perform the spatial selection. In consequence, the resulting performance will be very poor for highly selective query regions. As a solution, the extensibility services of the ORDBMS offer a conceptual framework to supplement the functional evaluation of user-defined predicates with index-based lookups.

Figure 2. Object-relational region query on polygon data for a region query_region

```
// Region query

SELECT id FROM polygons
WHERE INTERSECTS(geom, :query_region);
```

Extensible Indexing

An important requirement for spatial applications is the availability of user-defined access methods. Extensible indexing frameworks proposed by Stonebraker (1986) enable developers to register custom secondary access methods at the database server in addition to the built-in index structures. An object-relational indextype encapsulates stored functions for creating and dropping a custom index and for opening and closing index scans. The row-based processing of selections and update operations follows the iterator pattern (Gamma, Helm, Johnson, & Vlissides, 1995). Thereby, the indextype complements the functional implementation of user-defined predicates. Figure 3 shows some basic indextype methods invoked by extensible indexing frameworks. Additional functions exist to support query optimization, custom joins and user-defined aggregates. Assuming that we have encapsulated a spatial access method for two-dimensional polygons within the custom indextype SpatialIndex, we may create an index polygons_idx on the geom attribute of the polygons table by submitting the usual

Figure 3. Methods for extensible index definition and manipulation

Function	Task
index_create(), index_drop()	Create and drop a custom index.
index_open(), index_close()	Open and close a custom index.
index_fetch()	Fetch the next record from the index that meets the query predicate.
index_insert(), index_delete(), index_update()	Add, delete, and update a record of the index.

Figure 4. Creation of a custom index on polygon data

```
// Index creation

CREATE INDEX polygons_idx ON polygons(geom)
INDEXTYPE IS SpatialIndex;
```

DDL statement (Figure 4). If the optimizer decides to include this custom index into the execution plan for a declarative DML statement, the appropriate indextype functions are called by the built-in query processor of the database server. Thereby, the maintenance and access of a custom index structure is completely hidden from the user, and the desired data independence is achieved. Furthermore, the framework guarantees any redundant index data to remain consistent with the user data.

Talking to the Optimizer

Query optimization is the process of choosing the most efficient way to execute a declarative DML statement (Yu & Meng, 1998). Object-relational database systems typically support rule-based and cost-based query optimization. The extensible indexing framework comprises interfaces to tell the built-in optimizer about the characteristics of a custom indextype (Stonebraker & Brown, 1999). Figure 5 shows some cost-based functions that can be implemented to provide the optimizer with feedback on the expected index behavior. The computation of custom statistics is triggered by the usual administrative SQL statements. With a cost model registered at the built-in optimizer framework, the cost-based optimizer is able to rank the potential usage of a custom access method among alternative access paths. Thus, the system supports the generation of efficient execution plans for queries comprising user-defined predicates. This approach

Figure 5. Methods for extensible query optimization

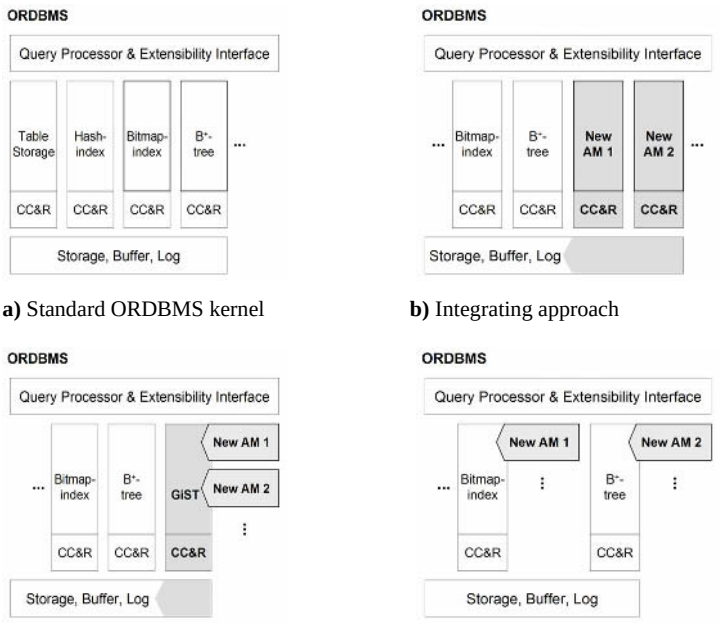
Function	Task
stats_collect(), stats_delete()	Collect and delete persistent statistics on the custom index.
predicate_sel()	Estimate the selectivity of a user-defined predicate by using the persistent statistics.
index_cpu_cost(), index_io_cost()	Estimate the CPU and I/O cost required to evaluate a user-defined predicate on the custom index.

preserves the declarative paradigm of SQL, as it requires no manual query rewriting.

Implementation of SPATIAL Access Methods

In the previous section, we outlined how object-relational database systems support the logical embedding of spatial indextypes into the declarative query language and into the optimizer framework. The required high-level interfaces can be found in any commercial ORDBMS and are continuously improved and extended by the database vendors. Whereas the embedding of a custom indextype is therefore well supported, its actual implementation within a fully-fledged database kernel remains an open problem. In the following, we discuss three basic approaches to implement the low-level functionality of a spatial access method: the *integrating*, the *generic* and the *relational approach* (Figure 6).

Figure 6. Approaches to implement custom access methods



a) Standard ORDBMS kernel

b) Integrating approach

c) Generic approach

d) Relational approach

The Integrating Approach

By following the integrating approach, a new spatial access method (AM) is hard-wired into the kernel of an existing database system (Figure 6b). In consequence, the required support of ACID properties, including concurrency control and recovery services (CC&R) has to be implemented from scratch and linked to the corresponding built-in components. Furthermore, a custom gateway to the built-in storage, buffer and log managers has to be provided by the developer of the new AM. Most standard primary and secondary storage structures are hard-wired within the database kernel, including plain table storage, hash indexes, bitmap indexes and B+-trees. Only a few non-standard access methods have been implemented into commercial systems in the same way, including the *R-Link-tree* in Informix IDS/UDO for spatially extended objects (Informix, 1999) and the *UB-tree* in TransBase/HC for multidimensional point databases (Ramsak, Markl, Fenk, Zirkel, Elhardt, & Bayer, 2000). The integrating approach comprises the *Extending Approach* and the *Enhancing Approach*.

The Extending Approach

Adding a completely new access method to a database system is quite expensive, because in addition to the actual index algorithms, all the concurrency, recovery and page management has to be implemented (R-Link-Tree, Bitmaps, External Memory Interval Tree). Carey, DeWitt, Graefe, Haight, Richardson, Schuh, Shekita and Vandenberg (1990) guess that the actual index algorithms only comprise about 30% of the overall code for the access method, while the other 70% are needed to integrate the access method properly into the database system.

Several approaches to facilitate the implementation of access methods and other components for special-purpose database systems have been proposed in database research. Under the assumption that it is practically impossible to implement a database management system capable to fulfill the demands of arbitrary application domains, tools and generic database components have been developed that should enable domain specialists to implement their required database system with minimum effort, whether a document management system or a geographical information system. The resulting systems might have completely different characteristics; for example, different query languages, access methods, storage management and transaction mechanisms.

The database system toolkit EXODUS (Carey et al., 1990; Carey M, DeWitt, Frank, Graefe, Richardson, Shekita, & Muralikrishna, 1991) provides a storage

Figure 7: B-tree routine next_node for different data types

```
next_node(n:node; key:integer);
    …
    while n.son[index].value < key
        increment(index);
    next := fetch(n.son[index].address);
    …
end;
```

a) next_node routine handling keys of type integer

```
next_node(n:node; key:real);
    …
    while n.son[index].value < key
        increment(index);
    next := fetch(n.son[index].address);
    …
end;
```

b) next_node routine handling keys of type real

```
template <keytype>
next_node(n:node; key:keytype);
    …
    while lessthan(n.son[index].value, key)
        increment(index);
    next := fetch(n.son[index].address);
    …
end;
```

c) next_node routine handling arbitrary data types

manager for objects, a library of access methods, a library of operator methods for the data model to generate, a rule-based generator for query optimizers, tools for constructing query languages and a persistent programming language for the definition of new access methods and query operators. Using these "tools," the domain specialist can build an application-specific database system with suitable access methods. Another system of this category is the database generator GENESIS (Batory, Barnett, Garza, Smith, Tsukuda, Twichell, Wise, 1990), which provides a set of composable storage and indexing primitives and a "database system compiler" for assembling an appropriate storage manager from a specification.

Unfortunately, these universally extensible database systems have essentially proven to be hard to use in practice. Though these systems support the user with the implementation of single database components, a lot of expertise is required to use them. In some ways, they also are a bit too inflexible and incomplete to implement a fully-fledged database management system. So in practice, few databases have been implemented using such toolkits or generators.

The Enhancing Approach

In contrast to the extending approach, the enhancing approach is much cheaper, since already existing access methods are augmented to support a broader range of data. The code of the access methods to be enhanced has to be adapted so it gets independent of the indexed data type. As an example, Figure 7a depicts the pseudocode of a B-tree index routine *next_node* that determines (for a given node and a key value) the node that has to be visited next within a B-tree traversal. This routine only works on key values of type *integer*, that is, an index using this routine can only be created on columns of type *integer*. In order to create B-tree indexes also on columns of type *real*, the type of the key parameter has to be changed accordingly (Figure 7b). In general, to support arbitrary (ordered) types, the B-tree code has to be modified so it can handle key parameters of any type. Figure 7c depicts a type-independent version of the *next_node* routine. Here, the key type is not determined, but it has to be instantiated when applying the index. The function *lessthan* has the same functionality as the operator '<' for built-in types. If the user defines a new type and wants to use the enhanced B-tree index for columns of this type, the user has to provide a corresponding *lessthan* function that can handle values of the new

Figure 8. User-defined data type FracNum

```
CREATE TYPE FracNum (num INTEGER; denom INTEGER)
```

a) Data type for fraction numbers

```
CREATE FUNCTION lessthan (f1 FracNum, f2 FracNum)
RETURN BOOLEAN
LANGUAGE SQL DETERMINISTIC
BEGIN
    RETURN (f1.num/f1.denom) < (f2.num/f2.denom);
END
```

b) Function lessthan for comparing fraction numbers

type. Alternatively, the built-in operator '<' could be overloaded, if the database system used supports this. As a further example, if the user defines a new type FracNum for the storage of fraction numbers (consisting of numerator and denominator) in the database system (Figure 8a), the user has to implement a special version of the function *lessthan* that takes two fraction numbers as parameters (Figure 8b). Whenever the routine *next_node* is called with a key parameter of type *FracNum*, the newly defined version of *lessthan* is used.

In general, to enhance (generalize) an access method in this way, all type-specific operations within the code of the access method have to be identified and isolated so the user can provide overloaded versions of these operations for his user-defined types. It is necessary to note that not every access method is appropriate for every data type. B-trees, for example, only can be used for types with a linear ordering. In contrast, R-trees are designed to support access to spatially extended and to multi-dimensional data. Depending on the access method and the predicates to be supported by the index, the user has to implement corresponding operations for new data types. To use an enhanced B-tree index, the user must provide implementations of the usual comparison operators '<', '≤', '>', '≥', and '=' for a new data type, whereas an R-tree index requires spatial operations like 'overlaps', 'contains', 'within', or 'equals'.

A further possibility to enhance existing access methods is to implement functional indexes that give quick access to the results of a function defined on the attributes of a table. The type of the function value has to be supported by the enhanced index.

In conclusion, we identify the following properties of the integrating approach:

Implementation: The implementation of a new spatial AM becomes very sophisticated and tedious if writing transactions have to be supported (Brown, 2001). In addition, the code maintenance is a very complex task, as new kernel functionality has to be implemented for any built-in access method. Moreover, the tight integration within the existing kernel source produces a highly platform-dependent solution tailor-made for a specific ORDBMS. The enhancement of pre-existing access methods to support user-defined data types and functional indexes is a straightforward task, but does not really augment the functionality of the database server (in the sense of having new ways for query processing).

- **Performance:** The integrating approach potentially delivers the maximal possible performance, if the access method is implemented in a closed environment, and the number of context switches to other components of the database kernel is minimized.
- **Availability:** The implementation requires low-level access to most kernel components. If the target ORDBMS is not distributed as open-source, the affected code and documentation will not be accessible to external database developers.

To sum up, the integrating approach is the method of choice only for a few, well-selected access methods serving the requirements of general database applications. It is not feasible for the implementation of rather specialized access methods.

The Generic Approach

To overcome restrictions of the integrating method, Hellerstein, Naughton and Pfeffer (1995) proposed a generic approach to implement new access methods in an ORDBMS. Their *Generalized Search Tree* (*GiST*) has to be built only once into an existing database kernel. The GiST serves as a high-level framework to plug in block-based tree structures with full ACID support (Figure 6c).

As in the previous approaches, the database implementor has to integrate the extensibility framework into the database server regarding all tedious tasks like concurrency and recovery. Once implemented, a domain specialist can use the GiST framework to derive new index types for particular applications. In contrast to database toolkits or generators, the index implementor does not have to stop the database server and recompile it every time an index type is added. It is just necessary to implement (overload) a number of predefined functions that define the behavior of keys in the tree. This is quite similar to the enhanced index approach at first glance. However, while enhanced indexes only support new data types for already existing index structures, it is possible to support completely new query predicates with the GiST framework. Both B-trees and R-trees are derivable from the GiST framework, for example. Such derived index types may not be as performant as directly integrated ones, but they require much less effort to realize.

Many extensions to the GiST framework have been presented, including generic support for concurrency and recovery (Kornacker, Mohan & Hellerstein, 1997), and additional interfaces for nearest-neighbor search, ranking, aggregation and selectivity estimation (Aoki, 1998). In detail, the GiST approach has the following characteristics:

- **Implementation:** Whereas the implementation of block-based spatial access methods on top of the GiST framework can be done rather easily, the intruding integration of the framework itself remains a very complex task. As an advantage, an access method developed for GiST can basically be employed on any ORDBMS that supports this framework. In contrast to the generic GiST implementation, the specialized functionality of a new access method is therefore platform independent.
- **Performance:** Although the framework induces some overhead, we can still achieve high performance for GiST-based access methods. Kornacker

(1999) has shown that they may even outperform built-in index structures by minimizing calls to user-defined functions.

- **Availability:** Due to its complex implementation, the GiST framework is not generally available in present-day systems. To our best knowledge, it has only been implemented in the open-source system PostgreSQL, but without concurrent access and write-ahead logging of updates for derived indexes. It is an open question whether and when a comparable functionality with industrial-strength implementation will be a standard component of major commercial ORDBMS.

The GiST concept basically delivers the desired properties to implement spatial access methods. It delegates crucial parts of the implementation to database vendors. Unfortunately, its full functionality is not available in any major commercial database system at present. Furthermore, database extensions should generically support many database platforms. Thus, the GiST concept would have to be implemented not only for one, but for all major ORDBMS.

The Relational Approach

A natural way to avoid the above obstacles is to map the spatial index structure to a relational schema organized by built-in access methods (Figure 6d). Such *relational access methods* are designed to operate on top of a relational query language. They require no extension or modification of the database kernel; thus, any off-the-shelf ORDBMS can be employed as it is. We identify the following advantages for the relational approach:

- **Implementation:** As no internal modification or extension to the database server is required, a relational access method can be implemented and maintained with less effort. Substantial parts of the custom access semantics may be expressed by using the declarative DML. Thereby, the implementation exploits the existing functionality of the underlying ORDBMS rather than duplicating basic database services as done in the integrating and generic approaches. Moreover, if we use a standardized DDL and DML like SQL:1999 (ANSI, 1999) to implement the low-level interface of our access method, the resulting code will be platform independent.
- **Performance:** The major challenge in designing a relational access method is to achieve both high usability and performance. The capability and efficiency of the relational approach was proven for interval data (Kriegel, Pötke, & Seidl, 2000; Kriegel, Pfeifle, Pötke, & Seidl, 2002) and 2D/3D spatial data (Kriegel, Pötke, & Seidl, 2001; Kriegel, Müller, Pötke, & Seidl, 2001).

- **Availability:** By design, a relational access method is supported by any relational database system. It requires the same functionality as an ordinary database user or a relational database application.

By following the relational approach to implement spatial access methods, we obtain a natural distinction between the basic services of all-purpose database systems and specialized, application-specific extensions. By restricting database access to the common SQL interface, spatial access methods and query procedures are well-defined on top of the core server components. In addition, a relational access method immediately benefits from any improvement of the ORDBMS infrastructure.

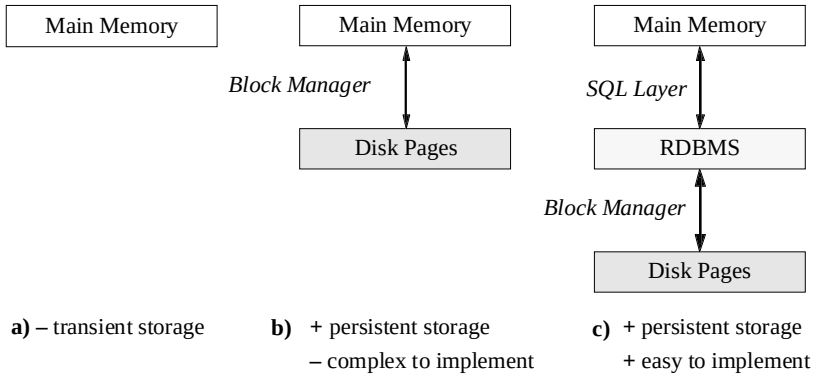
Basics of Object-Relational Spatial Indexing

The basic idea of relational access methods relies on the exploitation of the built-in functionality of existing database systems. Rather than extending any internal component of the database kernel, a relational access method just uses the native data definition and data manipulation language to process updates and queries on abstract data types. Without loss of generality, we assume that the underlying database system implements the standardized *Structured Query Language* SQL-92 (ANSI, 1992) with common object-relational enhancements in the sense of SQL:1999 (ANSI, 1999), including object types and collections.

Paradigms of Access Methods

A relational access method delegates the management of persistent data to an underlying relational database system by strictly implementing the index definition and manipulation on top of an SQL interface. Thereby, the SQL layer of the ORDBMS is employed as a *virtual machine*, managing persistent data. Its robust and powerful abstraction from block-based secondary storage to the object-relational model can then be fully exploited. This concept also perfectly supports database appliances, that is, dedicated database machines running the ORDBMS as a specialized operating system (Keim & Prawirohardjo, 1992; Oracle, 2000). We add the class of relational access methods as a third paradigm to the known paradigms of access methods for database management systems:

Figure 9. Paradigms and characteristics of access methods: a) main memory access methods, b) block-oriented access methods, c) relational access methods



- **Main Memory Access Methods:** Typical applications of these techniques can be found in main memory databases (DeWitt, Katz, Olken, Shapiro, Stonebraker, Wood, 1985; Garcia-Molina & Salem, 1992) and in the field of computational geometry (Preparata & Shamos, 1993). A popular example taken from the latter is the binary *Interval Tree* (Edelsbrunner, 1983). It serves as a basic data structure for plane-sweep algorithms; for example, to process intersection joins on rectangle sets. Main memory structures are not qualified for indexing persistent data, as they disregard the block-oriented access to secondary storage. (See Figure 9a.)
- **Block Oriented Access Methods:** These structures are designed to efficiently support the block-oriented I/O from and to external storage and are well suited to manage large amounts of persistent data. The External Memory Interval Tree (Arge & Vitter, 1996) is an example for the optimal externalization of a main memory access method. Its analytic optimality is achieved by adapting the fanout of the Interval Tree to the disk block size. In the absence of a generalized search tree framework (Hellerstein, Naughton, & Pfeffer, 1995), the implementation of such specialized storage structures into existing database systems, along with custom concurrency control and recovery services, is very complex. Furthermore, it requires intrusive modifications of the database kernel (Ramsak, Markl, Fenk, Zirkel, Elhardt, & Bayer, 2000). (See Figure 9b.)
- **Relational Access Methods:** In contrast, relational access methods including the Relational Interval Tree (Kriegel et al., 2000) are designed to

operate on relations rather than on dedicated disk blocks. The persistent storage and block-oriented management of the relations is delegated to the underlying database server. Therefore, the robust functionality of the database kernel, including concurrent transactions and recovery, can potentially be reused. A primary clustering index can be achieved by also delegating the clustering to the ORDBMS. For this, the payload data has to be included into the index relations, and the clustering has to be enabled by organizing these tables in a cluster or as index-organized tables (Srinivasan, Das, Freiwald, Chong, Jagannath, Yalamanchi, Krishnan, Tran, DeFazio, & Banerjee, 2000). (See Figure 9c.)

Relational Storage of Index Data

In the following, we will discuss the basic properties of relational access methods with respect to the storage of index data, query processing and the overhead for transaction semantics, concurrency control and recovery services. We start with a basic definition:

Definition 1 (Relational Access Method). An access method is called a *relational access method* if any index-related data is exclusively stored in and retrieved from relational tables. An instance of a relational access method is called a *relational index*. The following tables comprise the persistent data of a relational index:

(i) **User table:** a single table, storing the original user data being indexed.

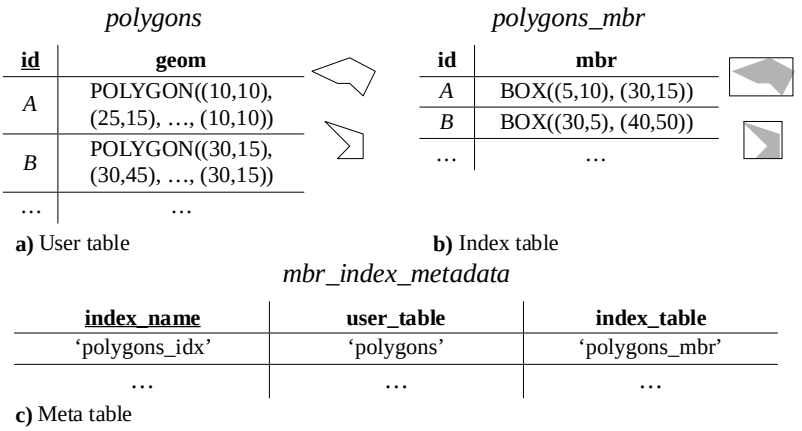
(ii) **Index tables:** n tables, $n \geq 0$, storing index data derived from the user table.

(iii) **Meta table:** a single table for each database and each relational access method, storing $O(1)$ rows for each instance of an index.

The stored data is called *user data*, *index data* and *meta data*.

To illustrate the concept of relational access methods, Figure 10 presents the minimum bounding rectangle list (*MBR-List*), a very simple example for indexing two-dimensional polygons. The user table is given by the object-relational table *polygons* (Figure 10a), comprising attributes for the polygon data type (*geom*) and the object identifier (*id*). Any spatial query can already be evaluated by sequentially scanning this user table. In order to speed up spatial selections, we decide to define an MBR-List *polygons_idx* on the user table. Thereby, an index table is created and populated (Figure 10b), assigning the minimum bounding rectangles (*mbr*) of each polygon to the foreign key *id*. Thus, the index table stores information purely derived from the user table. All schema objects

Figure 10. MBR-List, a simple example for a relational access method



belonging to the relational index (in particular the name of the index table) and other index parameters are stored in a global meta table (Figure 10c).

In order to support queries on the index tables, a relational access method can employ any built-in secondary indexes, including hash indexes, B+-trees and bitmap indexes. Alternatively, index tables may be clustered by appropriate primary indexes. Consequently, the relational access method and the database system cooperate to maintain and retrieve the index data (DeFazio, Daoud, Smith, & Srinivasan, 1995). This basic approach of relational indexing has already been applied in many existing solutions, including *Linear Quadtrees* (Tropf & Herzog, 1981; Ravada & Sharma, 1999; Freytag, Flasza, & Stillger, 2000) and *Relational R-trees* (Ravi, Ravada, Sharma, & Banerjee, 1999) for spatial databases, *Relational X-trees* (Berchtold, Böhm, Kriegel, & Michel, 1999) for high-dimensional nearest-neighbor search or inverted indexes for information retrieval on text documents (DeFazio, Daoud, Smith, & Srinivasan, 1995).

Operations on Relational Access Methods

In the strict sense of the definition, the procedural code of an arbitrary block-oriented storage structure can be immediately transformed to a relational access

method by replacing each invocation of the underlying block manager by an SQL-based DML operation. Thus, the original procedural style of an index operation remains unchanged, whereas its I/O requests are now executed by a fully-fledged RDBMS. The object-relational database server is thereby reduced to a plain block manager. In consequence, only a fraction of the existing functionality of the underlying database server is exploited. In this section, we define operations on relational access methods that maximize the architecture-awareness postulated in Jensen and Snodgrass (1999). This can be achieved by using declarative operations.

Cursor-Bound Operations

In order to guarantee a better exploitation of the database infrastructure, we have to restrict the possible number of DML operations submitted from a procedural environment:

Definition 2 (Cursor-Bound Operation). A query or update operation on a relational access method is termed *cursor-bound* if the corresponding I/O requests on the index data can be performed by submitting $O(1)$ DML statements, that is, by sequentially and concurrently opening in total $O(1)$ cursors provided by the underlying RDBMS.

Cursor-bound operations on relational access methods are largely bound to the declarative DML engine of the underlying RDBMS rather than to user-defined opaque code. Thus, the database server gains the responsibility for significant parts of the query and update semantics. Advantages of this approach include:

- **Declarative Semantics:** Large parts of a cursor-bound operation are expressed by using declarative SQL. By minimizing the procedural part and maximizing the declarative part of an operation, the formal verification of the semantics is simplified if we can rely on the given implementation of SQL to be sound and complete.
- **Query Optimization:** Whereas the database engine optimizes the execution of single, closed-form DML statements, a joint execution of multiple, independently submitted queries is very difficult to achieve (Sellis, 1988; Chen & Dunham, 1998; Braunmüller, Ester, Kriegel, & Sander, 2000). By using only a constant number of cursors, the RDBMS captures significant parts of the operational semantics at once. In particular, complex I/O operations — including external sorting, duplicate elimination or grouping — should be processed by the database engine and not by a user-defined procedure.

- **Cursor Minimization:** The CPU cost of opening a variable number of cursors may become very high. For typical applications, the resulting overhead sums up to 30% of the total processing time (Ramsak et al., 2000). In some experiments, we even reached barrier-crossing cost of up to 75% for submitting a variable number of pre-parsed DML statements out of a stored procedure. For cursor-bound operations, the relatively high cost of opening and fetching multiple database cursors remains constant with respect to the complexity of the operation and the database size.

Cursor-Driven Operations

A very interesting case occurs if the potential result of a cursor-bound operation can be retrieved as the immediate output of a *single* cursor provided by the DBMS. Thus, the semantics is revealed to the database server at once in its full completeness:

Definition 3 (Cursor-Driven Operation). A cursor-bound operation on a relational access method is called *cursor-driven* if it can be divided into two consecutive phases:

(*i*) **Procedural phase:** Index parameters are read from the meta tables. Query specifications are retrieved, and data structures required for the actual query execution may be prepared by user-defined procedures and functions. Additional DML operations on user data or index data are not permitted.

(*ii*) **Declarative phase:** In the second phase, only a single DML statement is submitted to the ORDBMS, yielding a cursor on the final results of the index scan which requires no post-processing by user-defined procedures or functions.

Note that any cursor-driven operation is also cursor-bound, while all I/O requests on the index data are driven by a single declarative DML statement. The major advantage of cursor-driven operations is their smart integration into larger execution plans. After the completion of the procedural phase, the single DML statement can be executed with arbitrary groupings and aggregations, supplemented with additional predicates or serve as a row source for joins. Furthermore, the integration into extensible indexing frameworks is facilitated, as the cursor opened in the declarative phase can be simply pipelined to the index scan routine. Note that the ability to implement cursor-bound and cursor-driven operations heavily relies on the expressive power of the underlying SQL interface, including the availability of recursive queries (Libkin, 2001).

The single DML statement submitted in the declarative phase may contain user-defined functions. The CPU cost of cursor-driven operations is significantly reduced if the number of barrier crossings due to calls to user-defined functions is minimized (Kornacker, 1999). We can achieve this by preprocessing any required transformation — for example, of a query specification — in the procedural phase and by bulk-binding the prepared data to the query statement with the help of transient collections. If such data structures become very large, a tradeoff has to be achieved between the minimization of barrier crossings and the main-memory footprint of concurrent sessions. Splitting a single query into multiple cursor-driven operations can then be beneficial.

To pick up the *MBR-List* example of the previous section, Figure 11a shows a simple window query on the database of two-dimensional polygons, testing the exact geometry of each stored polygon for intersection with the query rectangle. In order to use the relational index as primary filter, the query has to be rewritten into the form of Figure 11b. An efficient execution plan for the rewritten query may first check the intersection with the stored bounding boxes, and refine the result by performing the equijoin with the *polygons* table. Note that the window query is a cursor-driven operation on the MBR-List, having an empty procedural phase. Therefore, the index-supported query can be easily embedded into a larger context, as shown in Figure 11c. Already this small example shows that

Figure 11. MBR-List, a simple example for a relational access method

```
SELECT id FROM polygons
WHERE geom INTERSECTS BOX((0,0),(100,100));
```

a) Window query on the user table.

```
SELECT usr.id AS id FROM polygons usr, polygons_mbr idx
WHERE idx.mbr INTERSECTS BOX((0,0),(100,100))
AND idx.id = usr.id
AND usr.geom INTERSECTS BOX ((0,0),(100,100));
```

b) Window query using the relational index as primary filter.

```
SELECT id FROM polygon_type
WHERE type = 'LAKE'
AND id IN (
  SELECT usr.id FROM polygons usr, polygons_mbr idx
  WHERE idx.mbr INTERSECTS BOX((0,0),(100,100))
  AND idx.id = usr.id
  AND usr.geom INTERSECTS BOX ((0,0),(100,100))
);
```

c) Index-supported window subquery.

an object-relational wrapping of relational access methods is essential to control redundant data in the index tables and to avoid manual query rewriting. The usage of an extensible indexing framework preserves the physical independence of DML operations and enables the usual query optimization.

Although similarity queries or nearest neighbor queries ("*return the k polygons closest to a query point wrt. to a given metric*") can also be performed in a cursor-driven way by using the order-by clause with a *top-k*-filter, the efficiency of this approach is rather questionable (Carey & Kossmann, 1997).

Generic Schemes for Object-Relational Spatial Indexing

As an immediate result of the relational storage of index data and meta data, a relational index is subject to the built-in transaction semantics, concurrency control and recovery services of the underlying database system. In this section, we discuss the effectiveness and performance provided by the built-in services of the ORDBMS on relational access methods. For that purpose, we identify two generic schemes for the relational storage of index data, the *navigational* scheme and the *direct* scheme.

Navigational Scheme of Index Tables

Definition 4 (Navigational Scheme). Let $P = (T, R_1, ..., R_n)$ be a relational access method on a data scheme T and index schemes $R_1, ..., R_n$. We call P *navigational* $\Leftrightarrow (\exists\ t \subseteq T)\ (\exists\ r_i \subseteq R_i, 1 \le i \le n)$: at least one $r \in r_i$ is associated with rows $\{t_1, ..., t_m\} \subseteq t$ and $m > 1$.

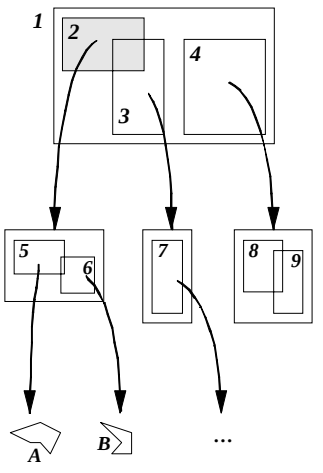
Therefore, a row in an index table of a navigational index may logically represent many objects stored in the user table. This is typically the case for hierarchical structures that are mapped to a relational schema. Consequently, an index table contains data that is recursively traversed at query time in order to determine the resulting tuples. Examples for the navigational scheme include the *Oracle Spatial R-tree* (Ravi Kanth, Ravada, Sharma, & Banerjee, 1999) and the *Relational X-tree* (Berchtold, Böhm, Kriegel, & Michel, 1999), which store the nodes of a tree directory in a flat table. To implement a navigational query as a cursor-bound operation, a recursive version of SQL, like SQL:1999 (ANSI, 1999; Eisenberg & Melton, 1999), is required.

Although the navigational scheme offers a straightforward way to simulate any hierarchical index structure on top of a relational data model, it suffers from the fact that navigational data is locked like user data. As two-phase locking on index tables is too restrictive, the possible level of concurrency is unnecessarily decreased. For example, uncommitted node splits in a hierarchical directory may lock entire subtrees against concurrent updates. Built-in indexes solve this problem by committing structural modifications separately from content changes (Kornacker & Banks, 1995). Unfortunately, this approach is not feasible on the SQL layer without breaking up the user transaction. A similar overhead exists with logging, as atomic actions on navigational data — for example, node splits — are not required to be rolled back in order to keep the index tables consistent with the data table. Therefore, relational access methods implementing the navigational scheme are only well suited for read-only or single-user environments.

Relational R-Trees: Spatial Example for Navigational Scheme

We illustrate the properties and drawbacks of the navigational scheme by the example of *Relational R-trees* like they have been used by Oracle developers Ravi Kanth, Ravada, Sharma and Banerjee (1999). Figure 12 depicts a hierar-

Figure 12. Relational mapping of an R-tree directory



a) Hierarchical directory

polygons_rtree

page id	page_lev	son_id	son_mbr
ROOT	3	1	BOX((0,0), (200,120))
1	2	2	BOX((0,0), (80,60))
1	2	3	BOX((60,20), (100,120))
1	2	4	BOX((140,20), (200,120))
2	1	5	…
2	1	6	…
5	0	A	…
6	0	B	…
…	…	…	…

b) Relational index table

Figure 13. Cursor-driven window query on a Relational R-tree

```
WITH RECURSIVE tree_traversal (page_lev, son_id, son_mbr) AS (
  SELECT page_lev, son_id, son_mbr FROM polygons_rtree
  WHERE page_id = ROOT
  UNION ALL
  SELECT next.page_lev, next.son_id, next.son_mbr
  FROM tree_traversal prior, polygons_rtree next
  WHERE prior.page_mbr INTERSECTS BOX((0,0),(100,100))
  AND prior.son_id = next.page_id
)                                          //declarative tree traversal
SELECT son_id AS id
FROM tree_traversal
WHERE page_lev = 0;                        //select data objects
```

a) Recursive window query on a Relational R-tree using SQL:1999

```
SELECT son_id AS id FROM polygons_rtree
WHERE page_lev = 0                         //select data object
START WITH page_id = ROOT
CONNECT BY
  PRIOR son_mbr INTERSECTS BOX((0,0),(100,100))
  AND PRIOR son_id = page_id;              //declarative tree traversal
```

b) Recursive window query on a Relational R-tree using Oracle SQL

chical R-tree along with a possible relational mapping (*page_id*, *page_lev*, *son_id*, *son_mbr*). The column *page_id* contains the logical page identifier while *page_lev* denotes its level in the tree. Thereby, 0 marks the level of the data objects and 1 marks the leaf level of the directory. The attribute *son_id* contains the *page_id* of the connected entry, while *son_mbr* stores its minimum bounding rectangle. Thus, *page_id* and *son_id* together comprise the primary key. In our example, the logical page ***2*** represents a partition of the data space that contains polygons *A* and *B*. The corresponding index row (***1***, 2, ***2***, …) is therefore logically associated with the rows (*A*, …) and (*B*, …) in the *polygons* user table (Figure 10). Thus, the Relational R-tree implements the navigational scheme of relational access methods.

The severe overhead of the navigational scheme already becomes obvious if a transaction inserts a new polygon and subsequently enlarges the bounding box of a node, for example, of the root node. Due to the common two-phase locking, this transaction will hold an exclusive lock on the row (***ROOT***, 3, ***1***, …) until commit or rollback. During this time, no concurrent transaction can insert polygons that induce an enlargement of the root region. The database server has to guarantee non-blocking reads (Oracle, 1999b) to support at least concurrent

queries on the Relational R-tree index. If the low concurrency of the Relational R-tree is acceptable, the relational mapping opens up a wide range of potential improvements (Kriegel, Pfeifle, Pötke, & Seidl, 2003).

To support the navigation through the R-tree table at query time, a built-in index can be created on the *page_id* column. Alternatively, the schema can be transformed to NF^2 (non- first normal form), where *page_id* alone represents the primary key, and a collection of (*son_id*, *son_mbr*) pairs is stored with each row. In this case, the static storage location of each tuple can be used as *page_id*, avoiding the necessity of a built-in index. A cursor-driven primary filter for a window query using recursive SQL is shown in Figure 13. We expect that future implementations of the SQL:1999 statement yield a depth-first traversal that is already hard-wired into the existing CONNECT BY clause of the Oracle server. The effectiveness of cursor-driven operations is illustrated in that the depicted statements already comprise the complete, pipelined query processing on the R-tree index.

Direct Scheme of Index Tables

Definition 5 (Direct Scheme). Let $P = (T, R_1, ..., R_n)$ be a relational access method on a data scheme T and index schemes $R_1,..., R_n$. We call P *direct* $\Leftrightarrow$ $(\forall\ t \subseteq T)$ $(r_i \subseteq R_i, 1 \le i \le n)$: each $r \in r_i$ is associated with a single row $t \in t$.

In consequence, for a relational access method of the direct scheme, each row in the user table is directly mapped to a set of rows in the index tables. Inversely, each row in an index table exclusively belongs to a single row in the user table. In order to support queries, the index table is organized by a built-in index, for example, a B+-tree. Examples for the direct scheme include our *MBR-List* (Figure 10), the *Linear Quadtree* (Samet, 1990), the one-dimensional *Relational Interval Tree* (Kriegel et al., 2000) and its optimization for interval sequences and multidimensional queries (Kriegel, Pötke, et al., 2001).

The drawbacks of the navigational scheme with respect to concurrency control and recovery are not shared by the direct scheme, as row-based locking and logging on the index tables can be performed on the granularity of single rows in the user tables. For example, an update of a single row *r* in the user table requires only the synchronization of index rows exclusively assigned to *r*. As the acquired locks are restricted to *r* and its exclusive entries in the index tables, they do not unnecessarily block concurrent operations on other user rows. In contrast to navigational indexes, the direct scheme inherits the high concurrency and efficient recovery of built-in tables and indexes.

Relational Interval Trees: Example for Direct Scheme

An access method implementing the direct scheme is the Relational Interval Tree (Kriegel, et al., 2000; Kriegel, Pötke, et al., 2001). Being a relational storage structure for interval data (*lower*, *upper*), it follows the concept of Edelsbrunner's main-memory interval tree (Edelsbrunner, 1983; Preparata & Shamos, 1993) by design and guarantees the optimal complexity for storage space and for I/O operations when updating or querying large sets of intervals.

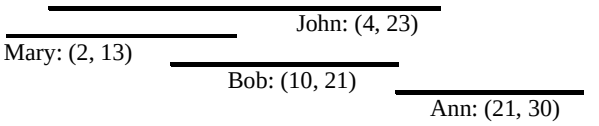
The structure of an RI-tree consists of a binary tree of height *h* which covers the range $[1, 2^h\text{-}1]$ of potential interval bounds. It is called the virtual backbone of the RI-tree since it is not materialized, but only the root value 2^{h-1} is stored persistently in a metadata table. Traversals of the virtual backbone are performed purely arithmetically by starting at the root value and proceeding in positive or negative steps of decreasing length 2^{h-i}, thus reaching any desired value of the data space in $O(h)$ CPU time and without causing any I/O operation. For the relational storage of intervals, the node values of the tree are used as artificial keys: Upon insertion of an interval, the first node that hits the interval when descending the tree from the root node down to the interval location is assigned to that interval.

An instance of the RI-tree then consists of two relational indexes which, in an extensible indexing environment, are preferably managed as index-organized tables. The indexes obey the relational schema *lowerIndex* (*node*, *lower*, *id*) and *upperIndex* (*node*, *upper*, *id*) and store the artificial key value *node*, the bounds *lower* and *upper*, and the *id* of each interval. An interval is represented by exactly one entry in each of the two indexes, and for inserting or deleting intervals, the *node* values are determined arithmetically without any I/O operation.

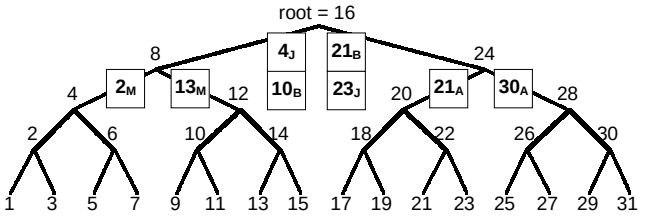
The illustration in Figure 14 provides an example for the RI-tree. Let us assume the intervals (2,13) for Mary, (4,23) for John, (10,21) for Bob and (21,30) for Ann (Figure 14a). The virtual backbone is rooted at 16 and covers the data space from 1 to 31 (Figure 14b). The intervals are registered at the nodes 8, 16 and 24. The interval (2,13) for Mary is represented by the entries (8, 2, Mary) in the lowerIndex and (8, 13, Mary) in the upperIndex since 8 is the registration node, and 2 and 13 are the lower and upper bound, respectively (Figure 14c).

Again, to minimize barrier crossings between the procedural runtime environment and the declarative SQL layer, an interval intersection query (*lower*, *upper*) is processed in two steps. The procedural query preparation step descends the virtual backbone from the root node down to *lower* and to *upper*, respectively. The traversal is performed arithmetically without causing any I/O operations, and the visited nodes are collected in two main-memory tables, *left*

Figure 14. Example for an RI-tree



a) Four intervals



b) Virtual backbone and registration positions of the intervals

lowerIndex (node, lower, id)

8, 2, Mary	16, 4, John	16, 10, Bob	24, 21, Ann

upperIndex (node, upper, id)

8, 13, Mary	16, 21, Bob	16, 23, John	24, 30, Ann

c) Resulting relational indexes *lowerIndex* and *upperIndex*

queries and *right queries*, as follows: nodes to the left of *lower* may contain intervals which overlap *lower* and are inserted into *left queries*. Analogously, nodes to the right of *upper* may contain intervals which overlap *upper* and are inserted into *right queries*. Whereas these nodes are taken from the paths, the set of all nodes between *lower* and *upper* belongs to the so-called *inner query*, which is represented by a single range query on the node values. All intervals registered at nodes from the *inner query* are guaranteed to intersect the query and, therefore, will be reported without any further comparison. The query preparation step is purely based on main memory and requires no I/O operations.

In the subsequent declarative query processing step, the transient tables are joined with the relational indexes *upperIndex* and *lowerIndex* by a single, three-fold SQL statement (Figure 15). The upper bound of each interval registered at nodes in *left queries* is compared to *lower*, and the lower bounds of intervals stemming from *right queries* are compared to *upper*. The *inner query*

Figure 15. SQL statement for an intersection query

```
SELECT id FROM upperIndex i, :leftQueries q
  WHERE i.node =q.node AND i.upper >=:lower
UNION ALL
SELECT id FROM lowerIndex i, :rightQueries q
  WHERE i.node = q.node AND i.lower <= :upper
UNION ALL
SELECT id FROM lowerIndex                    // or upperIndex
  WHERE node BETWEEN :lower AND :upper;
```

corresponds to a simple range scan over the intervals with nodes in (*lower*, *upper*).

Recently, a further relational access method implementing the direct scheme was presented (Arge & Chatham, 2003). Like the Relational Interval Tree, the *Relational Priority Search Tree* is a storage structure for handling interval data.

Kriegel et al. (2003) describe the *Linear Quadtree*, another access method with direct scheme implementation.

Conclusions

We presented the concept of object-relational spatial access methods that employ the infrastructure and functionality of existing object-relational database systems to provide efficient execution plans for the evaluation of user-defined predicates. We introduced cursor-bound and cursor-driven operations to maximize the achievable declarativity, usability and performance of operations. We identified two generic schemes for the relational mapping of index data, each having different properties with respect to the built-in locking and logging mechanisms of the underlying database engine: Whereas the *navigational* scheme seems only appropriate for single-user or read-only databases, the *direct* scheme fully preserves the effectivity and efficiency of built-in transactions, concurrency control and recovery services. The presented concepts have been illustrated by four spatial examples: The *MBR-List*, a trivial relational access method for demonstration purposes, the *Relational R-tree* showing the navigational scheme, along with *Relational Interval Tree*, an access method implementing the direct scheme.

Future research may investigate whether there are generic patterns to develop a relational indexing scheme for any given index structure. Again, a careful analysis of the potentials and the overhead of relational data management is a major point of interest. The development of more powerful extensibility frameworks supporting features as generic indexing interfaces and user-defined join algorithms is a challenge for forthcoming years.

References

American National Standards Institute: *ANSI X3.135-1992/ISO 9075-1992 (SQL-92)* (1992). New York.

American National Standards Institute: *ANSI/ISO/IEC 9075-1999 (SQL:1999, Parts 1-5)* (1999). New York.

Aoki, P.M. (1998). Generalizing "Search" in generalized search trees. *Proceedings of the 14th International Conference on Data Engineering (ICDE)*, (pp. 380-389).

Arge L., & Chatham, A. (2003). Efficient object-relational interval management and beyond. *Proceedings of the Eighth International Symposium on Spatial and Temporal Databases (SSTD), LNCS 2750*, (pp. 66-82).

Arge, L., & Vitter, J.S. (1996). Optimal dynamic interval management in external memory. *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, (pp. 560-569).

Batory, D.S., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., & Wise, T.E. (1990). GENESIS: An extensible database management system. In S.B. Zdonik & D. Maier (Eds.), *Readings in object-oriented database systems* (pp. 500-518). San Francisco: Morgan Kaufman.

Berchtold, S., Böhm, C., Kriegel, H.-P., & Michel, U. (1999). Implementation of multidimensional index structures for knowledge discovery in relational databases. *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK), LNCS 1676*, (pp. 261-270).

Bliujute, R., Saltenis, S., Slivinskas, G., & Jensen, C.S. (1999). Developing a DataBlade for a new index. *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, (pp. 314-323).

Braunmüller, B., Ester, M., Kriegel, H.-P., & Sander, J. (2000). Efficiently supporting multiple similarity queries for mining in metric databases.

Proceedings of the 16th International Conference on Data Engineering (ICDE), (pp. 256-267).

Brown, P. (2001). *Object-relational database development: A plumber's guide*. Menlo Park, CA: Informix Press.

Carey, M.J., & Kossmann, D. (1997). On saying "enough already!" in SQL. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (pp. 219-230).

Carey, M.J., DeWitt, D.J., Frank, D., Graefe, G., Richardson, J.E., Shekita, E.J., & Muralikrishna, M. (1991). The architecture of the EXODUS extensible DBMS. In K.R. Dittrich, U. Dayal, & A.P. Buchmann (Eds.), *On object-oriented database systems* (pp. 231-256). New York: Springer.

Carey, M.J., DeWitt, D.J., Graefe, G., Haight, D.M., Richardson, J.E., Schuh, D.T., Shekita, E.J., & Vandenberg, S.L. (1990). The EXODUS extensible DBMS project: An overview. In S.B. Zdonik & D. Maier (Eds.), *Readings in object-oriented database systems* (pp. 474-499). San Francisco: Morgan Kaufmann.

Chen, F.-C.F., & Dunham, M.H. (1998). Common subexpression processing in multiple-query processing. *IEEE Transcript on Knowledge and Data Engineering, 10*(3), 493-499.

Chen, W., Chow, J.-H., Fuh, Y.-C., Grandbois, J., Jou, M., Mattos, N., Tran, B., & Wang, Y. (1999). High level indexing of user-defined types. *Proceedings of the 25th International Conference on Very Large Databases (VLDB),* (pp. 554-564).

DeFazio, S., Daoud, A., Smith, L.A., & Srinivasan, J. (1995). Integrating IR and RDBMS using cooperative indexing. *Proceedings of the 18th ACM SIGIR Conference on Research and Development in Information Retrieval,* (pp. 84-92).

DeWitt, D.J., Katz, R.H., Olken, F., Shapiro, L.D., Stonebraker, M., & Wood, D.A. (1984). Implementation techniques for main memory database systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (pp. 1-8).

Edelsbrunner, H. (1983). A new approach to rectangle intersections. *International Journal of Computer Mathematics, 13,* 209-229.

Eisenberg, A., & Melton, J. (1999). SQL:1999, formerly known as SQL3. *ACM SIGMOD Record, 28*(1), 131-138.

Freytag, J.-C., Flasza, M., & Stillger, M. (2000). Implementing geospatial operations in an object-relational database system. *Proceedings of the 12th International Conference on Scientific and Statistical Database Management (SSDBM),* (pp. 209-219).

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns*. Boston: Addison Wesley Longman.

Garcia-Molina, H., & Salem, K. (1992). Main memory database systems: An overview. *IEEE Transcript on Knowledge and Data Engineering, 4*(6), 509-516.

Hellerstein, J.M., Naughton, J.F., & Pfeffer, A. (1995). Generalized search trees for database systems. *Proceedings of the 21st International Conference on Very Large Databases*, (pp. 562-573).

IBM Corporation. (1999). *IBM DB2 Universal Database Application Development Guide, Version 6*. Armonk, NY.

Informix Software Inc. (1998). *DataBlade Developers Kit User's Guide, Version 3.4*. Menlo Park, CA: Informix Press.

Informix Software Inc. (1999). *Informix R-Tree Index User's Guide, Version 9.2*. Menlo Park, CA: Informix Press.

Jensen, C.S., & Snodgrass, R.T. (1999). Temporal data management. *IEEE Transcript on Knowledge and Data Engineering, 11*(1), 36-44.

Keim, D.A., & Prawirohardjo, E.S. (1992). *Datenbankmaschinen – Performanz durch Parallelität*. Reihe Informatik 86, BI Wissenschaftsverlag, Mannheim.

Kornacker, M. (1999). High-performance extensible indexing. *Proceedings of the 25th International Conference on Very Large Databases (VLDB)*, (pp. 699-708).

Kornacker, M., & Banks, D. (1995). High-concurrency locking in R-trees. *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, (pp. 134-145).

Kornacker, M., Mohan, C., & Hellerstein, J.M. (1997). Concurrency control in generalized search trees. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (pp. 62-72).

Kriegel, H.-P., Müller, A., Pötke, M., & Seidl, T. (2001). Spatial data management for computer aided design (Demo). *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (p. 614).

Kriegel, H.-P., Pfeifle, M., Pötke, M., & Seidl, T. (2002). A cost model for interval intersection queries on RI-trees. *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM)*, (pp. 131-141). Edinburgh, UK.

Kriegel, H.-P., Pfeifle, M., Pötke, M., & Seidl, T. (2003). *The paradigm of relational indexing: A survey*. BTW 2003, 285-304.

Kriegel, H.-P., Pötke, M., & Seidl, T. (2000). Managing intervals efficiently in object-relational databases. *Proceedings of the 26th International Con-*

ference on Very Large Databases (VLDB), Datenbanksysteme fur Business, Technologie und Web (BTW) (pp. 407-418).

Kriegel, H.-P., Pötke, M., & Seidl, T. (2001). Interval sequences: An object-relational approach to manage spatial data. *Proceedings of the Seventh International Symposium on Spatial and Temporal Databases (SSTD), LNCS 2121*, (pp. 481-501).

Libkin, L. (2001). Expressive power of SQL. *Proceedings of the Eighth International Conference on Database Theory (ICDT)*, (pp. 1-21).

Oracle Corp. (1999a). *Oracle8i Data Cartridge Developer's Guide, Release 2 (8.1.6*). Oracle Corporation, Redwood Shores, California.

Oracle Corp. (1999b). *Oracle8i Concepts, Release 8.1.6.* Oracle Corporation, Redwood Shores, California.

Oracle Corp. (2000). *Oracle8i Appliance – An Oracle White Paper.* Oracle Corporation, Redwood Shores, California.

PostgreSQL Global Development Group (2002). *PostgreSQL 7.3.2 Programmer's Guide.* PostgreSQL Global Development Group, Berkeley, California.

Preparata, F.P., & Shamos, M.I. (1993). *Computational geometry: An introduction* (5th edition). New York: Springer.

Ramsak, F., Markl, V., Fenk, R., Zirkel, M., Elhardt, K., & Bayer, R. (2000). Integrating the UB-tree into a database system kernel. *Proceedings of the 26th International Conference on Very Large Databases (VLDB)*, (pp. 263-272).

Ravada, S., & Sharma, J. (1999). Oracle8i Spatial: Experiences with extensible databases. *Proceedings of the Sixth International Symposium on Large Spatial Databases (SSD), LNCS 1651*, (pp. 355-359).

Ravi Kanth, K.V., Ravada, S., Sharma, J., & Banerjee, J. (1999). Indexing medium-dimensionality data in Oracle. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (pp. 521-522).

Samet, H. (1990). *Applications of spatial data structures*. Boston: Addison Wesley Longman.

Sellis, T.K. (1998). Multiple-query optimization. *ACM Transactions on Database Systems (TODS), 13*(1), 23-52.

Srinivasan, J., Das, S., Freiwald, C., Chong, E.I., Jagannath, M., Yalamanchi, A., Krishnan, R., Tran, A.-T., DeFazio, S., & Banerjee, J. (2000). Oracle8i Index-Organized Table and Its Application to New Domains. *Proceedings of the 26th International Confrence on Very Large Databases (VLDB)*, (pp. 285-296).

Srinivasan, J., Murthy, R., Sundara, S., Agarwal, N., & DeFazio, S. (2000). Extensible indexing: A framework for integrating domain-specific indexing schemes into Oracle8i. *Proceedings of the 16th International Conference on Data Engineering (ICDE)*, (pp. 91-100).

Stonebraker, M. (1986). Inclusion of new types in relational data base systems. *Proceedings of the Second International Conference on Data Engineering (ICDE)*, (pp. 262-269). San Francisco: Morgan Kaufmann.

Stonebraker, M., & Brown, P. (1999). *Object-relational DBMSs – Tracking the next great wave*. Morgan Kaufman.

Stonebraker, M., & Kemnitz, G. (1991). The Postgres next generation database management system. *Commun. ACM, 34*(10), 78-92.

Tropf, H., & Herzog, H. (1981). Multidimensional range search in dynamically balanced trees. *Angewandte Informatik, 23*(2), 71-77.

Yu, C.T., & Meng, W. (1998). *Principles of database query processing for advanced applications*. San Francisco: Morgan Kaufmann.

Chapter IV

Quadtree-Based Image Representation and Retrieval

Maude Manouvrier
LAMSADE – Université Paris-Dauphine, France

Marta Rukoz
CCPD – Universidad Central de Venezuela, Venezuela

Geneviève Jomier
LAMSADE – Université Paris-Dauphine, France

Abstract

This chapter is a survey of quadtree uses in the image domain, from image representation to image storage and content-based retrieval. A quadtree is a spatial data structure built by a recursive decomposition of space into quadrants. Applied to images, it allows representing image content, compacting or compressing image information, and querying images. For 13 years, numerous image-based approaches have used this structure. In this chapter, the authors underline the contribution of quadtree in image applications.

Introduction

A *quadtree* (Finkel & Bentley, 1974; Klinger, 1971) is a well-known unbalanced spatial data structure built by recursive divisions of space in four equal-size disjoint quadrants. This chapter focuses on image domain. A quadtree has been used frequently to represent an image or picture in various applications. For example, Ahmad and Grosky (1997, 2003), Albuz, Kocalar and Khokhar (2000), Kim and Kim (2000), Lin, Tamer Özsu M, Oria and Ng (2001), Lu, Ooi and Tan (1994), Malki, Boujemaa, Nastar and Winter (1999) and Rukoz, Manouvrier and Jomier (2002) exploit it in the purpose of content-based image retrieval. Baligar, Patnaik and Nagabhushana (2003), Cheng and Li (1996), Jackson, Mahmoud, Stapleton and Gaughan (1997), Kim and Lee (2002), Li, Knipe and Cheng (1997), Shusterman and Feder (1994) and Strobach (1991) compress images using quadtrees. The quadtree is also used in computer graphics by Samet and Webber (1988); in image processing by Lin (1997a, 1997b), Smith and Chang (1994) and Yang, Chung and Tsai (2000); in Geographical Information Systems (GIS) by Aref and Samet (1997) and Shaffer, Samet and Nelson (1990); and in image databases by Jomier, Manouvrier and Rukoz (2000), Manouvrier, Rukoz and Jomier (2002), Tzouramanis, Vassilakopoulos and Manolopoulos (1998-2001) and Vassilakopoulos, Manolopoulos and Economou (1993-1995).

This chapter surveys the different applications of quadtree in the image domain. In the first part, the principles of quadtree representation are recalled. The second part presents several approaches minimizing the memory space used by encoding image quadtrees in a linear form or by compressing images using quadtrees. The third part gives an overview of the different approaches proposed for the storage and manipulation of clusters of images. Finally, the last part deals with the Content-Based Image Retrieval approaches using quadtrees.

Quadtree-Based Image Representation

Different types of data, like curves, surfaces or volumes, can be represented by quadtrees. A survey of the different quadtree types is presented by Samet (1984, 1990) and online demos are proposed by Brabec and Samet (2003). The most widely known quadtree, called *region quadtree*, allows cutting an image in regions or quadrants according to a given split criterion (for example, color homogeneity). As explained by Shusterman and Feder (1994), a quadtree allows representing images at different levels of resolution. This section recalls the general principles of quadtree and presents approaches using it to store image feature vectors.

Figure 1. (a) The Lena image and its quadtree decomposition, using different percentages of color homogeneity: (b) 70%, (c) 90% and (d) 100%

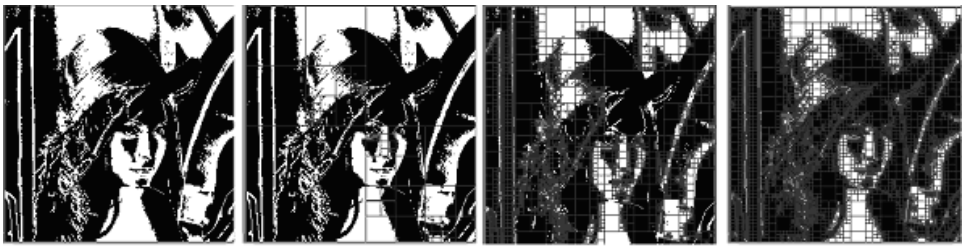


Image Quadtree

To be represented by a quadtree, an image is recursively divided into four equal-size quadrants until a stopping condition is met: Each node of the quadtree represents a quadrant of the image. Examples of split criteria are color or texture homogeneity of an image quadrant or a maximal number of feature points in each image quadrant. This process is called quadtree decomposition, or quadtree segmentation, by De Natale and Granelli (2001). Figure 1, for example, shows several quadtree decompositions of the Lena image, using different percentages of color homogeneity.

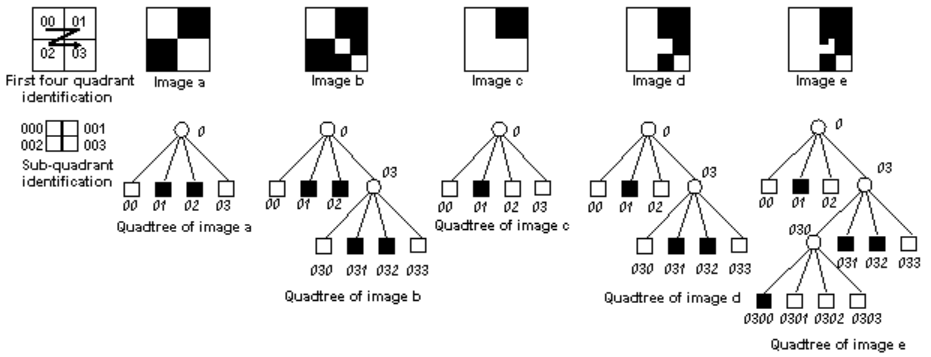
The root node of a quadtree represents the initial quadrant containing the whole image. If an image does not conform to the chosen split criterion, then the root has four descendant nodes representing the four first-level image quadrants. A node is a leaf when its corresponding image quadrant conforms to the split criterion; otherwise, the node is internal or non-terminal. In case of *binary images*, that is, containing *black* (B) and *white* (W) pixels, internal nodes are characterized as *gray* (G). To make the explanation clear, we use (in Figure 2) a special simple case of binary images, where the split criterion, which stops the subdivision of an image quadrant, is the homogeneity according to the black or white pixel colors.

Let k be the number of internal nodes in a quadtree. As explained by Cheng and Li (1996), a non empty quadtree has $4k+1$ nodes: k internal nodes and $3k+1$ leaf nodes. A quadtree is called *full* when all its leaves appear at the same level. A full balanced quadtree having h levels, with $h \geq 1$, contains $\sum_{i=1}^{h} 4^{i-1} = \frac{4^h - 1}{3}$ nodes.

To easily retrieve the quadtree node associated with an image quadrant and vice versa, a quadtree node and its corresponding image quadrant use the same identifier. In Abel (1984), Gargantini (1982), Jomier et al. (2000), Samet (1984) and Tzouramanis et al. (2000), different encoding methods are used to associate an identifier (also called *locational code*, *quadcode* or *locational key*) with a quadtree node. In this chapter, we use a *Z-ordering*, following the NW, NE, SW, SE directions, as shown in Figure 2. The numeral *0* identifies the initial quadrant representing the whole image. Numerals *0*, *1*, *2* or *3*, following their parent node identifier *0*, identify the four first-level quadrants. Recursively, sub-quadrants of an image quadrant *n* are identified by *nx* where $x \in \{0,1,2,3\}$. Let *N* be the set of quadtree node identifiers. Two nodes with the same identifier *n* ($n \in N$) in two different quadtrees are called *homologous nodes*.

A quadtree can be used to handle positional information of image features. For example, De Natale and Granelli (2001) exploit the quadtree structure for image color segmentation. Chakrabarti, Ortega-Binderberger, Porkaew, Zuo and Mehrotra (2000) exploit it for shape, and Ahmad and Grosky (1997, 2003) for locating object features. In De Natale and Granelli (2001), a quadtree segmentation is used to extract the distribution of dominant colors in an image where maximum and minimum quadrant sizes are fixed. The authors define the dominant color of a quadrant as the color with the higher percentage of occurrences inside the region represented by the quadrant. Each color image is represented by a quadtree, whose nodes (internal and leaf) store the dominant color of the corresponding image quadrant. In Ahmad and Grosky (1997, 2003), images have a symbolic representation and are recursively decomposed into a spatial arrangement of feature points in a quadtree. In this case, each leaf node represents an image quadrant containing zero or exactly one feature point and each internal node contains the number of feature points in their rooted sub-tree. In Chakrabarti et al.(2000), quadtree decomposition is used to represent two-

Figure 2. Five synthetic binary images and their region quadtree representation



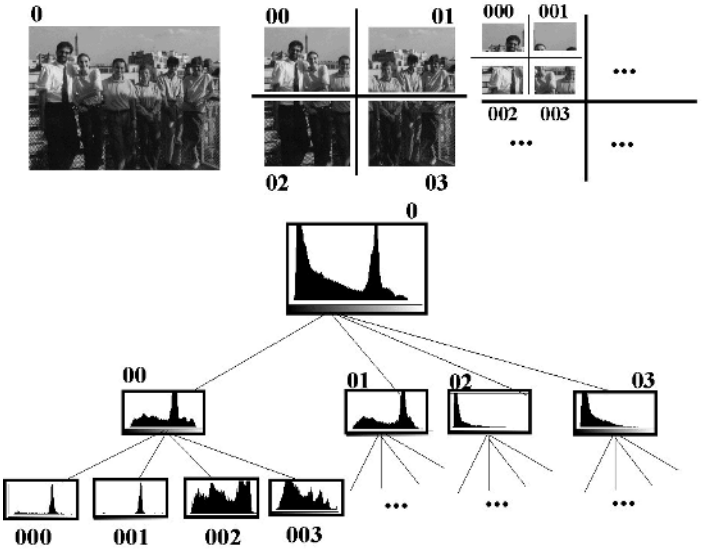
dimensional shapes. Each black leaf node in the quadtree represents a part of the decomposed shape. The resolution chosen to decompose the shape (that is, the size of the quadrant representing a homogeneous region) can be adapted depending on the quality of the shape representation.

Quadtree-Based Image Feature Vector

Several approaches use quadtree to store image features. In these approaches, each image is represented by a quadtree having a fixed number of levels, usually a full 3-level quadtree, which appeared sufficient in Lin et al. (2001). Each quadtree node (internal or leaf) represents an image quadrant by storing its features; for example, a color histogram as in Lu et al. (1994) and Lin et al. (2001), a shape feature vector as in Kim and Kim (2000) or a combination of color, shape and texture captured via histograms as in Malki et al. (1999). We call such a structure a *Quadtree-based Feature Vector* (QFV). It is used as a *multi-level filtering,* as it is suitable to a *coarse-to-fine* representation.

Figure 3 gives an example of a multi-level histogram: each quadtree node contains the color histogram of the corresponding image quadrant. For instance, the histogram labeled *00* corresponds to quadrant *00*, given by the first subdivision of the image. Histograms stored in each child node (labeled *000* to

Figure 3. An example of a quadtree-based feature vector containing color histograms



003) of quadtree node *00* correspond to the four sub-quadrants of quadrant *00* (top right in Figure 3).

Image Compression or Compaction Using Quadtree

Quadtrees can be used to compact or compress images. *Compaction* means lossless compression: the exact original image can be recovered. In case of *lossy compression*, only a close approximation of the original image can be obtained. In this chapter, we use the term *compaction* to denote lossless compression methods and *compression* to denote lossy ones. The following subsections present several approaches for compacting images by linear quadtrees or compressing images using quadtree.

Linear Quadtrees for Binary Images

As shown by Stewart (1986), the hierarchical implementation of quadtree uses pointers to nodes and is costly in memory space. To avoid this problem, several approaches store quadtrees in a linear way. A linear representation of a quadtree is a list of values that stores the hierarchical tree structure. Node values are encoded following a pre-order, also called depth-first order, as in Abel (1984); Gargantini (1982); Kawaguchi and Endo (1980); and Yang, Chung and Tsai (2000); or following a breadth first order, as in Chang and Chang (1994) and Lin (1997a). As explained by Chen (2002), the linear quadtree representation of an image generally requires less storage space than its bit pattern representation. This compact encoding is generally used for binary images. Table 1 shows the linear quadtrees of images *a*, *b* and *e* represented in Figure 2 according to the approaches presented below.

Gargantini (1982) and Abel (1984) represent a binary image quadtree as a list of its black leaf nodes. Gargantini (1982) encodes each black node by a quaternary code with digits *0* (for NW), *1* (for NE), *2* (for SW) or *3* (for SE) in base 4, where each successive digit represents the quadrant subdivision from which it originates according to a depth-first traversing (following a Z-ordering). All quaternary codes have the same number *h* of digits, where *h* is the number of levels of the quadtree (height of the quadtree). If a black node is at level *p*, $p<h$, then its quaternary code contains only *(h-p)* digits and ends with *p* symbols *X*, called "don't-care symbol" by Chen (2002). Abel (1984) codes the SW, NW, SE, NE directions by *1*, *2*, *3* or *4*, following an N-ordering. The identifier values are given

Table 1. Linear quadtrees of images a, b and e of Figure 2

Linear quadtree approach	Image *a*	Image *b*	Image *e*
Gargantini (1982):	{1,2}	{1X,2X,31,32}	{1XX,300,31X,32X}
Abel (1984):	(11,14)	(110,131,134,140)	(1310,1322,1340,1400)
Kawaguchi & Endo (1980):	(0110	(011(0110	(010((1000110
Chang & Chang (1994):	0110	0001 011 0110	0001 010 1000 110 1000
Lin (1997a):	0110	0113 0110	0102 3110 1000
Yang et al. (2000):	<12>	<66,12>	<57,14,1>

in base 5 and begin by a *1*, symbol *1* representing the root node. The fill character is a zero: If a black node is at level p, $p<h$, then the identifier contains p zeros. Moreover, in Abel (1984), all quadtree node identifiers are stored in a B^+-tree.

Kawaguchi and Endo (1980), Chang and Chang (1994), Lin (1997a) and Yang et al. (2000) code all quadtree nodes by a *position code*, indicating if the node is internal or not, and code all leaf nodes by a *color code*, following a Z-ordering. The position and color code appellations come from Chen (2002). In all these approaches, symbol *1* represents the color code of a black leaf node and symbol *0* represents the color code of a white one. The root node is not coded. The *Depth-First Expression* of Kawaguchi and Endo (1980) is a linear implementation, where a left parenthesis represents a gray (internal) node. The quadtree is coded according to a depth-first ordering. In the *Fixed Binary Linear Quadtree* (FBLQ) code of Chang and Chang (1994), internal nodes are coded by position code *1* and leaf nodes by position code *0*, except for leaf nodes at the bottom level and except if the root node has four leaf nodes. Moreover, each leaf node is coded by its color code, following its position code if it exists, in a breadth-first traversal of the quadtree. Thus, a list containing four positional codes is immediately followed by a list of color codes whose length depends on the number of zero (leaf nodes) in the previous positional code list. Because each leaf node has two codes, position and color, the FBLQ code length is qualified as *variant* by Chen (2002). In the *Constant Bit-length Linear Quadtree* (CBLQ) code of Lin (1997a), the numeral *2* codes an internal node if at least one of its descendants is internal. Otherwise, if all its descendants are leaves, a *3* codes the internal node. A quadtree is represented by a CBLQ code following a breadth first order. In the *Compact Improved Quadtree* or Compact-IQ of Yang et al. (2000), the entire image is encoded by a list $P=<P_1, P_2,...,P_k>$ where k is the number of gray nodes in the quadtree of the image. The quadtree is read using a depth-first order. Each gray node G_u, $u \in [1,k]$, is encoded using the node code (whose value is *0* for white, *1* for black and *2* for internal node) of its four descendants. More formally, the coding value of a gray node G_u, noted P_u, is given by:

$$P_u = \sum_{x=0}^{3} C_x(G_u) \times 3^x$$

where $C_x(G_u)$ is the code associated with the x-direction child of G_u, $x \in \{0,1,2,3\}$ denoting directions NW, NE, SW and SE. For example, quadtree a in Figure 2 is represented by the list $<12>$, coding the value of internal node 0, because $0*3^0+1*3^1+1*3^2+0*3^3=12$. Quadtree b in Figure 2 is represented by the list $<66,12>$, coding the values of internal nodes 0 and 03, because $0*3^0+1*3^1+1*3^2+2*3^3=66$ and $0*3^0+1*3^1+1*3^2+0*3^3=12$. Furthermore, Yang et al. (2000) use another coding method to reduce the number of bits required for representing each value P_u - for details we refer to (Yang et al., 2000).

According to Chen (2002), breadth-first linear coding, like FBLQ (Chang & Chang, 1994) or CBLQ (Lin, 1997a), requires less storage space than depth-first representation, such as the proposal of Gargantini (1982). However, the operations performed on breadth-first linear coding are more complex and more time-consuming than those performed on depth-first representation (for details on quadtree-based operations, see Samet, 1984; Lin, 1997b; and Manouvrier et al., 2002). Thus, Chen (2002) proposes algorithms for code transformations between breadth-first (FLBQ and CLBQ) and depth-first (Gargantini, 1982) linear quadtrees in order to exploit the advantages of both representations: an image can be stored or transmitted in its corresponding breadth-first linear quadtree and can be manipulated in its corresponding depth-first linear quadtree representation.

Lossless Compression using Linear Quadtree

The approaches presented above compact binary images by linear quadtrees. Other approaches, as Albuz, Kocalar and Khokhar (2000), and Baligar, Patnaik and Nagabhushana (2003), extend the linear quadtree to lossless coding of grayscale or color images. Baligar et al. (2003) propose a lossless coding for gray-level images using a predictive coding method. In this approach, an image is decomposed into a number of fixed-size blocks, and "prediction coefficients" are determined. Then, the image is represented by an error image, obtained by subtracting predictive values of pixels from their exact values. Each error image is compacted by a linear quadtree. The experimental results indicate that the proposed method is more effective than the Lossless JPEG (JPEG-LS) method (ISO, 1997) in terms of coding performance.

When images contain only black-and-white pixels or grayscale ones, the compression is lossless because pixel values are easy to represent (see previous

subsection). To compact color images, each image must be decomposed into bit planes or gray-level planes, each one corresponding to a dimension of the color feature space of the image. In Albuz et al. (2000), for example, each color image of the database is represented in the CIELAB-based color model and is segmented in four disjoint color planes, a pixel only belonging to a single color plane. An image results in four sub-images, each one containing the pixels of one of the four color planes and being represented by a binary quadtree. Thus, each image is represented by four binary quadtrees, each one representing the homogeneous regions of the image according to the corresponding color plane. The lossless compression consists of coding each quadtree independently by linear representation.

Quadtree-Based Lossy Compression Methods

Two kinds of approaches use quadtrees to compress images: those completely based on quadtrees and those using quadtrees to improve existing compression methods.

In the approaches presented in the previous two sections, quadtrees are used to compact images: Each leaf node contains the exact value of the corresponding image quadrant (which can be a pixel). Lossy compression methods, such as Strobach (1991), Shusterman and Feder (1994), Cheng and Li (1996), Li et al. (1997) and Kim and Lee (2002), differ from lossless ones in two points. First, several thresholds, one for each resolution level, are used to decompose an image quadrant in a quadtree. Second, the number of bits used to represent each quadtree node value is not the same for different nodes: The value of a node corresponding to a small image quadrant can be represented by fewer pixels than a node value representing a bigger image quadrant. According to Li et al. (1997) and Kim and Lee (2002), the performance of quadtree-based compression methods is similar or may outperform the JPEG algorithm presented in Pennebaker and Mitchell (1992).

Quadtrees can also be used to improve existing compression methods. For example, Ramos and Hemami (1996) adapt the block-based transform coding method JPEG, described in Pennebaker and Mitchell (1992). In this approach, image edges are decomposed in a quadtree, and the quadtree decomposition identifies the image blocks used by the JPEG compression algorithm. In the same way, Jackson et al. (1997) use a quadtree decomposition to improve a fractal image compression method.

The approaches presented above compact or compress a single image. The following section presents approaches for minimizing the space used to store a cluster of images.

Quadtree-Based Storage and Management of Image Clusters

The clustering of images consists of grouping together images with a predefined relationship. This section presents different quadtree-based approaches focused on optimizing the storage of image clusters. The goal is to maximize the sharing of common parts between quadtrees. The first subsection describes several distances used to evaluate the ratio of common parts between two quadtrees. The second subsection presents several approaches based on overlapping mechanisms. The last subsection presents several approaches for storing a cluster of images in a single quadtree and sharing common parts between images. A survey of these approaches is presented in Manouvrier et al. (2002).

Sharing Ratio Measurement

To evaluate the sharing of common parts between two image quadtrees, a distance or a similarity measure between quadtrees can be defined. De Natale and Granelli (2001), for example, have defined a quadtree structure similarity computed by a *quadtree warping* process: The difference between two quadtrees is evaluated through the number of changes in the structure (leaf split or leaves merge) that need to be performed to make both quadtree structures equivalent. For example, to make equivalent quadtrees *a* and *b*, represented in Figure 2, node *03* of quadtree *a* should be split in four white leaves. More formally, let $N_c(i,j)$ be the number of changes in the quadtrees *i* and *j*, in the warping process. The *Quadtree Structure Similarity* (QSS) between image quadtrees *i* and *j* is defined by:

$$QSS(i, j) = \frac{1}{Nc(i, j) + 1}.$$

For example, the *QSS* distance between image quadtrees *a* and *b*, represented in Figure 2, is equal to: $QSS(a, b)=1/5=0.2$ because four leaves are created in quadtree *a*. As explained by De Natale and Granelli (2001), in case of perfect matching between both quadtrees *i* and *j*, $QSS(i, j)=1$.

Rukoz et al. (2002) have defined several distance metrics between quadtrees (see the Quadtree-Based Distances section), and particularly a distance between quadtree structures called *Q*-distance. This distance uses the concept of node difference: Two homologous nodes are different if they have different

types (leaf or internal) and/or different values. The *Q*-distance computes the ratio of the number of different homologous nodes between two quadtrees to the total number of nodes in both quadtrees. The *Q*-distance between two image quadtrees *i* and *j* is zero, *Q(i,j)=0*, when all leaf nodes are at the same position and have the same value in both quadtrees. More formally, the *Q*-distance between image quadtrees *i* and *j* is defined as a sum of d_n-distances between quadtree nodes *n*, normalized by *U(i,j)*, the number of nodes in both quadtrees *i* and *j*:

$$Q(i,j) = \frac{\sum_{n \in N} d_n(i,j)}{U(i,j)}.$$

When homologous nodes *n* are both internal or are both leaves with the same value in quadtrees *i* and *j*, then d_n(i,j)=0. Otherwise, when node *n* is internal in one quadtree and is a leaf in the other quadtree, or when a node *n* exists only in one quadtree and not in the other, then d_n(i,j)=1. For example, the *Q*-distance between image quadtrees *a* and *b*, represented in Figure 2, is equal to *Q(a,b)= 5/9=0.55*. The numerator is 5 because five homologous nodes *03*, *030* to *033* have different values in the quadtrees of image *a* and image *b*, $d_{03}(a,b) = d_{030}(a,b) = d_{031}(a,b) = d_{032}(a,b) = d_{033}(a,b) = 1$. Nodes *030* to *033* do not exist in the quadtree of image *a*. The denominator is 9 because nine nodes *0*, *00* to *03* and *030* to *033* appear in the union of node identifiers of both quadtrees.

The following subsections present several approaches for storing similar images organized in quadtrees. The main goal of these approaches is to reduce the memory space used by image quadtrees by sharing common parts between them. The *Q*-distance can be used in these approaches to organize the images of the database: The smaller the *Q*-distance between image quadtrees, the greater the sharing of node values between their quadtrees. For instance, it is possible to organize the images in a hierarchy in order to maximize the sharing between a parent image and a child image. In this case, an image *i* is inserted into the hierarchy as a descendant of an image *j*, if for all image *k* of the hierarchy, $Q(i,j) \leq Q(i,k)$. See Manouvrier et al. (2002) for more details.

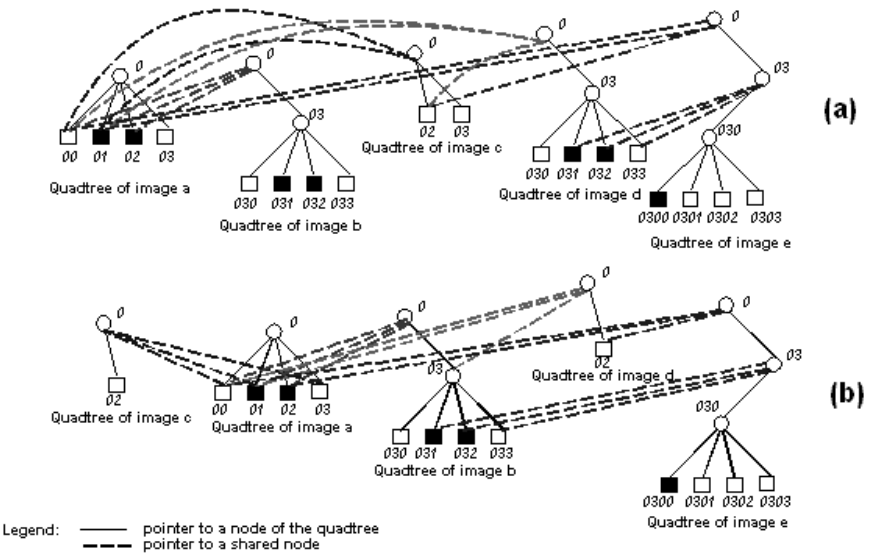
Overlapping Mechanisms

The technique of overlapping trees was initially presented by Burton et al. (1985) to manage the evolution of text files. The mechanism of overlapping has been extended to sequences of a given data structure, particularly to the hierarchical quadtrees by Vassilakopoulos et al. (1993) and linear quadtrees by Lin (1997a).

Vassilakopoulos et al. (1993) propose a technique of overlapping to represent sequences of similar binary raster images using hierarchical quadtrees. The quadtree of image *i* in the sequence overlaps the quadtree of image *(i-1)*: both quadtrees share identical parts (that is, homologous nodes with the same value). The shared nodes are referenced in the quadtree of *(i-1)* from the quadtree of *i*. When a leaf node has different values in two successive quadtrees, all the nodes appearing in the path, from the root to the non-shared node, are copied in the quadtree of image *i*. Each quadtree node has a reference counter that contains the number of quadtrees currently sharing the node. All nodes with a reference counter greater than 1, together with all descendants of such nodes, constitute shared information. As explained by Vassilakopoulos et al. (1993), this counter allows performing deletion of a particular quadtree from the overlapped family. An example of overlapped quadtrees is presented in Figure 4. Figure 4(a) represents the overlapping of quadtrees of Figure 2, using a linear order (from image *a* to image *e*). Figure 4(b) represents the same quadtrees overlapped using a *Q*-distance: The smaller the distance between quadtrees, the bigger the sharing. Thus, quadtrees *b* and *c* overlap quadtree *a*, quadtree *d* overlaps quadtree *b* and is overlapped by quadtree *e*.

The approach of Vassilakopoulos et al. (1993) has been extended to linear representations in Tzouramanis et al. (1998-2001). In these extensions, black

Figure 4. Quadtrees of Figure 2 overlapped using an (a) linear or (b) Q-distance based order



node identifiers are stored in a B-tree-based index: Tzouramanis et al. (1998-1999) use a classical B-tree (Bayer & McCreight, 1972), Tzouramanis et al. (2000) use a Multiversion B-tree (Becker et al., 1996) and Tzouramanis et al. (2001) use a Time-Split B-Tree (Lomet & Salzberg, 1989). Moreover, the overlapping mechanism is applied to the B-tree-based index tree, each index representing a linear quadtree. Thus, in Tzouramanis et al.(1998-2001), index trees are subject to the overlapping mechanism similar to the Overlapping Quadtrees.

Among the linear representations presented in the Linear Quadtrees for Binary Images section, only Lin (1997a, 1997b) proposes a procedure for coding a sequence of images by linear quadtrees, called *overlapped CBLQ code*; that is, a totally ordered sequence, with the goal of an efficient management of sequences of video images. In the overlapped CBLQ code, the first and last images in a sequence are coded using the CBLQ code (see the Linear Quadtrees for Binary Images section), and all images in the sequence, except the first one, are coded using an overlapped CBLQ representation. This overlapped representation encodes the differences between two successive images. The difference between two successive images *i* and *(i-1)* is obtained through a two-step procedure. Step 1: The first four nodes of quadtree of *i* and *(i-1)* are compared. If a node has the same value in both quadtrees, then it is coded by *A* in the overlapped coding of *i*. If homologous nodes are different, two cases appear: If one node is gray, then the code is *B*, otherwise the code is *C* or *D* according to the black or white value of each node (see Table 2). Step 2: A letter *A*, *C* or *D* codes each descendant of a node coded by *B* in step 1, according to their respective values in quadtrees of *(i-1)* and *i* (see Table 2). If a node *n* coded by a *B* in step 1 is not subdivided in quadtree *(i-1)* but is subdivided in quadtree *i*, then each value of nodes *nx* in quadtree *i* is compared with the value of node *n* in the quadtree *(i-1)*.

As an example, the coding of the five images *a*, *b*, *c*, *d* and *e*, represented in Figure 2, is:

0110	CBLQ code of image *a*
AAAB ADDA	CBLQ code of image *b* encoding the differences between *b* and *a*
AACB ACCA	Overlapped CBLQ code of *c* encoding the differences between *c* and *b*
AAAB ADDA	Overlapped CBLQ code of *d* encoding the differences between *d* and *c*
AAAB CAAA DAAA	Overlapped CBLQ code of *e* encoding the differences between *e* and *d*
0102 3110 1000	CBLQ code of image *e*

Table 2. Quadtree node coding by the overlapped CBLQ code of Lin (1997a)

Value of a node in the first quadtree:	White	White	Gray	Gray	Black	Black
Value of a node in the second quadtree:	Gray	Black	Black	White	White	Gray
Overlapped code:	C	D	C	D	C	D

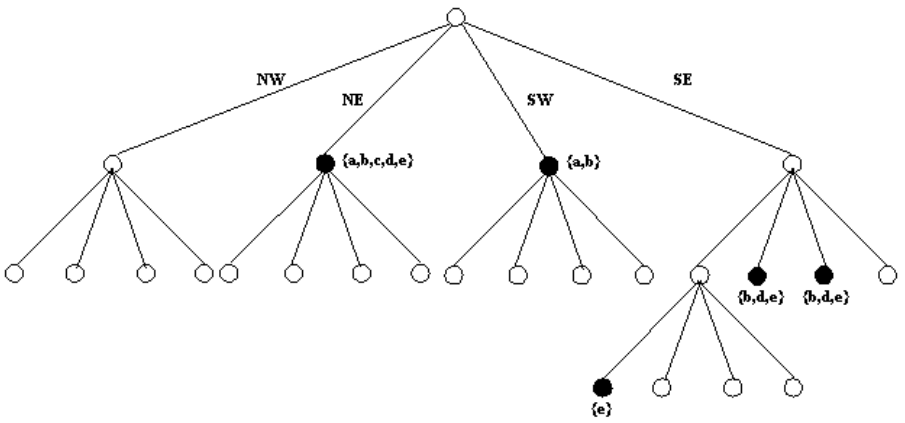
Inverted Quadtrees

When all images of a cluster are stored in a single quadtree, such a structure is called an *Inverted Quadtree*. This section presents three approaches of Inverted Quadtrees proposed by Cheiney and Touir (1991) and Vassilakopoulos and Manolopoulos (1995) for binary images, and by Jomier et al. (2000) for binary, grayscale or color images.

In an inverted representation, a quadtree node is associated with a set of image identifiers. An inverted quadtree representing the five images of Figure 2 appears in Figure 5. Each node *n* in the inverted quadtree is associated with either a set of image identifiers (black nodes in Figure 5), whose quadtree contains a black node *n*, or with an empty set (white nodes in Figure 5) if no image quadtree contains a black node *n*.

The *Fully* or *FI-Quadtr*ee of Cheiney and Touir (1991) consists of a full balanced quadtree. Each node holds a bit string of maximum length (the maximum number of images in the database), each bit designating a separate image. A black node *n* of an image quadtree *i* is identified by a *1* in the i^{th} bit of the bit string associated with the inverted quadtree node *n*. This structure is static: It can hold a predefined number of images. This number can be increased after a total reorganization of the structure. On the other hand, in the *Dynamic* or *DI-quadtree* of Vassilakopoulos and Manolopoulos (1995), each node of the inverted quadtree points to a list containing only identifiers of images that have the corresponding black node in their quadtree. The list is implemented by chained segments so that the structure is dynamic: Any number of images can be added.

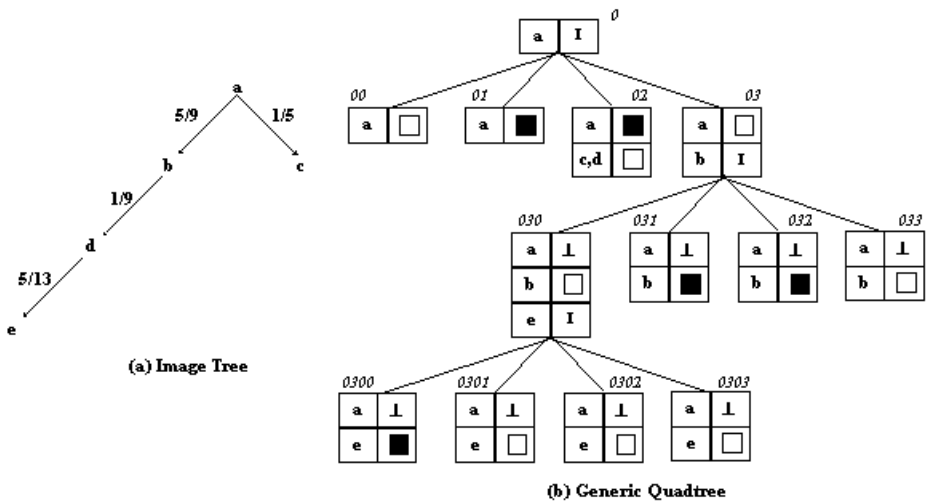
Figure 5. An inverted quadtree storing image quadtrees of Figure 2



The *Generic Quadtree* of Jomier et al. (2000) is an inverted structure that allows the management of binary, gray-scale or color images. Its nodes are called *generic nodes*. For each node appearing in an image quadtree, there is a generic node with the same identifier. A generic node *n* represents all nodes *n* of the quadtrees of images belonging to the cluster. Each generic node may be seen as a table with two columns and one or several lines. Each line *l* of a generic node *n* contains a list of image identifiers and a value *v* of quadtree node: *v* is the value of node *n* in each image quadtree whose identifier *i* appears in line *l*. A generic node can take any value in the following logical OR-sequence: ⊥, meaning that the node *does not exist* in the quadtree of images whose identifiers appear in the corresponding line (see generic node *030* in Figure 6b); OR *I*, meaning that the node is internal — it has four descendants (see generic node *0* in Figure 6b); OR *black* if it is a black leaf, OR *white* if it is a white leaf, and so forth.

To reduce the size of generic nodes, a cluster is organized in a tree called *Image Tree* (see Figure 6a), where an image *j* is inserted in the Image Tree as a descendant of an image *i* according to the *Q*-distance (see the Sharing Ratio Measurement section), and an *implicit* sharing is defined. The implicit sharing is based on the following rule: E*xcept if the identifier of an image i is explicitly associated with another value v, image i shares the value with its parent image*. Applying the sharing rule, all the nodes *n* of the quadtrees representing images that descend from image *i* implicitly share the value *v* (see generic node

Figure 6. Image tree and generic quadtree of images of Figure 2



00 in Figure 6b). This implicit sharing is stopped when a descendant image identifier appears in another line of generic node *n*, that is, it is associated with another value *v'* (see generic node *02* in Figure 6b — Only image quadtree *b* implicitly shares the value *black* with image quadtree *a*).

Quadtree-Based Image Retrieval

The quadtree structure is used in several Content-Based Image Retrieval approaches in order to capture the spatial composition of features in images (for example, color, texture, shape). This section presents approaches using quadtrees for content-based image retrieval. The first section presents approaches using quadtrees to index spatio-temporal databases storing consecutive historical binary raster images. The second subsection shows how the quadtree can be used as a *multi-level filtering* (or a *coarse-to-fine*) structure, and presents several approaches using quadtrees to query images using multilevel (or multi-precision) similarity matching.

Spatio-Temporal Query Processing

Quadtrees allow window query processing. As explained by Aref and Samet (1997), a window query is analogous to a range query in the spatial domain, where the result contains all the database objects overlapping a part of a space represented by a range of coordinates. Using a quadtree, a window query returns all the quadtree nodes overlapping a rectangular subregion termed a window. In Figure 7, for example, quadrants 1 to 4 are returned by the query window represented by the dotted lines. To find the query result, the window is decomposed into sub-windows; Aref and Samet (1997) have proposed a window decomposition algorithm. Tzouramanis et al. (1998-2001) apply window queries to consecutive historical raster binary images represented by overlapping linear quadtrees (stored in B-tree like index structures; see the Overlapping Mechanisms section). Each linear quadtree is associated with a unique timestamp. As explained by the authors, the proposed structure allows answering queries as "*find the black regions intersecting, or completely covering a window query, at each time point within a time interval*."

Content-Based Image Retrieval

Quadtree-Based Distances

To compare images represented by Quadtree-Based Feature Vectors (QFV – see the Quadtree-Based Image Feature Vector section), distances between images represented by quadtrees have been defined by Rukoz et al. (2002) and are summed up below.

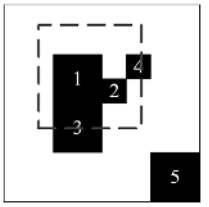
Let $\delta_n(i,j)$ be a normalized metric distance ($\delta_n(i,j) \in [0,1]$) between feature vectors contained in homologous nodes n of two QFV, i and j. $\delta_n(i,j)$ can be any geometric distance of the Minkowski L_p family[1] and can be a weighted distance. Let $\Delta(i,j)$ be the distance between two images i and j, represented by a quadtree-based feature vector. Δ-distance is defined as a weighted sum of normalized δ_n-distances between feature vectors stored in quadtree nodes n, weighted by coefficients w_n, $w_n \geq 0$:

$$\Delta(i,j) = \frac{\sum_n w_n \delta_n(i,j)}{\sum_n w_n}.$$

Δ-distance is normalized by the denominator $\sum_n w_n$ (at least one w_n must be different from zero): $\Delta \in [0,1]$. Values of w_n-coefficients depend on the user needs.

Particular cases of Δ-distance can be found in Kim and Kim (2000), Lin et al. (2001), Lu et al. (1994) and Malki et al (1999) (see the following subsection). The Q-distance, presented in the Sharing Ratio Measurement section, is a particular case of Δ-distance. Rukoz et al. (2002) defined two other particular cases of Δ-distance: $\Delta^{(p)}(i,j)$ and $\Delta_r(i,j)$. $\Delta^{(p)}(i,j)$ is a distance between image quadtrees i and j, without taking into account details after a certain quadtree level p: For all nodes n appearing from root level (level 0) to level p, $w_n>0$, and for all nodes n, appearing in a deeper level d ($d>p$), $w_n=0$. $\Delta_r(i,j)$ is a distance between image quadtrees i and j, where $w_n>0$ for all nodes n corresponding to the quadrants selected by the user and representing a region r. $w_n=0$ for the other quadtree nodes. This distance can be used to compute local or pattern search, which consists of retrieving images having regions similar to the query pattern given by the user. The user selects one or several quadrants in a grid image (4×4 or 16×16 grid) representing a region r that becomes the query pattern q_r: An image i is similar to the query pattern q_r if $\Delta_r(i,q_r)$ is below a given threshold α.

Figure 7. A binary image and query window, adapted from Tzouramanis et al. (2001)

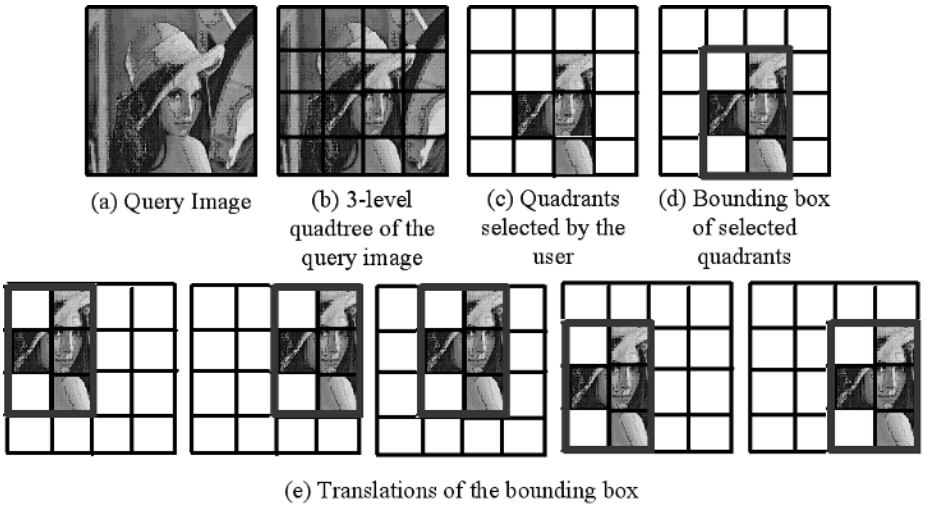


Quadtree-Based Feature Vector or Image Retrieval Approaches

The quadtree is used in several Content-Based Image Retrieval approaches, by De Natale and Granelli (2001), Kim and Kim (2000), Lu et al. (1994), Lin et al. (2001), Malki et al. (1999) and Ahmad and Grosky (1997, 2003), in order to capture the spatial distribution of features in images (for example, color, texture, shape or feature points). Lu et al. (1994) and Lin et al. (2001) represent each image of the database by a full fixed-depth balanced quadtree. Each node of the balanced quadtree contains a color histogram that characterizes the colors of the corresponding quadrant in the image. The authors define a distance between images based on the distance between the histograms of the sub-regions. They use a Δ-like distance, where w_n is a surface coefficient equal to 4^{-p} and δ_n is the color histogram distance between histograms stored in homologous nodes n appearing at level p. The top-level histograms are first compared. If they match within some threshold value, the next level will be searched, and so on. As explained by Lin et al. (2001), the distance metrics at higher precision levels provide a better discrimination power.

Malki et al. (1999) also represent each image by a full fixed-depth balanced quadtree. This approach allows performing region queries. Image signatures are computed on each sub-image and are stored in nodes of quadtree-based feature vectors. To perform a region query, the user has to specify the regions of interest at a chosen level. A bounding box is defined as the smallest sub-image containing the regions of interest (Figure 8). Each image of the database is compared to the initial query image and the query images obtained after a translation of the bounding box containing the regions of interest. The distance defined by the authors is a normalized linear combination of distances, and is a Δ_r-like distance. All chosen regions in the query image have the same size, thus all nodes n representing the regions selected by the user in the query image have the same w_n coefficient. The w_n-coefficient values are zero for the other nodes n which do not represent the selected regions.

Figure 8. An example of a query pattern, adapted from Malki et al. (1999)



In Kim and Kim (2000), a shape is divided into four sub-regions by two principal axes corresponding to the two eigenvectors at the center of the mass of the shape. Each sub-region is subdivided into four sub-regions in the same way. The sub-division process is repeated a predetermined number of times. A fixed-depth, balanced quadtree representation with its nodes corresponding to regions of the shape is derived from this process. Four parameters, invariant to translation, rotation and scale, are calculated for the corresponding regions of each node, while two parameters are extracted from the root node. The shape descriptor is represented as a vector of all parameters (reading the quadtree following a breadth first order). The similarity distance used to compare two shapes is a Δ-like distance where δ_n is a city block[2] distance (say L_1 or D_1) between quadtree nodes, and where w_n is equal for all nodes n.

In De Natale and Granelli (2001), each image of the database is represented by a quadtree-based color description described by two arrays: a binary array representing the quadtree structure and a label array representing the dominant color associated with each node. To be compared, two quadtrees representing images of the database must have an equivalent structure resulting from a warping procedure (see the Sharing Ratio Measurement section). When both quadtree structures are equivalent, a quadtree color similarity is computed, based on the difference between each dominant color stored in the quadtree nodes. The similarity between two images is evaluated according to both similarities (structure and color). The user queries images by selecting an image of the database or by drawing a sketch of the request color structures (that is, indicating

the positions and the values of dominant colors in a grid). The distance used is a combination of the QSS similarity, defined in the Sharing Ratio Measurement section, and Δ-like distance where $w_n>0$ for all leaf nodes and $w_n=0$ for all internal nodes.

In Ahmad and Grosky (1997,2003), each image of the database is represented by a set of representative feature points, extracted using image processing techniques. Unbalanced quadtrees are used to determine and represent the distribution of feature points in the images, the decomposition process stopping when each feature point is exactly contained in an image quadrant. To compare images, the authors define a Δ-like distance, where δ_n computes the difference between the proportion of feature points stored in each node *n*. Like the approaches presented above, images are filtered at each quadtree level, using a $\Delta^{(p)}$-like distance.

Summary and Future Trends

Originally proposed to index spatial databases, quadtrees are implemented in several commercial DBMS, such as DB2 (Adler, 2001) or Oracle (Kothuri et al., 2002). Subsequently, quadtrees have been adapted to represent images, and are implemented in content-based image retrieval research prototypes, such as DISIMA (Lin et al., 2001) or IKONA (Malki et al., 1999), and in the Virage Image Engine (Gupta & Jain, 1997). Quadtrees allow characterizing the spatial distribution of the information contained in images. This distribution is used to improve image storage or image retrieval algorithms. This chapter has presented several approaches that use quadtrees for image representation, image storage, image compression or content-based image retrieval. Table 3 gives a summary and classification of the approaches referenced in this chapter.

There is still much to be done in this domain to render quadtrees more efficient and flexible in image management systems. First, the approaches surveyed tailor quadtrees to meet the needs of specific applications (storage/compaction/compression or content-based retrieval). Efforts should be continued to yield more generic methods. Second, several methods optimize quadtree usage in a two-step procedure: The first step reduces the search space using another index method, such as extendible hash in Lin et al. (2001) or k-d-tree in Malki et al. (1999), and the second step computes distances using quadtrees. The index methods only use the quadtree from the root downwards. Thus, new approaches should be developed to allow accessing the quadtree at intermediate levels. Finally, another promising direction lies in taking advantage of the quadtree structure to extract image signatures, like in Ahmad and Grosky (2003), Albuz

Table 3. Summary of image quadtree-based approaches

Subject description	References
Quadtree definition	Brabec & Samet, 2003; Finkel & Bentley, 1974; Klinger et al., 1971; Samet, 1984, 1990
Binary image representation	Abel, 1984; Chang & Chang, 1994; Cheiney & Touir, 1991; Chen, 2002; Gargantini, 1982; Kawaguchi & Endo, 1989; Lin, 1997a, 1997b; Tzouramanis et al., 1998-2001; Vassilakopoulos et al., 1993-1995; Yang et al., 2000
Color image representation	Albuz et al., 2000; De Natale & Granelli, 2001; Lin et al., 2001; Lu et al., 1994, Malki et al., 1999
Texture representation	Malki et al., 1999; Smith & Chang, 1994
Shape representation	Chakrabarti et al., 2000; Kim & Kim, 2000
Quadtree-based feature vector	Kim & Kim, 2000; Lin et al., 2001; Lu et al., 1994; Malki et al., 1999
Image sequence or image cluster representation	Cheiney & Touir, 1991; Jomier et al., 2000; Lin,1997a; Manouvrier et al., 2002; Tzouramanis et al., 1998-2001; Vassilakopoulos et al., 1993-1995
Linear quadtree representation	Abel, 1984; Chang & Chang, 1994; Kawaguchi & Endo, 1990; Gargantini, 1982; Lin, 1997a; Tzouramanis et al., 1998-2001; Yang et al., 2000
Image compaction	Albuz et al., 2000; Baligar et al., 2003 ; Chang & Chang, 1994; Lin, 1997a, 1997b; Yang et al., 2000
Image compression	Jackson et al., 1997; Kim & Lee, 2002; Li et al., 1997 ; Strobach, 1991; Shusterman & Feder, 1994 ; Ramos & Hemami, 1996
Image management	Jomier et al., 2000; Lin, 1997b; Manouvrier et al., 2002; Yang et al., 2000
Quadtree-based distances	Ahmad & Grosky, 1997, 2003; De Natale & Granelli, 2001; Jomier et al., 2000; Kim & Kim, 2000; Lu et al., 1994; Lin et al., 2001; Malki et al., 1999; Rukoz et al., 2002
Image retrieval and image querying	Ahmad & Grosky, 1997, 2003; Albuz et al., 2000; Chakrabarti et al., 2000; Cheiney & Touir, 1991; De Natale & Granelli, 2001; Kim & Kim, 2000; Lin et al., 2001, Lu et al., 1994; Malki et al., 1999; Smith & Chang, 1994; Tzouramanis et al., 1998-2001

et al. (2000) or De Natale and Granelli (2001). Quadtree-based image signatures, generally of small size, are used in a first level of filtering process, improving the image retrieval performance by reducing the set of candidate images to be compared.

Acknowledgment

The authors would like to thank the referees of this chapter and Claudia Bauzer-Medeiros for their helpful comments and suggestions.

References

Abel, D.J. (1984). A B+tree structure for large quadtrees. *International Journal of Computer Vision, Graphics and Image Processing, 27*(1), 19-31.

Adler, D.W. (2001). DB2 Spatial extender - Spatial data within the RDBMS. *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001),* (pp. 687-690).

Ahmad, I., & Grosky, W.I. (1997). Spatial similarity-based retrievals and image indexing by hierarchical decomposition. *Proceedings of the International Database Engineering and Applications Symposium (IDEAS),* (pp. 269-278).

Ahmad, I., & Grosky, W.I. (2003). Indexing and retrieval of images by spatial constraints. *Journal of Visual Communication and Image Representation, 14*(3), 291-320.

Albuz, E., Kocalar, E.D., & Khokhar, A.A. (2000). Quantized CIELAb* space and encoded spatial structure for scalable indexing of large color image archives. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (4),* 1995-1998.

Aref, W.G. & Samet, H. (1997). Efficient window block retrieval in quadtree-based spatial databases. *GeoInformatica, 1*(1), 59-91.

Baligar, V.P., Patnaik, L.M., & Nagabhushana, G.R. (2003). High compression and low order linear predictor for lossless coding of grayscale images. *Image and Vision Computing, 21*(6), 543-550.

Bayer, R., & McCreight, E.M. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica, 1*(3), 173-189.

Becker, B., Gschwind, S., Olher, T., Seeger, B., & Widmayer, P. (1996). An asymptotically optimal multiversion B-tree. *The VLDB Journal, 5*(4), 264-275.

Brabec, F., & Samet, H. Spatial index demos. Retrieved February 13, 2004 from *www.cs.umd.edu/users/brabec/quadtree/index.html*

Burton, F.W., Huntbach, M.M., & Kollias, J.Y.G. (1985). Multiple generation text files using overlapping tree structures. *The Computer Journal, 28*(4), 414-416.

Chakrabarti, K., Ortega-Binderberger, M., Porkaew, K., Zuo, P., & Mehrotra, S. (2000). Similar shape retrieval in MARS. *Proceedings of the IEEE International Conference on Multimedia and Expo (II),* 709-712.

Chang, H.K., & Chang, J-W. (1994). The fixed binary linear quadtree coding scheme for spatial data. *The International Society for Optical Engineering SPIE, Visual Communications and Image Processing*, *2308*, 1214-1220.

Cheiney, J.P., & Touir, A. (1991). FI-Quadtree: A new data structure for content-oriented retrieval and fuzzy search. *Proceedings of the Second International Symposium on Spatial Databases (SSD'91)*, (pp. 23-32).

Chen, P-M. (2002). Variant code transformations for linear quadtrees. *Pattern Recognition Letters*, *23*(11), 1253-1262.

Cheng, H., & Li, X. (1996). On the application of image decomposition to image compression. *Proceedings of the Communications and Multimedia Security II: IFIP TC6/TC11 International Conference on Communications and Multimedia Security*, (pp. 116-127).

De Natale, F.G.B., & Granelli, F. (2001). Structured-based image retrieval using a structured color descriptor. *Proceedings of the International Workshop on Content-Based Multimedia Indexing (CBMI'01)*, (pp. 109-115).

Finkel, R.A., & Bentley, J.L. (1974). Quadtrees: A data structure for retrieval on composite keys. *Acta Informatica*, *4*, 1-9.

Gargantini, I. (1982). An effective way to represent quadtrees. *Communications of ACM*, *25*(12), 905-910.

Gupta, A., & Jain, R. (1997). Visual information retrieval. *Communication of. ACM*, *40*(5), 70-79.

ISO/IEC JTC1/SC29 WG1 (JPEG/JBIG) (1997). Lossless and near-lossless coding of continuous tone still images (JPEG-LS). Retrieved October 28, 2003, from *www.jpeg.org/public/fcd14495p.pdf*

Jackson, D.J., Mahmoud, W., Stapleton, W.A., & Gaughan, P.T. (1997). Faster fractal image compression using quadtree recomposition. *Image and Vision Computing*, *15*(10), 759-767.

Jomier, G., Manouvrier, M., & Rukoz, M. (2000). Storage and management of similar images. *Journal of the Brazilian Computer Society (JBCS)*, *3*(6), 13-26.

Kawaguchi, E., & Endo, T. (1980). On a method of binary picture representation and its application to data compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*(1), 27-35.

Kim, H-K., & Kim, J-D. (2000). Region-based shape descriptor invariant to rotation, scale and translation. *Signal Processing: Image Communication*, *16*(1-2), 87-93.

Kim, H-S., & Lee, J-Y. (2002). Image coding by fitting RBF-surfaces to subimages. *Pattern Recognition Letters, 23*(11), 1239-1251.

Klinger, A. (1971). Patterns and search statistics. *Optimizing methods in statistics* (pp. 303-337). New York: Academic Press.

Kothuri, R.K.V., Ravada, S., & Abugov, D. (2002). Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data. *Proceedings of the ACM SIGMOD International Conference on Management of Data, (SIGMOD 2002),* (pp. 546-557).

Li, X., Knipe, J., & Cheng, H. (1997). Image compression and encryption using tree structures. *Pattern Recognition Letters, 18*(11-13), 1253-1259.

Lin, S., Tamer Özsu, M., Oria, V., & Ng, R. (2001). An extendible hash for multi-precision similarity querying of image databases. *Proceedings of the 27th International Conference on Very Large DataBases (VLDB 2001),* (pp. 221-230).

Lin, T-W. (1997a). Compressed quadtree representations for storing similar images. *Image and Vision Computing, 15*(11), 833-843.

Lin, T-W. (1997b). Set operations on constant bit-length linear quadtrees. *Pattern Recognition, 30*(7), 1239-1249.

Lomet, D., & Salzberg, B. (1989). Access methods for multiversion data. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (pp. 315-324).

Lu, H., Ooi, B-C., & Tan, K-L. (1994). Efficient image retrieval by color contents. *Proceedings of the International Conference on Applications of Databases (ADB),* (pp. 95-108). Lecture notes in *Computer Science* 819, Springer-Verlag.

Malki, J., Boujemaa, N., Nastar, C., & Winter, A. (1999). Region queries without segmentation for image retrieval by content. *Proceedings of the Third International Conference on Visual Information Systems (Visual'99),* (pp. 115-122). Lecture Notes in Computer Science 1614.

Manouvrier, M., Rukoz, M., & Jomier, G. (2002). Quadtree representations for storage and manipulation of cluster of images. *Image and Vision Computing, 20*(7), 513-527.

Pennebaker, W.B., &. Mitchell, J.L. (1992). *JPEG: Still image data compression standard* (1st edition). Kluwer Academic Publishers.

Ramos, M.G., & Hemami, S.S. (1996). Edge-adaptive JPEG image compression. *Proceedings of the SPIE Symposium on Visual Communications and Image Processing, 27*(27), 100-110.

Rukoz, M., Manouvrier, M., &. Jomier, G. (2002). Distances de similarité d'images basées sur les arbres quaternaires. *Proceedings of the 18èmes*

Journées Bases de Données Avancées (BDA'02), (pp. 307-326).

Samet, H. (1984). The quadtree and related hierarchical structures. *ACM Computing Surveys*, *16*(2), 187-260.

Samet, H. (1990). *The design and analysis of spatial data structures*. Reading, MA: Addison Wesley.

Samet, H., & Webber, R.E. (1988). Hierarchical data structures and algorithms for computer graphics. Part I: Fundamentals. *IEEE Computer Graphics and Applications*, *8*(3), 48-68.

Shaffer, C.A., Samet, H., & Nelson, R.C. (1990). QUILT: A geographic information system based on quadtrees. *International Journal on Geographical Information Systems*, *4*(2), 103-131.

Shusterman, E., & Feder, M. (1994). Image compression via improved quadtree decomposition algorithms. *IEEE Transactions on Image Processing*, *3*(2), 207-215.

Smith, J.R., & Chang, S-F. (1994). Quad-tree segmentation for texture-based image query. *Proceedings of the Second Annual ACM International Conference on Multimedia*, (pp. 279-286).

Stewart, I.P. (1986). Quadtrees: Storage and scan conversion. *The Computer Journal*, *29*(1), 60-75.

Strobach, P. (1991). Quadtree-structured recursive plane decomposition coding of images. *IEEE Transactions on Signal Processing*, *SP-39*(6), 1380-1397.

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (1998). Overlapping linear quadtrees: A spatio-temporal access method. *Proceedings of the 6th ACM/International Symposium on Advances in Geographical Information Systems (ACM-GIS 98)*, (pp. 1-7).

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (1999). Processing of spatio-temporal queries in image databases. *Proceedings of the East-European Conference on Advances in Databases and Information Systems (ADBIS'99)*, (pp. 85-97).

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2000). Overlapping linear quadtrees and spatio-temporal query processing. *The Computer Journal*, *43*(4), 325-343.

Tzouramanis, T., Vassilakopoulos, M., & Manolopoulos, Y. (2001). Time-split linear quadtrees for indexing image databases. *Proceedings of the 8th IEEE International Conference on Image Processing (ICIP 2001)*, (pp. 733-736).

Vassilakopoulos, M., & Manolopoulos, Y. (1995). Dynamic inverted quadtree - A structure for pictorial databases. *Information Systems, special issue on Multimedia Information Systems*, *20*(6), 483-500.

Vassilakopoulos, M., Manolopoulos, Y., & Economou, K. (1993). Overlapping quadtrees for the representation of similar images. *Image and Vision Computing*, *11*(5), 257-262.

Yang, Y-H., Chung, K-L., & Tsai, Y-H. (2000). A compact improved quadtree representation with image manipulations. *Image and Vision Computing*, *18*(3), 223-231.

Endnotes

1 The L_p distance between two images *i* and *j*, represented by feature vectors $(i_1, i_2, ...i_n)$ and $(j_1, j_2, ...j_n)$, is defined by $L_p = \left[\sum_{k=1}^{n} |i_k - j_k|^p\right]^{\frac{1}{p}}$, p≥1.

2 The sum of the absolute difference of each parameters stored in homologous quadtree nodes.

Chapter V

Indexing Multi-Dimensional Trajectories for Similarity Queries

Michail Vlachos, IBM T.J. Watson Research Center, USA

Marios Hadjieleftheriou, University of California - Riverside, USA

Eamonn Keogh, University of California - Riverside, USA

Dimitrios Gunopulos, University of California - Riverside, USA

Abstract

With the abundance of low-cost storage devices, a plethora of applications that store and manage very large multi-dimensional trajectories (or time-series) datasets have emerged recently. Examples include traffic supervision systems, video surveillance applications, meteorology and more. Thus, it is becoming essential to provide a robust trajectory indexing framework designed especially for performing similarity queries in such applications. In this regard, this chapter presents an indexing scheme that can support a wide variety of (user-customizable) distance measures while, at the same

time, it guarantees retrieval of similar trajectories with accuracy and efficiency.

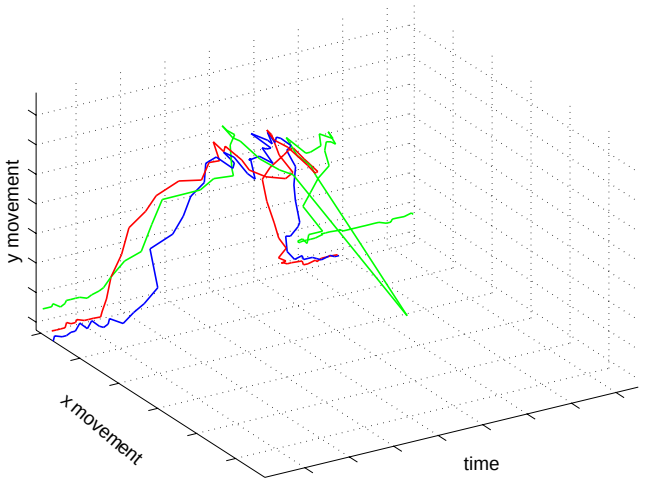
Introduction

Multi-dimensional trajectory data are prevalent in diverse fields of interest, such as environmental information systems, meteorology, traffic supervision, wireless telephony, video tracking (Betke, 2002), and so forth. In such applications it is very common to record multiple measurements concurrently (for example, spatio-temporal datasets that record the *x* and *y* positions of objects over time). Therefore, any dataset that involves storage of data with multiple attributes can be considered and treated as a set of multi-dimensional trajectories. An example of 2-dimensional trajectories appears in Figure 1. Recent advances in wireless communications, sensor devices and GPS technology have made it possible to collect large amounts of spatio-temporal data, increasing the interest in performing data mining tasks (Barbara, 1999; Roddick, 2000). Examples include tracking animals, recording weather conditions, gathering human motion data generated by tracking various body joints (Shimada, 2000; Arikan, 2002), tracing the evolution of migrating particles in biological sciences and more.

Some very important data mining operations for multi-dimensional trajectories involve the discovery of objects that *move similarly* or others that *follow closely a given query pattern*. An important consideration for these operations is the *similarity/distance measure* that will be used for discovering the most appropriate trajectories (for example, Euclidean distance). A major difficulty that affects the choice of a good similarity measure is the presence of noise (introduced due to electromagnetic anomalies, transceiver problems and so forth). Another obstacle is that objects may follow similar motion patterns (spatial domain) but at different rates (temporal domain). Hence, the similarity models should be robust to noise, and should support elastic and imprecise matches. For example, in Figure 1 all three trajectories follow similar paths; quantifying the similarity of these paths depends on the similarity measure that will be used. In addition, it is clear that existing outliers might distort the "expected" value of the similarity between the trajectories.

Some of the most widely used similarity measures are functions with large computational cost that make similarity query evaluation a challenging task. Consequently, in order to produce query results promptly, it is crucial to speed up the execution of these functions. For that purpose, low-cost *upper/lower-bounding* approximations of the similarity functions can be used initially to help prune most dissimilar trajectories faster.

Figure 1. Two-dimensional trajectories projected over time



Moreover, for very large datasets, in order to perform similarity retrieval efficiently, it is vital to organize intelligently the trajectories on secondary storage using an indexing scheme, in order to avoid the exhaustive linear scan over all the data. It is also essential, in that respect, to design indexing schemes that can support a wide variety of similarity measures, thus being adaptive to diverse application domains and flexible enough to satisfy varying user needs.

The rest of this chapter is organized as follows. First, various similarity measures for multi-dimensional trajectories, along with possible upper/lower-bounding techniques, are presented. Then, a multi-dimensional indexing scheme is discussed. Finally, we present test cases of real applications that utilize the trajectory indexing schemes for supporting similarity search queries.

Similarity/Distance Measures

As already mentioned, the most desirable properties of similarity/distance measures are: (i) To be robust to outliers; (ii) To support flexible time-aligned matches; and (iii) To have low computational cost. Some of these requirements are conflicting, making it hard to find similarity/distance functions that satisfy all three. In that regard, this section concentrates on three similarity/distance models for trajectories, each one satisfying a subset of the above properties.

The first and most widely used distance measure is the L_p-norm (for example, Euclidean distance). The second is an extension of *Dynamic Time Warping* (DTW) (Berndt, 1994) for higher dimensions. Finally, the third is a modification

of the *Longest Common Subsequence* (LCSS) (Levenshtein, 1966) specially adapted for continuous values.

The L_p-norm

Let A and B be 2-dimensional trajectories of length n, where $A = ((a_{x,1}, a_{y,1}), ..., (a_{x,n}, a_{y,n}))$ and $B = ((b_{x,1}, b_{y,1}), ..., (b_{x,n}, b_{y,n}))$ (sets of 2-dimensional points corresponding to one location per sampled time instant). For simplicity, for the rest of the chapter we consider only 2-dimensional sequences; extensions to more dimensions are straightforward.

The L_p-norm between A and B is defined as:

$$L_p(A,B) = \sum_{i=1}^{n} L_p((a_{x,i}, a_{y,i}),(b_{x,i}, b_{y,i})) .$$

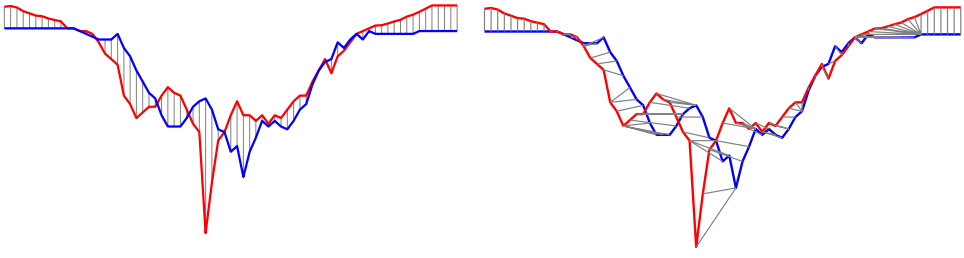
Essentially, the L_p-norm is the sum of the distances between the corresponding points in time, which gives a *measure of the distance* between the two trajectories. A 1-dimensional example is shown in Figure 2(a).

One shortcoming of this distance measure is that it cannot be defined for trajectories of different lengths. One can overcome this problem by sliding the shorter trajectory over the longest one and keeping the minimum computed distance overall, although, this results in increased computational cost. Another drawback of this model is that it cannot deal well with outliers (it degrades rapidly in the presence of noise). Finally, it is very sensitive to small distortions in the time axis — trajectories that follow very similar patterns but are slightly shifted in time will never be matched (Vlachos, 2002). However, it should be mentioned that the advantage of this simple metric is that it can be used to index trajectories, by utilizing dimensionality reduction techniques in conjunction with metric index structures (Agrawal, 1993; Faloutsos, 1994; Yi, 2000).

Dynamic Time Warping

Most real-world phenomena can evolve at varying rates. For that reason – even though the vast majority of research on trajectory/time-series data mining has focused on the Euclidean distance — for virtually all real-world systems there is a need for similarity/distance measures that allow time warping (that is, elastic matching in the time domain). For example, in molecular biology it is well

Figure 2. (a) The Euclidean distance (L_2-norm); and (b) the DTW distance



(a) *(b)*

understood that functionally related genes will express themselves in similar ways, but possibly at different rates (Aach, 2001; Bar-Joseph, 2002). For that reason DTW has been used extensively.

Let A and B be 2-dimensional trajectories of lengths n and m respectively. Also, let $Head(A) = ((a_{x,1},a_{y,1}),...,(a_{x,n-1},a_{y,n-1}))$ be the first n-1 points of A.

The DTW distance between A and B is:

$$DTW(A,B) = L_p((a_{x,n},a_{y,n}),(b_{x,m},b_{y,m})) + \min\begin{Bmatrix} DTW(Head(A),Head(B)) \\ DTW(Head(A),B) \\ DTW(A,Head(B)) \end{Bmatrix}.$$

The computation of DTW utilizes a dynamic programming technique. If the possible allowable matching in time is constrained within at most distance δ (the warping length) from each point, the computational cost of the algorithm is in the order of $O(\delta \cdot (n+m))$. This function gives a *measure of the distance* between two trajectories, as well. Figure 2(b) shows an example of DTW between two 1-dimensional trajectories.

The major drawback of this measure is that its efficiency deteriorates for noisy data. The algorithm matches individually all the points of a trajectory one by one. Thus, it also matches the outliers distorting the true distance. Another problem is that it is not suitable for use with most typical indexing techniques, since it violates the triangular inequality. One additional shortcoming is that it suffers from excessive computational cost for large warping lengths. Nevertheless,

restricting the allowed warping length substantially speeds up the computation of DTW and yields other benefits as well (Vlachos, 2003).

Longest Common Subsequence

An alternative similarity measure is LCSS, which is a variation of the edit distance (Levenshtein, 1966). The basic idea is to match two trajectories by allowing them to stretch along the time dimension without rearranging the order of the elements but allowing some of them to remain *unmatched*. The length of the resulting subsequence can be used as the similarity measure (Agrawal, 1995; Bollobas, 1997; Bozkaya, 1997; Das, 1997). Let A and B be 2-dimensional trajectories of lengths n and m respectively.

Given integers δ and ε, we define the *LCSS* distance between A and B as:

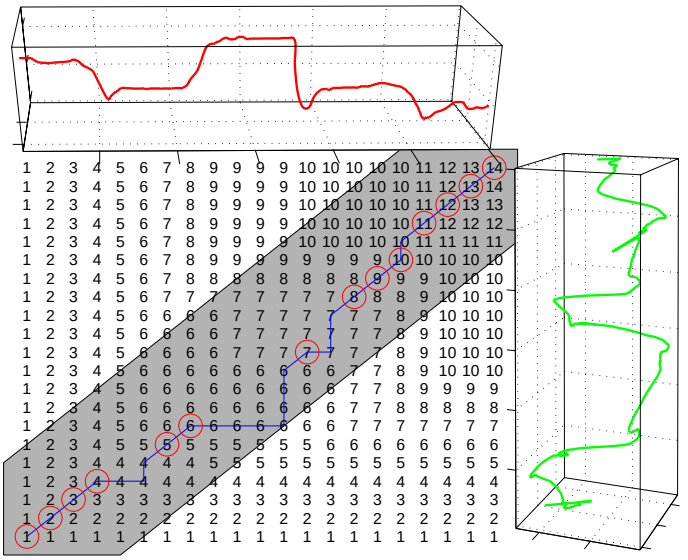
$$LCSS_{\delta,\varepsilon}(A,B) = \begin{cases} 0, & if\ A\ or\ B\ is\ empty \\ 1 + LCSS_{\delta,\varepsilon}(Head(A), Head(B)), & \\ \quad if\ |a_{x,n} - b_{x,m}| < \varepsilon\ and\ |a_{y,n} - b_{y,m}| < \varepsilon & \\ \quad and\ |n - m| \le \delta & \\ max(LCSS_{\delta,\varepsilon}(Head(A), B),\ LCSS_{\delta,\varepsilon}(A, Head(B))), & \\ \quad otherwise & \end{cases}.$$

The warping length δ controls the flexibility of matching in time and constant ε is the matching threshold in space. Contrary to the other two distance measures discussed so far, LCSS is a measure of similarity between two trajectories (not of distance).

This function can be computed efficiently using dynamic programming and has complexity in the order of $O(\delta \cdot (n+m))$, if only a matching window δ in time is allowed (Das, 1997). An instance of the dynamic programming execution between two trajectories is depicted in Figure 3, where the gray region indicates the allowed warping length.

The value of LCSS is unbounded and depends on the length of the compared trajectories. Thus, it needs to be normalized in order to support trajectories of variable length. The distance derived from the LCSS similarity can be defined as follows:

Figure 3. Execution of the dynamic programming algorithm for LCSS (the warping length is indicated by the gray area (δ=6))



The normalized distance $D_{\delta,\varepsilon}$ expressed in terms of the LCSS similarity between A and B is given by:

$$D_{\delta,\varepsilon}(A,B) = 1 - \frac{LCSS_{\delta,\varepsilon}(A,B)}{\min(n,m)}.$$

Even though LCSS presents similar advantages to DTW, it does not share its unstable performance in the presence of outliers. Nevertheless, this similarity measure is non-metric; hence, it cannot be directly utilized with most typical indexing schemes.

Multi-Dimensional Indexing

If an appropriate similarity/distance measure can be decided based on a specific application domain, similarity queries can be answered in a straightforward way using exhaustive search — that is, computing the similarity/distance function

between the query and every trajectory in the database. However, most trajectory datasets can reach the scale of Terabytes in size and, in addition, most popular similarity/distance functions can have a very large computational cost. Therefore, comparing a query to all the trajectories becomes intractable in such cases.

It is thus imperative to avoid examining the trajectories that are very distant to the query. This can be accomplished by discovering a close match as early as possible during the search, which can be used as a pruning threshold. A fast pre-filtering step can be employed that eliminates the majority of distant matches so that the costly, but accurate, similarity functions will be executed only for a small subset of qualifying trajectories (Keogh, 2002; Yi, 1998).

Various indexing schemes can be used to aid the pre-filtering step. For example, the trajectories can be approximated using Minimum Bounding Rectangles (MBRs) and then stored in a multi-dimensional index structure (Hadjieleftheriou, 2002). Alternatively, they can be represented as high-dimensional points and then stored in an index after using any dimensionality reduction technique (Agrawal, 1993).

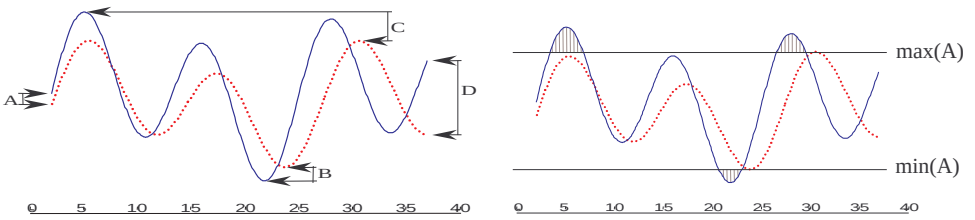
In order to improve query response times even further, much faster upper/lower-bounding functions of the actual similarity/distance measures should be derived (these are functions that consistently over/underestimate the true distance). This step is also necessary since some distance measures do not obey the triangular inequality; hence upper/lower-bounding functions that are metric are essential. These functions should guarantee no false dismissals. Intuitively, given an actual distance D of some trajectory to a query sequence, one should be able to prune other trajectories for which the less-expensive lower-bounds D_{LB} have already been computed and it holds that $D_{LB} > D$. If, in addition, upper-bounds D_{UB} of the distance functions can be computed, any trajectory having D_{LB} larger than the minimum computed D_{UB}, can also be pruned; its actual distance from the query is definitely larger than D_{UB}. (The inverse is true for *similarity* functions like LCSS.)

The next sections present various upper/lower-bounding techniques for the similarity/distance measures discussed so far. Then we show how these bounds can be used in combination with various indexing schemes as an efficient pre-filtering step.

Lower-Bounding the DTW

Most lower-bounding functions for DTW were originally defined only for 1-dimensional time-series (and are presented for the 1-dimensional case below). However, their extensions for a multi-dimensional setting are straightforward.

Figure 4. (a) Lower-bounding measure introduced by Kim (2001); (b) lower-bounding measure introduced by Yi (1998)



(a) *(b)*

The lower-bounding function introduced in Kim (2001) (hereafter referred to as LB-Kim) works by extracting a 4-tuple feature vector from each trajectory that consists of the first and last points of the trajectory, together with the maximum and minimum values. The maximum of the squared differences of the corresponding features is reported as the lower-bound. Figure 4(a) illustrates the idea.

The lower-bounding function introduced by Yi (1998) (hereafter referred to as LB-Yi) takes advantage of the observation that all the points in one trajectory that are larger (smaller) than the maximum (minimum) of the other trajectory must contribute at least the squared difference of their value and the maximum (minimum) value of the other trajectory to the final DTW distance. Figure 4(b) illustrates the idea. An important observation is that LB-Kim can be used as-is with any traditional multi-dimensional index structures. In contrast, LB-Yi cannot; it can be used only in conjunction with FastMap — a technique for approximate indexing of DTW (Faloutsos, 1995). The idea is to embed the trajectories into Euclidean space such that the distances between them are approximately preserved, and then use an index structure. The LB-Yi function is used to prune some of the inevitable false hits that will be introduced by FastMap.

A more robust lower-bounding technique was proposed by Keogh (2002) (hereafter referred to as LB-Keogh). Consider for simplicity a 1-dimensional time-series $A=(a_{x,1}, ..., a_{x,n})$. We would like to perform a very fast DTW match with warping length δ between trajectory A and query $Q=(q_{x,1}, ..., q_{x,n})$. Suppose that we replicate each point $q_{x,i}$ for δ time instants before and after time i. The surface that includes all these points defines the area of possible matching in time, between the trajectory and the query. Everything outside this area should not be matched. We call the resulting area around query Q the *Minimum Bounding Envelope* (MBE) of Q (Figure 5).

The notion of the bounding envelope can be extended for more dimensions; for example, the $MBE_\delta(Q)$ for a 2-dimensional query trajectory $Q=((q_{x,1}, q_{y,1}), ..., (q_{x,n}, q_{y,n}))$ covers the area between the following trajectories:

$$EnvLow \le MBE_\delta(Q) \le EnvHigh$$

where for dimension d at position i:

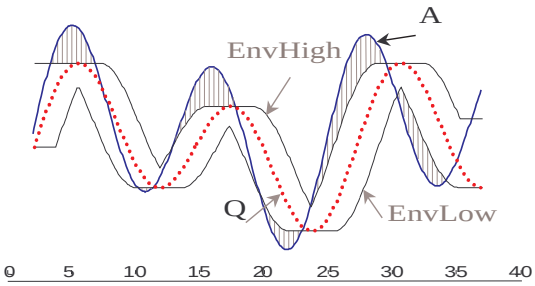
$$EnvHigh_{d,i} = max(q_{d,j}), \quad |i-j| \le \delta$$
$$EnvLow_{d,i} = min(q_{d,j}), \quad |i-j| \le \delta \ .$$

LB-Keogh works as follows: First, the MBE(Q) of query Q is constructed. Then, the distance of MBE(Q) from all other trajectories is evaluated. For trajectories Q and A (assuming dimensionality D), the distance between A and MBE(Q) is:

$$DTW(MBE(Q), A) = \sqrt{\sum_{d=1}^{D}\sum_{i=1}^{n} \begin{cases} (a_{d,i} - EnvHigh_{d,i})^2, & \text{if } a_{x,i} > EnvHigh_{d,i} \\ (a_{d,i} - EnvLow_{d,i})^2, & \text{if } a_{x,i} < EnvLow_{d,i} \\ 0, & \text{otherwise} \end{cases}} \ .$$

This function is the squared sum of the Euclidean distance between any part of A not falling within the envelope of Q and the nearest (orthogonal) edge of the envelope of Q, as depicted in Figure 5.

Figure 5. An illustration of LB-Keogh (the original trajectory Q (shown dotted), is enclosed in the MBE of EnvHigh *and* EnvLow*)*



Since the tightness of the bound is proportional to the number and length of the gray hatch lines, it is clear that this lower-bound provides a tighter approximation compared to LB-Kim or LB-Yi. In addition, this measure will always be at least as tight as LB-Yi, which is a special case arising when the warping length is allowed to cover the total length of the trajectory.

One can prove that LB-Keogh *lower-bounds* the actual time warping distance. A proof for the 1-dimensional case was given by Keogh (2002). This measure has since been used extensively in literature (Zhu, 2003; Rath, 2002; Vlachos, 2003).

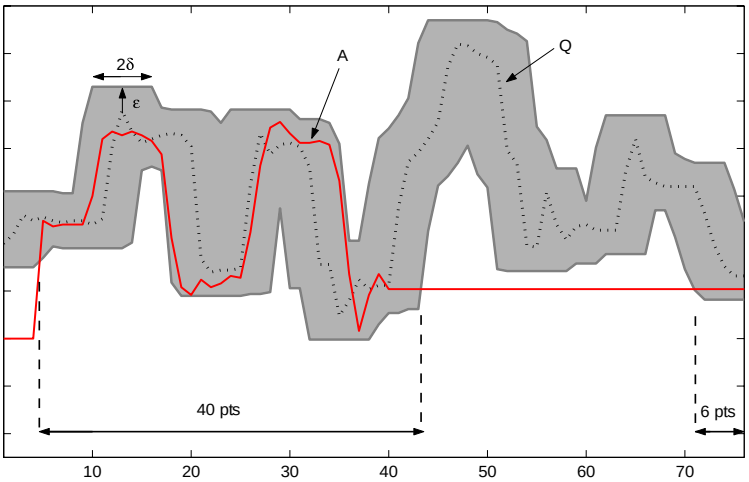
Upper-Bounding the LCSS

We would like to perform a very fast LCSS match with warping length δ and within space ε between trajectory A and query $Q = (q_{x,1}, ..., q_{x,n})$. The query MBE in this case should be extended within ε in space, above and below, in order to incorporate this parameter as well (Figure 6).

The LCSS similarity between the envelope of Q and a trajectory A is defined as:

$$LCSS(MBE(Q), A) = \sum_{i=1}^{n} \begin{cases} 1, & \text{if A[i] within envelope} \\ 0, & \text{otherwise} \end{cases}$$

Figure 6. MBE within δ in time and ε in space of a trajectory (everything that lies outside this envelope should not be matched)



This value represents an *upper-bound* for the *similarity* of Q and A. We can use the MBE (Q) to compute a *lower-bound* on the *distance* between trajectories as well:

Lemma 1. For any two trajectories Q and A the following holds:

$$D_{\delta,\varepsilon}(MBE(Q), A) \leq D_{\delta,\varepsilon}(Q, A)$$

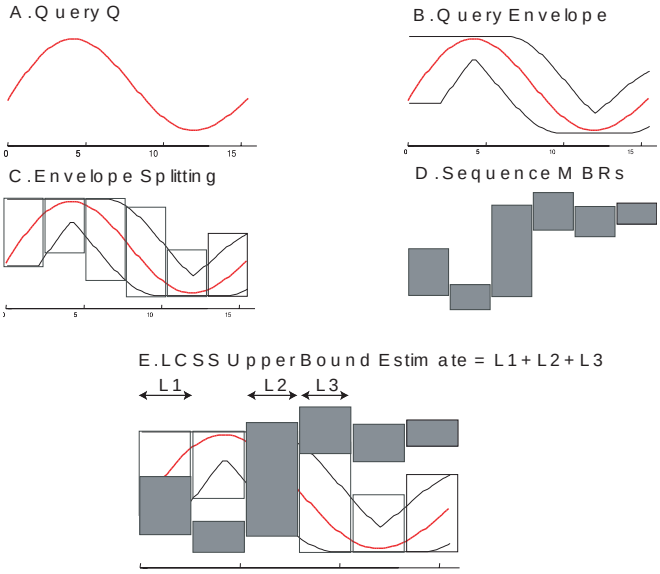
Using Upper/Lower-Bounds for Quick Trajectory Pruning

Previous sections described various ways for upper/lower-bounding the similarity/distance measures between two trajectories. According to the GEMINI framework (Agrawal, 1993), the approximated distance functions can be used to create an index that guarantees no false dismissals. However, the described upper/lower-bounds need to be computed using the raw trajectory data. This section presents a robust multi-dimensional indexing scheme for trajectories (which appeared in Vlachos, 2003) that can be used with any similarity/distance measure discussed so far and that does not need to access the actual trajectory data but only their compact approximations.

This scheme is based on the following principles. First, the trajectories are approximated using a number of multi-dimensional MBRs which are then stored in an R-tree (Guttman, 1984). For a given query Q, MBE(Q) is computed and decomposed into smaller MBRs. The resulting query MBRs are probed as range searches in the R-tree to discover which trajectory approximations intersect with Q and could be potential candidates. Trajectories that are very distant are never accessed and, thus, they are instantly pruned. Access to the raw trajectory data is restricted to only a few candidates. This procedure is schematically illustrated in Figure 7.

This index is very compact since it stores only the substantially reduced in size trajectory approximations, and its construction time scales well with the trajectory length and the dataset cardinality. Therefore, this method can be utilized for massive data mining tasks. One of the significant advantages of this approach is its generality and flexibility. The user is given the ability to pose queries of variable warping length without the need to reconstruct the index. By adjusting the width of the bounding envelope on the query, the proposed method can support L_p-Norms and constrained or full warping. Also, the user can choose between faster retrieval with approximate solutions, or exact answers on the expense of prolonged execution time. Other work on multi-dimensional trajec-

Figure 7. A query is extended into a bounding envelope and approximated with MBRs (Overlap between the query and the trajectory MBRs stored in the index suggests possible matches.)



tories can support only the use of the Euclidean distance (Lee, 2000; Kahveci, 2001).

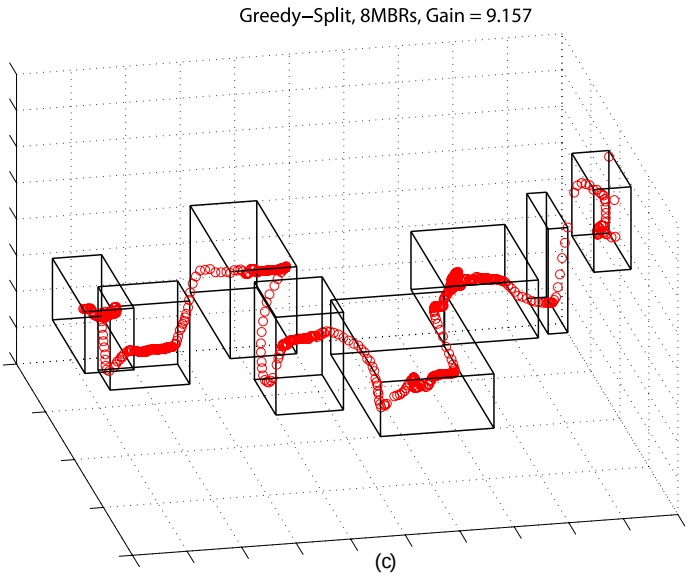
Generating Efficient Trajectory Approximations

The trajectories are indexed using a multi-dimensional R-tree. Before being inserted into the index, they need to be approximated using multi-dimensional MBRs. Approximating each trajectory with only one MBR introduces a lot of empty space and deteriorates index performance substantially, even though it reduces to a minimum the required amount of space for building the index structure. Alternatively, it is much more beneficial to use finer object approximations by decomposing each trajectory into multiple MBRs (Figure 8). It can be proven that:

Lemma 2. Minimizing the volume of the trajectory approximations minimizes the expected similarity approximation error.

Clearly, the best approximation of a trajectory (or an MBE) using a fixed number of MBRs is the set of MBRs that completely contains the trajectory and

Figure 8. Approximating a trajectory with 8 MBRs



minimizes the volume consumption. Various approaches can be considered for finding the best approximation possible. A detailed analysis appears in Hadjieleftheriou (2002).

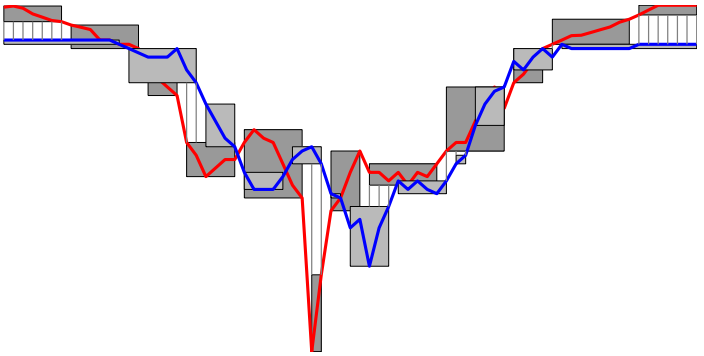
Upper/Lower-Bounds Using MBR Approximations

Suppose that an index with the MBR trajectory approximations has already been built and the user provides a query Q. The goal is to discover the k closest trajectories to Q according to the given similarity measure. The pre-filtering step should compute the upper/lower-bound estimates of the similarity/distance between the query and the MBRs of the indexed trajectories. Since MBR approximations are used instead of the raw data, the upper/lower-bounds of previous sections cannot be directly applied on this indexing scheme; special upper/lower-bounding functions have to be derived. Based on the geometric properties of the trajectory MBRs and their intersections, estimates of the L_p-norm, DTW and LCSS similarity measures can be obtained.

Lower-Bounds for L_p-Norm

Suppose that each trajectory P is decomposed into a number of MBRs. The i-th MBR of P consists of the numbers: $M_{P,i} = \{(tl, th) \mid (l_1, h_1), ..., (l_D, h_D)\}$ (considering

Figure 9. Lower-bound of the Euclidean distance when MBR approximations are used



D-dimensional trajectories). Given MBRs $M_{T,i}$ and $M_{R,j}$, belonging to objects *T* and *R* respectively, we define:

$$\bigcap_t (M_{T,i}, M_{R,j}) = \| \textit{Intersection} \|_t$$

the intersection of the two MBRs in the time dimension.

When any L_p-norm is used as a similarity measure, first the query is split into a number of MBRs and then the distances between the query MBRs and the indexed trajectory MBRs are evaluated. For that purpose function MINDIST between MBRs M_Q and M_P is defined:

$$MINDIST(M_Q, M_P) = \sqrt{\sum_{2 \le d \le D} \bigcap_t (M_Q, M_R) \times x_d^2}\ , \text{ where}$$

$$x_d = \begin{cases} \max(h_{Q,d}, h_{P,d}) - min(l_{Q,d}, l_{Q,d}), & if\ (h_{Q,d} > h_{P,d} \quad \& \quad l_{Q,d} > l_{P,d}) \\ & or\ (h_{P,d} > h_{Q,d} \quad \& \quad l_{P,d} > l_{Q,d}) \\ max \begin{cases} max(h_{Q,d}, h_{P,d}) - max(l_{Q,d}, l_{P,d}) \\ min(h_{Q,d}, h_{P,d}) - min(l_{Q,d}, l_{P,d}) \end{cases}, & otherwise \end{cases}$$

The total L_p-norm distance (Figure 9) is the sum of the MINDISTs between all query and trajectory MBR boundaries that intersect in time.

Lower-Bounds for DTW

To compute the lower-bound for the DTW, the following steps are required. First, the query envelope is computed. Then, the envelope is split into multiple MBRs. Finally, the MINDIST function is used to calculate the distance of the query MBRs from the trajectory MBRs contained in the index.

The computation is practically the same as for the L_p-norm. The only difference is that an envelope is applied on the query first. An example is shown in Figure 10.

Upper-Bounds for LCSS

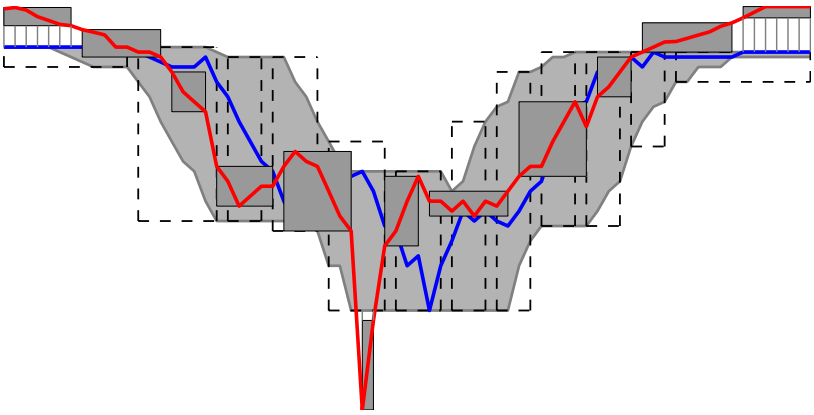
An upper-bound estimate for the LCSS is the following:

$$L(P) = \sum_m \sum_n M_{Q,m} \bigcap\nolimits_t M_{P,n}$$

where Q is a query and P is trajectory. This estimate computes the intersection *in time* of all query MBRs intersecting with any of P's MBRs.

Using these bounds, one can retrieve good candidates from the index that will help eliminate most similarity/distance function evaluations.

Figure 10. Lower-bounded DTW distance when the query envelope and a data trajectory are split into a number of MBRs



Test Cases

This section presents some test cases of real applications for the similarity indexing techniques discussed in this chapter.

Automatic Transcription of Handwritten Manuscripts

The Library of Congress contains more than 54 million manuscripts, and there is an imperative need for preserving these historical documents (Figure 11). Storing these manuscripts in image format is unrealistic, since only the conversion to text would result in more than 20 Terabytes of data.

Therefore, there is an increasing interest to automatically transcribe these documents. However, optical character recognition techniques (OCR) cannot be fruitfully utilized in this case due to the handwritten nature of these documents and the degradations of the paper medium.

Recently, new techniques have been proposed that combine trends from pattern recognition and time-series similarity in order to achieve this difficult task (Rath, 2002). The basic idea is that one can first identify the different words in a manuscript, manually annotate a subset of them, and then automatically classify the remaining words.

The process involves the following steps: (i) From a manuscript image, images of the different words are extracted (word spotting); (ii) Writing variations (slant/skew/baseline) are corrected for the extracted words; (iii) Features for

Figure 11. Part of a George Washington manuscript in the Library of Congress

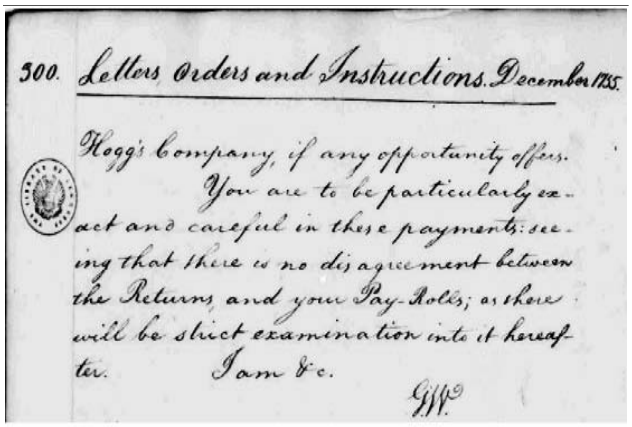
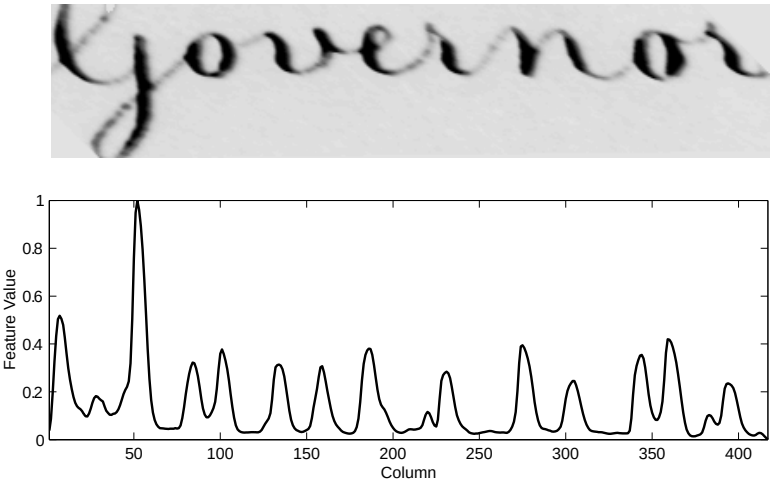
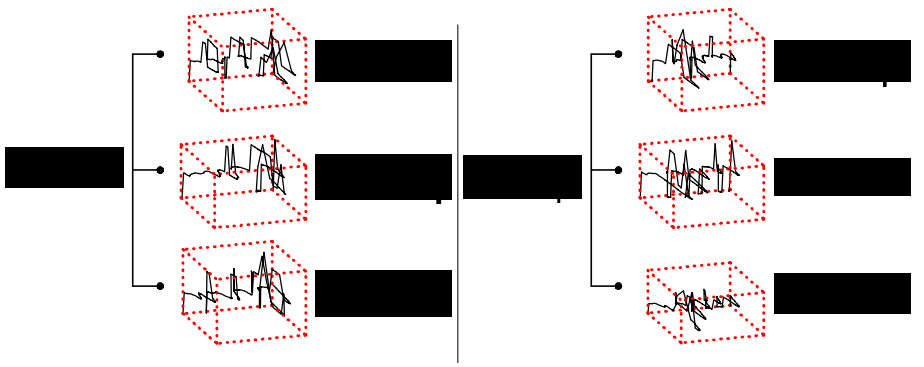


Figure 12. Top: the word 'governor' extracted from a George Washington manuscript; bottom: one possible time-series feature extracted from every column of the image (sum of intensity values)



each image are extracted. Such features can include the sum of intensity values per column, or the ink/paper transitions per column etc. (Figure 12); (iv) Manual annotations of a word subset are produced; and (v) Classification of the remaining words based on the annotated ones is performed.

Figure 13. Manuscript word annotation (3-nearest-neighbor matches based on the extracted time-series features and utilizing multi-dimensional DTW as the distance measure)



The features that are extracted are stored as multi-dimensional time-series. The annotation process can then use the multi-dimensional extensions of DTW or LCSS measures. In Figure 13 we see the results of a 3-Nearest-Neighbor search for a variety of words, produced using the techniques discussed in this chapter.

Similarity Search in Motion-Capture Data

The popularity of motion capture, or *mocap*, in CG movies and three-dimensional computer games has motivated the development of many techniques to tackle the laborious problem of motion search and motion editing. The existence of large libraries of human motion capture has resulted in a growing demand for content-based retrieval of motion sequences without using annotations or other metadata. Using the notions described in this chapter, one can address efficiently the issue of rapidly retrieving perceptually similar occurrences of a particular motion in a long mocap sequence or unstructured mocap database for the purpose of replicating editing operations with minimal user-input. One or more editing operations on a given motion are made to affect all similar matching motions (for example, change all walking motions into running motions, and so forth).

The first step of motion-editing consists of a similarity search portion, where using a *query-by-example* paradigm the animator first selects a particular motion by specifying its start and end time, and the system searches for similar occurrences in a mocap database (Figure 14). For maximum usability, the mocap matching engine must provide fast response to user queries over extended unlabeled mocap sequences, whilst allowing for spatial and temporal deviations in the returned matches.

Figure 14. Matches to the user query motion are returned using an efficient search index, previously extracted from the motion database

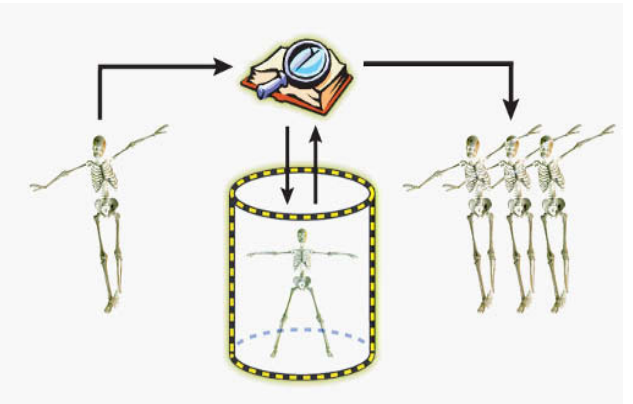
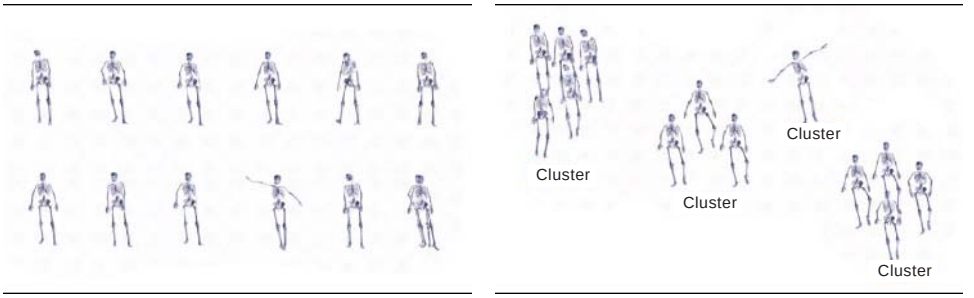


Figure 15. Some matching results in the motion-capture database for the query: "Find all walking motions" (The final results can be grouped into similar motions using a hierarchical clustering algorithm and presented to the animator in a more meaningful manner.)



Using the techniques described in this chapter, each motion is split up in multi-dimensional MBRs and stored in an index structure (Cardle, 2003). Support for LCSS and DTW similarity/distance measures caters for noisy motion with variations in the time axis. Therefore one can find matches independent of the speed at which the mocap data were recorded.

The animator has the crucial ability to interactively select the body areas utilized in the matching, so that, for example, all instances of a walking motion are returned, irrespective of the upper body motion. The results of such a query are shown in Figure 15. Finally, in the case where many potential matches exist, the query results can be clustered and presented in a more organized manner, allowing the animator to rapidly dismiss undesirable classes of matches.

Conclusions

This chapter presented a robust multi-dimensional trajectory indexing scheme for similarity queries. Pertinent similarity measures were discussed, along with various upper/lower-bounding techniques for speeding up query evaluation. A fast pre-filtering step with the use of R-trees, aiming at pruning a large number of non-qualifying trajectories based on MBR approximations, was elaborated. Finally, two real applications were discussed as test cases to testify for the utility and efficiency of the presented techniques.

References

Aach, J., & Church, G. (2001). Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17, 495-508.

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. *Proceedings of the Fourth FODO*, (pp. 69-84).

Agrawal, R., Lin, K., Sawhney, H.S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling and translation in time-series databases. *Proceedings of the VLDB*, (pp. 490-501).

Arikan, O., & Forsyth, D. (2002). Interactive motion generation from examples. *Proceedings of the ACM SIGGRAPH*, (pp. 483-490).

Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T., & Simon, I. (2002). A new approach to analyzing gene expression time series data. *Proceedings of the Sixth RECOMB*, (pp. 39-48).

Barbara, D. (1999). Mobile computing and databases: A survey. *IEEE TKDE*, 108-117.

Berndt, D., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, 359-370.

Bollobas, B., Das, G., Gunopulos, D. & Mannila, H. (1997). Time-series similarity problems and well-separated geometric sets. *Proceedings of the 13th SCG*, (pp. 243-307).

Bozkaya, T., Yazdani, N., & Ozsoyoglu, M. (1997). Matching and indexing sequences of different lengths. *Proceedings of the CIKM*, (pp. 128-135).

Cardle, M., Vlachos, M., Brooks, S., Keogh, E., & Gunopulos, D. (2003). Fast motion capture matching with replicated motion editing. *The 29th ACM SIGGRAPH, Sketches and Applications.*

Das, G., Gunopulos, D., & Mannila, H. (1997). Finding similar time series. *Proceedings of the PKDD Symposium*, (pp. 88-100).

Faloutsos, C., & Lin, K.-I. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *Proceedings of the ACM SIGMOD*, (pp. 163-174).

Faloutsos, C., Ranganathan, M., & Manolopoulos, I. (1994). Fast subsequence matching in time series databases. *Proceedings of the ACM SIGMOD*, (pp. 419-429).

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of the SIGMOD*, (pp. 47-57).

Hadjieleftheriou, M., Kollios, G., Tsotras, V., & Gunopulos, D. (2002). Efficient indexing of spatiotemporal objects. *Proceedings of the EDBT*, (pp. 251-268).

Kahveci, T., & Singh, A.K. (2001). Variable length queries for time series data. *Proceedings of the IEEE ICDE*, (pp. 273-282).

Kahveci, T., Singh, A., & Gurel, A. (2002). Similarity searching for multi-attribute sequences. *Procedings of the SSDBM*, (pp. 175-184).

Keogh, E. (2002). Exact indexing of dynamic time warping. *Proceedings of the VLDB*, (pp. 406-417).

Kim, S., Park, S., & Chu, W. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. *Proceedings of the ICDE*, (pp. 607-614).

Lee, S.-L., Chun, S.-J., Kim, D.-H., Lee, J.-H., & Chung, C.-W. (2000). Similarity search for multidimensional data sequences. *Proceedings of the ICDE*, (pp. 599-608).

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, *10*(10), 707-710.

Rath, T., & Manmatha, R. (2002). Word image matching using dynamic time warping. *Technical Report MM-38*. Amherst: Center for Intelligent Information Retrieval, University of Massachusetts Amherst.

Roddick, J.F., & Hornsby, K. (2000). International workshop on temporal, spatial and spatio-temporal data mining.

Shimada, M., & Uehara, K. (2000). Discovery of correlation from multi-stream of human motion. *Proceedings of Discovery Science*, (pp. 290-294).

Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., & Keogh, E. (2003). Indexing multi-dimensional time-series with support for multiple distance measures. *Proceedings of the SIGKDD*, (pp. 216-225).

Vlachos, M., Kollios, G., & Gunopulos, D. (2002). Discovering similar multidimensional trajectories. *Proceedings of the ICDE*, (pp. 673-684).

Yi, B.-K., & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary Lp norms. *Proceedings of the VLDB*, (pp. 385-394).

Yi, B.-K., Jagadish, H.V., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. *Proceedings of the ICDE*, (pp. 201-208).

Zhu, Y., & Shasha, D. (2003). Warping indexes with envelope transforms for query by humming. *Proceedings of the SIGMOD*, (pp. 181-192).

Section III

Query Processing and Optimization

Chapter VI

Approximate Computation of Distance-Based Queries[1]

Antonio Corral, University of Almeria, Spain

Michael Vassilakopoulos
Technological Educational Institute of Thessaloniki, Greece

Abstract

In spatial database applications the similarity or dissimilarity of complex objects is examined by performing distance-based queries (DBQs) on data of high dimensionality (a generalization of spatial data). The R-tree and its variations are commonly cited as multidimensional access methods that can be used for answering such queries. Although the related algorithms work well for low-dimensional data spaces, their performance degrades as the number of dimensions increases (dimensionality curse)*. To obtain acceptable response time in high-dimensional data spaces, algorithms that obtain approximate solutions can be used. In this chapter, we review the most important approximation techniques for reporting sufficiently good results quickly. We focus on the design choices of efficient approximate DBQ algorithms that minimize the response time and the number of I/O operations over tree-like structures. The chapter concludes with possible future research trends in the approximate computation of DBQs.*

Introduction

The term "Spatial Database" refers to a database that stores data for phenomena on, above or below the earth's surface, or in general, various kinds of multidimensional entities of modern life, for example, the layout of a Very Large Scale Integration (VLSI) design. In a computer system, the spatial data are represented by points, line segments, regions, polygons, volumes and other kinds of geometric entities and are usually referred to as spatial objects. For example, a spatial database may contain polygons that represent building footprints from a satellite image, or points that represent the positions of cities, or line segments that represent roads.

The key characteristic that makes a spatial database a powerful tool is its ability to manipulate spatial data, rather than simply storing and representing them. One of the main targets of such a database is answering queries related to the spatial properties of data. Some typical spatial queries are the following.

- A "Point Location Query" seeks for the objects that fall on a given point (for example, the country where a specific city belongs).
- A "Range Query" seeks for the objects that are contained within a given region, usually expressed as a rectangle or a sphere (for example, the taxis that are inside the historic center of a city).
- A "Join Query" may take many forms. It involves two or more spatial datasets and discovers pairs (or tuples in the case of more than two datasets) of objects that satisfy a given spatial predicate, such as *overlap* (for example, the pairs of boats and stormy areas for boats sailing across a storm).
- Finally, a "Nearest Neighbor Query" seeks for the objects residing more closely to a given object. In its simplest form, it discovers one such object (the Nearest Neighbor). Its generalization discovers K such objects (K Nearest Neighbors), for a given K (for example, the K ambulances closer to a spot where an accident with K injured persons occurred).

As extensions of the above "fundamental" spatial queries, new query forms have emerged. For example, to examine the similarity or dissimilarity of large sets of complex objects, high-dimensional feature vectors are extracted from them and organized in multidimensional indexes. The most important property of this feature transformation is that the feature vectors correspond to points in the multidimensional Euclidean space (a kind of generalized spatial data). Then, DBQs are applied on the multidimensional points.

The multidimensional access methods belonging to the R-tree family, the R*-tree (Beckmann, Kriegel, Schneider, & Seeger, 1990) and particularly the X-tree

(Berchtold, Kiem, & Kriegel, 1996), are considered good choices for indexing high-dimensional point datasets in order to perform DBQs. This is accomplished by branch-and-bound algorithms that employ distance functions and pruning heuristics based on Minimum Bounding Rectangles (MBRs) in order to reduce the search space. The performance of these algorithms degrades as the number of dimensions increases *(dimensionality curse)*, and the effects of this curse are widely analyzed by Berchtold, Böhm and Kriegel (1998). However, it can be improved if the search space is restricted somehow. For practical purposes, in many situations approximate results that can be obtained significantly faster than the precise ones are usually as valuable. Another possible direction for reducing the cost of DBQs, sacrificing a limited part of accuracy, is to use heuristic search techniques on R-trees, such as local search, simulated annealing and genetic algorithms.

In this chapter, we review the most important DBQ-related approximation techniques that can be applied on R-tree-like structures (widely accepted spatial access methods). We focus on the design choices of efficient approximate DBQ algorithms that minimize the response time and the number of I/O operations while reporting sufficiently good results. The chapter concludes with possible future research trends in the approximate computation of DBQs.

Distance Based Queries

Distance functions are typically based on a distance metric (satisfying the non-negativity, identity, symmetry and triangle-inequality properties) defined on points in space. A general distance metric is called L_t-distance (L_t), L_t-metric or Minkowski distance between two points, $p = (p_1,p_2,\ldots,p_d)$ and $q = (q_1,q_2,\ldots,q_d)$, in the d-dimensional space, $\Re^d$, and it is defined as follows:

$$L_t(p,q) = \left(\sum_{i=1}^{d} |p_i - q_i|^t \right)^{1/t}, \quad if \quad 1 \le t < \infty \qquad \text{and}$$

$$L_\infty(p,q) = \max_{1 \le i \le d} |p_i - q_i|, \quad if \quad t = \infty .$$

For t = 2 we have the Euclidean distance and for t = 1 the Manhattan distance. These are the most known L_t-metrics in the spatial database context. Often, the Euclidean distance is used as the distance function but, depending on the

application, other distance functions may be more appropriate (Laurini & Thomson, 1992).

In spatial database applications, distance metrics are used in several kinds of DBQs. The most representative and known distance-based queries in the spatial database framework are the following (Hjaltason & Samet, 1998; Corral, Manolopoulos, Theodoridis, & Vassilakopoulos, 2000; Chan, 2001; Böhm & Krebs, 2002; Corral et al. 2003; Shou, Mamoulis, Cao, Papadias, & Cheung, 2003):

(1) δ-distance range query. It involves one spatial dataset, a query point and a distance threshold δ. The answer is a set of spatial objects from the input dataset that are within distance δ from the query point.

(2) K nearest neighbors query (K-NNQ). It involves one spatial dataset and a query point. It discovers the K distinct objects from the input dataset that have the K smallest distances from the query point.

(3) δ-distance range join query (buffer query). It involves two spatial datasets and a distance threshold δ. The answer is a set of pairs of spatial objects from the two input datasets that are within distance δ from each other.

(4) Iceberg distance joins. It involves two spatial datasets, a distance threshold δ and a cardinality threshold K. The answer is a set of pairs of objects from the two input datasets that are within distance δ from each other, provided that the first object appears at least K times in the join result.

(5) K closest pairs query (K-CPQ). It involves two spatial datasets and a cardinality threshold K. It discovers the K distinct pairs of objects from the two input datasets that have the K smallest distances between them.

(6) K nearest neighbors join. It involves two spatial datasets and a cardinality threshold K. The answer is a set of pairs from the two datasets that includes, for each of the objects of the first dataset, the pairs formed with each of its K nearest neighbors in the second dataset.

(7) K-Multi-way distance join query. A *K*-Multi-way distance join query involves n spatial datasets, a query graph (a weighted directed graph that defines directed itineraries between the n input datasets) and a cardinality threshold *K*. The answer is a set of *K* distinct n-tuples (tuples of n objects from the n datasets obeying the query graph) with the *K* smallest $D_{distance}$-values (the $D_{distance}$ function is a linear function of distances of the n objects that make up this n-tuple, according to the edges of the query graph).

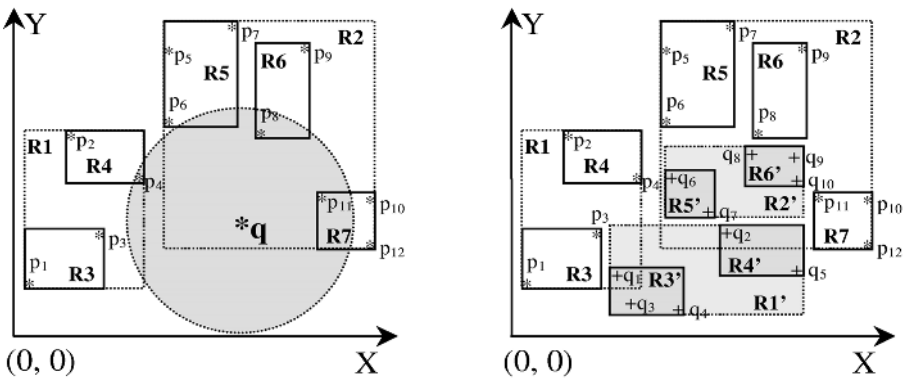
DBQs are very useful in many applications that use spatial data for decision making and other demanding data-handling operations. For example, in the case of *K* closest pairs query, the first dataset may represent the cultural landmarks

of the United States, while the second set may contain the most populated places of North America. The answer of this kind of query will consist of the *K* closest pairs of cities and cultural landmarks, providing an order to the authorities for the efficient scheduling of tourist facilities creation and so forth. The *K* value could be dependent on the budget of the authorities allocated for this purpose. In this chapter, we will focus on *K*-NNQ and *K*-CPQ to study the design of approximate algorithms. For the completeness of presentation, Figure 1 illustrates examples of the DBQs that we are going to study in this chapter (*K*-NNQ and *K*-CPQ). In the left part of the figure, a set of 2-dimensional points organized according to an R-tree structure (MBR-based index) is depicted. Suppose that we want to find the three nearest neighbors (3-NNs) to the query point q. It is easy to discover that the 3-NNs are (p_{11}, p_8, p_4). On the other hand, in the right part of the figure, two different sets of 2-dimensional points (organized according to two different R-tree structures, one depicted with non-colored MBRs and another depicted with shaded MBRs) are own. If we want to obtain the three closest pairs of points (3-CPs) of the two sets, then the result is ((p_8, q_8), (p_{11}, q_{10}), (p_4, q_6)).

R-Tree Background

Branch-and-bound has been the most successful technique for designing algorithms that answer queries on tree structures. Lower and upper bounding functions are the basis of the computational efficiency of branch-and-bound algorithms. Numerous branch-and-bound algorithms (incremental and non-

Figure 1. Examples of sets of 2-dimensional points and their MBRs for the computation of the K-NNQ (left) and the K-CPQ (right)



incremental) for spatial queries using spatial access methods have been studied in the literature (Hjaltason & Samet, 1998; Corral et al., 2000). The fundamental assumption used in the DBQ algorithms is that the spatial datasets are indexed by structures of the R-tree family. The R-tree and its variants (Gaede & Günther, 1998) are considered excellent choices for indexing various kinds of spatial data (points, line-segments, rectangles, polygons and so forth) and have already been adopted in commercial systems, for example, Informix (Brown, 2001) and Oracle (Oracle Technology Network, 2001).

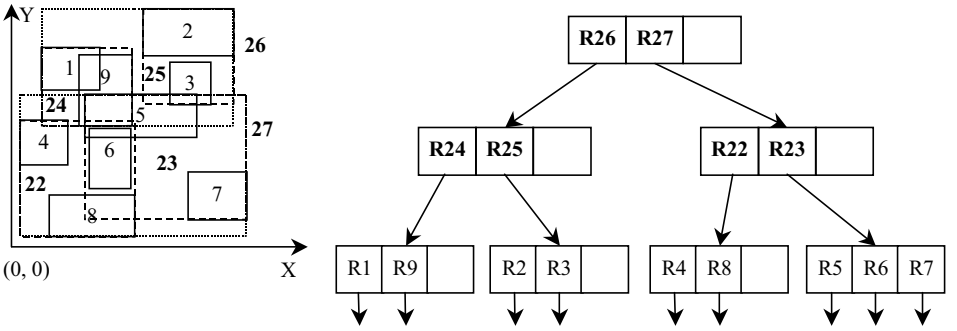
R-trees are hierarchical, height-balanced multidimensional data structures designed for secondary storage, and they are generalizations of B-trees for multidimensional data spaces. They are used for the dynamic organization of d-dimensional objects represented by their Minimum Bounding d-dimensional hyper-Rectangle (MBR). An MBR is determined by two d-dimensional points that belong to its faces, one that has the d minimum and one that has the d maximum coordinates (these are the endpoints of one of the diagonals of the MBR). Each R-tree node corresponds to the MBR that contains its children. The tree leaves contain pointers to the objects of the database instead of pointers to children nodes. The nodes are implemented as disk pages.

Figure 2 depicts some rectangles on the left and the corresponding R-tree on the right. Dotted lines denote the bounding rectangles of the subtrees that are rooted in inner nodes.

Like other tree-like index structures, an R-tree index partitions the multidimensional space by grouping objects in a hierarchical manner. A subspace occupied by a tree node in an R-tree is always contained in the subspace of its parent node, that is, the *MBR enclosure property*. According to this property, an MBR of an R-tree node (at any level except the leaf level) always encloses the MBRs of its descendent R-tree nodes. This characteristic of spatial containment between MBRs of R-tree nodes is commonly used by spatial join algorithms, as well as by distance-based query algorithms. Note that the MBR and the MBR enclosure property are *fundamental approximation mechanisms*. At the leaf level, an MBR approximates the area covered by the objects it encloses, while at internal levels an MBR approximates the area covered by the subtrees of the corresponding node. Another important property of the R-trees is the *MBR face property*. This property says that every face of any MBR of an R-tree node (at any level) touches at least one point of some spatial object in the spatial database. This characteristic is used by distance-based query algorithms.

Many variations of R-trees have appeared in the literature. A survey was authored by Gaede & Günther (1998). One of the most popular and efficient variations is the R*-tree (Beckmann et al., 1990). The R*-tree added two major enhancements to the original R-tree, when a node overflow is caused. First, rather than just considering the area, the node-splitting algorithm in R*-trees also

Figure 2. An example of an R-tree



minimizes the perimeter and overlap enlargement of the MBRs. Second, an overflow node is not split immediately, but a portion of entries of the node is reinserted from the top of the R*-tree (forced reinsertion). On the other hand, the X-tree (Berchtold et al., 1996) is another variation that avoids splits that could result in a high degree of overlap of MBRs in the internal R*-tree nodes. Experiments (Berchtold et al., 1996), showed that X-tree outperforms R*-tree for point query and nearest neighbor queries in high-dimensional data spaces (that is, when the space dimensionality is larger than, say, 16).

Corral et al. (2000) presented a generalization of the function that calculates the minimum distance between points and MBRs (MINMINDIST). Let M(A, B) represent an MBR in d-dimensional space, where A = (a_1, a_2, ..., a_d) and B = (b_1, b_2, ..., b_d), such that $a_i \le b_i$, for $1 \le i \le d$, are the endpoints (d-dimensional points) of one of its major diagonals. Given two MBRs M_1(A, B) and M_2(C, D), *MINMINDIST(M_1, M_2)* is defined as follows:

$$MINMINDIST(M_1, M_2) = \sqrt{\sum_{i=1}^{d} y_i^2},$$

$$y_i = \begin{cases} c_i - b_i, & if\ c_i > b_i \\ a_i - d_i, & if\ a_i > d_i \\ 0, & otherwise \end{cases}.$$

We can apply this distance function to pairs of any kind of elements (MBRs or points) stored in R-trees during the computation of branch-and-bound algorithms to reduce the search space (in the pruning process). The most important

properties of MINMINDIST are the following: (1) If any of the two or if both MBRs degenerateto a point or to two points, then we obtain the minimum distance between a point and an MBR or between two points. (2) *Lower-bound property*: for each pair (p, q) of multidimensional points enclosed by a pair of MBRs (M_p, M_q), it holds that MINMINDIST(M_p, M_q) ≤ MINMINDIST(p, q). (3) *MINMINDIST is monotonically non-decreasing with the R-tree levels*. For a given pair of MBRs (M_p, M_q) enclosed by another pair of MBRs (M'_p, M'_q), where M'_p covers M_p and M'_q covers M_q, it holds that: MINMINDIST(M'_p, M'_q) ≤ MINMINDIST(M_p, M_q).

The general pruning heuristic for DBQs over R-trees is the following: *"if MINMINDIST(M_1, M_2) > z, then the pair of MBRs (M_1, M_2) will be discarded"*, where *z* is the threshold distance value; for example, it is the *K*-th nearest neighbor (*K*-NNQ) or the *K*-th closest pair (*K*-CPQ) that have been found so far.

Approximate Computation Techniques

Approximate solutions are important in the spatial query processing context, because such solutions can provide good bounds of the optimum result (sacrificing accuracy) and, in practice, they can be obtained by users much faster than precise techniques. Designing good approximate algorithms is not an easy task, because these algorithms must fully exploit the inherent properties of the data structure and yet keep the computation time as short as possible. Most of the known approximate algorithms can be classified into one of the two following approximate strategies (Clarkson, 1994; Arya, Mount, Netanyahu, Silverman, & Wu, 1998; Zezula, Savino, Amato, & Rabitti, 1998, Papadias, Mantzourogiannis, Kalnis, Mamoulis, & Ahmad, 1999; Ioannidis & Poosala, 1999; Ciaccia & Patella, 2000; Chakrabarti, Garofalakis, Rastogi, & Shim, 2000; Papadias & Arkoumanis, 2002; Corral, Cañadas, & Vassilakopoulos, 2002a): search space reduction and heuristic search techniques.

For the *search space reduction*, when it is not computationally feasible to enumerate all the possible cases, we may systematically enumerate only a partial fraction using different methods. The computational cost of an algorithm can be reduced if the search space is restricted by some means, at the possible expense of the accuracy of the result. The *heuristic search techniques* are methods that provide good performance (with high probability on the average, as shown by experimentation) but cannot be guaranteed always to provide the best solution (for example, local search, simulated annealing or genetic algorithms). In the

next two sections, we present these two families of strategies and discuss the resulting approximate techniques.

Search Space Reduction

This section presents the most representative search space approximation techniques:

- ε-approximate method (Arya et al., 1998; Ciaccia & Patella, 2000). Given any positive real e ($\varepsilon \geq 0$) as maximum distance relative error to be tolerated, the result of a DBQ (*K*-NNQ or *K*-CPQ) is $(1 + \varepsilon)$-approximate if the distance of its i-th item is within relative error e (or a factor $(1 + \varepsilon)$) of the distance of the i-th item (point for *K*-NNQ or pair of points for *K*-CPQ) of the exact result of the same DBQ, $1 \leq i \leq K$. For example, an $(1+\varepsilon)$-approximate answer to the *K*-CPQ is an ordered sequence (the result is sorted in ascending order of distance) of *K* distinct pairs of points $((p_1', q_1'), (p_2', q_2'), ..., (p_K', q_K')) \in (P \times Q)^K$, such that (p_i', q_i') is the $(1 + \varepsilon)$-approximate closest pair of the i-th closest pair (p_i, q_i) of the exact result $((p_1, q_1), (p_2, q_2), ..., (p_K, q_K)) \in (P \times Q)^K$, that is $(dist(p_i', q_i') - dist(p_i, q_i)) / dist(p_i, q_i) \leq \varepsilon$, $1 \leq i \leq K$. When the pruning heuristic is applied on a branch-and-bound algorithm, an item *X* (pair of MBRs or points) is discarded if $MINMINDIST(X) > z/(1+\varepsilon) \Leftrightarrow MINMINDIST(X) + (MINMINDIST(X)*\varepsilon) > z \Leftrightarrow MINMINDIST(X) > z - (MINMINDIST(X)*\varepsilon)$. The decrease of *z* (distance value of the *K*-th closest pair found so far during the processing of the algorithm) depends on the value of *MINMINDIST(X)* multiplied by ε (an unbounded positive real). Note that for $\varepsilon = 0$, the approximate algorithm behaves like the exact version and outputs the precise solution.
- α-allowance method (Corral et al., 2002a). When the pruning heuristic is applied, an item *X* is discarded if $MINMINDIST(X) + \alpha(z) > z$, where *z* ($z \geq 0$) is the current pruning distance value (for example, the distance of the *K*-th nearest neighbor found so far, for *K*-NNQ) and $\alpha(z)$ is an allowance function. In order to apply this approximate method, $\alpha(z)$ is assumed to satisfy that $\alpha(z) \geq 0$ for all *z* and $z_1 \leq z_2$ implies that $z_1 - \alpha(z_1) \leq z_2 - \alpha(z_2)$. Typical forms of $\alpha(z)$ are: $\alpha(z) = \beta$ (β is a non-negative constant) and $\alpha(z) = z*\gamma$ (γ is a constant with $0 \leq \gamma \leq 1$). For the second case, $MINMINDIST(X) + (z*\gamma) > z \Leftrightarrow MINMINDIST(X) > z - (z*\gamma) \Leftrightarrow MINMINDIST(X) > z*(1 - \gamma)$. In this case, the decrease of *z* depends on γ (positive real in the interval [0, 1]). We obtain the exact algorithm when $\gamma = 0$.
- N-consider method (Corral et al., 2002a). In this case, the approximate branch-and-bound algorithm only considers a specified portion, or percent-

age N ($0<N\leq1$) of the total number of items examined by the respective exact algorithm, when visiting an internal (N_I) or leaf (N_L) node for K-NNQ, or a pair of internal (N_I) or leaf (N_L) nodes for K-CPQ. With this approximate technique, we try not to waste time computing distances that will not improve the approximate result significantly. The exact version of the algorithm is obtained when $N = 1$ ($N_I = N_L = 1$).

- Time-consider method (Corral et al., 2002b). With this technique, the algorithm retrieves the best possible (exact or approximate) result within a given total processing time threshold (*total_time*), that is, the algorithm is stopped at the time *total_time*, reporting the result found so far. We will obtain the exact version of the algorithm when *total_time* = ∞. We are going to consider this an approximate technique, since the response time is an important parameter to be controlled in DBQs. Note that this method is not a native search-space reduction method, since it reduces search space by side effect: Since the computation is not completed, the space is not exhaustively searched.

The previous MBR-based distance function, pruning heuristic and approximation techniques (based on the search space reduction, can be embedded into approximate branch-and-bound algorithms for DBQs that are applied over indexes belonging to the R-tree family and obtain sufficiently good approximate results quickly. In order to design approximate branch-and-bound algorithms for processing *K*-NNQs or *K*-CPQs in a non-incremental way (*K* must be known in advance), an extra data structure that holds the *K* nearest neighbors or closest pairs, respectively, is necessary. This data structure, called *K-heap*, is organized as a maximum binary heap (Corral et al., 2000). It is possible to apply the four approximation techniques to incremental and non-incremental branch-and-bound algorithms (iterative or recursive) for DBQs (*K*-NNQ or *K*-CPQ) using tree-like structures belonging to the R-tree family, although we will only apply them to the non-incremental ones. (The incremental approximation alternatives for *K*-NNQ and *K*-CPQ using R-trees can be obtained in a straightforward manner by adapting the non-incremental approaches so as to follow Best-First search.) In the following (due to space limitations), only one such non-incremental application is presented. More details appear in Corral et al. (2002a) and Corral, Cañadas, and Vassilakopoulos(2002b).

For the ε-approximate method ($\varepsilon \geq 0$), we are going to present an iterative (characterized by I) and non-incremental approximate branch-and-bound algorithm (following a Best-First search) for *K*-NNQ (KNNI) between a set of points P stored in an R-tree (R_p) and a query point q (*z* is the distance value of the *K*-th nearest neighbor found so far and at the beginning $z = \infty$). For the iterative approach, we need a minimum binary heap, *H*, of references to nodes of the R-

tree <MINMINDIST, $Addr_p$>, which are kept ordered by increasing values of MINMINDIST; the item with the minimum MINMINDIST is visited first. The algorithm can be described by the following steps:

KNNI1 Start from the root of the R-tree and initialize the minimum binary heap *H*.

KNNI2 If you access an internal node, then calculate MINMINDIST for each possible MBR M_i. Insert into the minimum binary heap *H* those references to nodes of MBRs having MINMINDIST(M_i, q) $\leq z / (1+\varepsilon)$.

KNNI3 If you access a leaf node, then calculate MINMINDIST(p_i, q) between q and each possible point (p_i) stored in the node. If this distance is smaller than or equal to *z*, then remove the root of the *K-heap* and insert the new point p_i, updating this structure and *z*.

KNNI4 If the minimum binary heap *H* is empty, then stop.

KNNI5 Get the item on top of the minimum binary heap <MINMINDIST, $Addr_p$>. If this item has MINMINDIST $> z / (1 + \varepsilon)$, then stop. Else, repeat the algorithm from **KNNI2** for this item.

The main advantage of the recursive branch-and-bound algorithms is that they transform the global problem into smaller local ones at each tree level and we can apply pruning heuristics over every subproblem for reducing the search space. Moreover, for improving the I/O and CPU cost of the recursive branch-and-bound algorithm for *K*-CPQ, two techniques are used. The first improvement aims at reducing the number of I/O operations, and it consists in a *Global LRU buffer*. The second enhancement aims at reducing the CPU cost by using the *distance-based plane-sweep technique* (Corral et al., 2000) to avoid processing all possible combinations of pairs of R-tree items from two internal or leaf nodes.

The performance of the resulting space reduction algorithms, as well as the trade-off between efficiency of each algorithm and accuracy of the result, have been studied by Corral et al. (2002a and 2002b). The parameters γ and N (varying in the range [0, 1]) can act as adjusters of this trade-off. The case of the parameter ε is different, since it is an unbounded positive real ($\varepsilon \geq 0$) and the users cannot directly control the accuracy of the result. The ε-approximate method could be considered a method that allows controlling of this trade-off, only if suitable upper bounds are determined for the parameter ε. Such bounds depend on the distance distribution and the dimensionality. Since the main target is to be able to control the above trade-off effectively and easily, the α-allowance technique (as a distance-based approximate method) is used for the development of the hybrid approximate scheme that follows.

The main problem of the approximate algorithms for DBQs using R-trees in high-dimensional data spaces is *not to waste time computing distances that will not significantly improve the result* (Ciaccia & Patella, 2000). In this context we propose to combine the approximation techniques (N-consider, α-allowance and Time-consider) in a recursive branchand-bound algorithm in order to improve the balance between the cost and the accuracy of the result. Such a combination would be interesting, because the N-consider technique (structure-based approximate method) is appropriate for reducing the response time and the distance computations, whereas the α-allowance technique (distance-based approximate method) is recommended for obtaining a good accuracy of the result (Corral et al., 2002a, 2002b). Moreover, the Time-consider method is suggested for the users to decide the total time consumed by the algorithm. On the other hand, the recursive algorithm explores the search space in a Depth-First order, finding many approximate solutions (although, it may take a very long time to obtain the best solution if it does not traverse the search path in the right direction) and improving their qualities along time.

This combination consists of three consecutive filters (refinements in the pruning process) at internal levels of the R-trees. In the first filter, we use the Time-consider method to establish the total processing time of the algorithm. For the second filter, we adopt the N-consider approximation technique (based on the structure), producing a set of candidates. Each candidate is then examined by the third filter, using the α-allowance approximation technique (based on distance). At the leaf level, since we try to avoid distance computations, which will not significantly improve the result, N-consider and Time-consider are adopted as approximation techniques. As an example, the recursive (characterized by R) and non-incremental hybrid (characterized by H) approximate branch-and-bound algorithm for processing *K*-CPQ (for *K*-NNQ, the KNNR algorithm is similar) between two sets of points (P and Q) indexed in two R-trees (R_P and R_Q) with the same height can be described by the following steps (z is the distance value of the *K*-th closest pair found so far and at the beginning $z = \infty$):

KCPRH1 Start from the roots of the two R-trees.

KCPRH2 If you access two internal nodes and the consumed time is larger than *total_time*, then stop. Else, choose only a portion *Total'* = N_I*Total of all possible pairs of MBRs (Total) stored in the nodes, and apply the *distance-based plane-sweep technique* over Total' in order to obtain *ENTRIES'*, the reduced set of pairs of candidate entries. Propagate downwards recursively only for those pairs of MBRs from *ENTRIES'* having MINMINDIST(M_i, M_j) $\leq z * (1 - \gamma)$.

KCPRH3 If you access two leaf nodes and the consumed time is larger than *total_time*, then stop. Else, choose only a portion *Total'* = N_L*Total of all possible pairs of points (Total) stored in the nodes, and apply the *distance-based plane-sweep technique* over Total' in order to obtain *ENTRIES'*, the reduced set of pairs of candidate entries. For those pairs of MBRs from *ENTRIES'* having MINMINDIST(M_i, M_j) smaller than or equal to *z*, then remove the root of the *K-heap* and insert the new pair of points (p_i, q_j), updating this structure and *z*.

If we want to obtain the exact solution of *K*-CPQ, we can run the KCPRH algorithm using $N_I = N_L = 1.0$, $\gamma = 0.0$ and *total_time* = ∞. Moreover, applying the same idea over the previous KCPI algorithm, it would be straightforward to combine the same approximate techniques (N-consider, α-allowance and Time-consider) in an iterative branch-and-bound algorithm in order to design a hybrid iterative approximate algorithm (KCPIH). For *K*-NNQ (KNNI algorithm) the combination is similar.

Heuristic Search Techniques

In this section we present heuristic techniques (local search, simulating annealing and genetic algorithms) that can be combined with tree-based spatial indexing (R-trees) in order to reach good enough, but not necessarily optimal, solutions for DBQs quickly.

Local Search

Local search or iterative improvement methods start with a random solution called *seed*, and then try to reach a local optimum by performing uphill moves (that is, by visiting neighbors with higher similarity). When they reach a local optimum (from which further uphill moves are not possible) they restart the same process from a different seed until the time threshold is exceeded. Algorithms based on this general concept have been successfully employed for a variety of problems (Papadias et al., 1999, 2001; Papadias & Arkoumanis, 2002).

In the approximation context, local search is applied to improve the quality of initial approximate solutions obtained by some means. The general scheme is to repeat a search for a better solution in the neighborhood *N(x)* of a given approximate solution *x*. Various algorithms can be designed depending on how neighborhood *N(x)* is defined, and *N(x)* can be specified either explicitly or

implicitly (that is, by an algorithm to compute the elements in *N(x)*). The general scheme is described as the following steps (*input:* an approximate solution *x*; and *output:* an improved approximate solution *x*):

LS1 Given an *x*, test whether there is a feasible solution with a better value of a cost function in *N(x)*.

LS2 If yes, then set x to represent the improved solution and return to *LS1*. Otherwise, output *x* as an approximate optimal solution and halt.

In Papadias and Arkoumanis (2002), an *Indexed Local Search* (ILS) was proposed for approximate processing of multiway spatial joins that uses indexes (R-trees) to improve solutions. This algorithm can be easily adapted for DBQs in spatial databases. The general pseudo-code of the ILS algorithm, similar to II in Ioannidis and Kang (1990), is illustrated as follows:

```
bestSolution = S∞ /* S∞ is a fictitious state whose cost is ∞ */
while not (Time threshold) {
        S = random seed
        while not (Local_Minimum(S)) {
                determine an R-tree_i
                S' = find_best_solution(Root of R-tree_i, S)
                if cost(S') < cost(S) then S = S'
        } /* end while not Local_Minimum(S) */
        if cost(S) < cost(bestSolution) then bestSolution = S
} /* end while not Time threshold */
```

For example, in the case of the closest pair query, at the beginning of the algorithm S will be a random pair of points (p_i, q_j), and its neighborhood (S') is defined by fixing one of the elements of the pair (for example, p_i). Then an NNQ (find_best_solution) over the other R-tree (for example, R_Q) is performed, using p_i as a query point and distances as cost functions. If the result of this query is q_j, the condition of Local_Minimum(S) is true and the algorithm exits the inner loop. The process continues until the time threshold is reached.

There have been many attempts to include some deterministic features in local searches and achieve a more systematic exploration of the problem space. In the graph terminology, "memory" mechanisms guarantee that the algorithm will not find the same nodes repeatedly by keeping a list of visited nodes. These nodes

become forbidden *(tabu)* in the graph, forcing the algorithms to move to new neighborhoods. *Guided Indexed Local Search* (GILS) (Papadias & Arkoumanis, 2002) combines the above ideas by keeping a memory, not of all the solutions visited, but of the assignments at local minimums. In particular, the pseudo-code of GILS is similar to ILS. The difference is that GILS generates one random seed only during its execution and has some additional code for penalty assignment.

Simulated Annealing

Simulated Annealing (Kirkpatrick, Gelat, & Vecchi, 1983) is a heuristic search technique based on ideas from physics (derived by analogy to the process of annealing crystals). The essential idea is not to restrict the search algorithm to moves in the search space that decrease the cost function (for a cost function we are trying to minimize), but to also allow (with some probability) moves that can increase the cost function. In principle, this allows a search algorithm to escape from a local minimum. The probability of such non-decreasing moves is set to be quite high early in the process and is gradually decreased as the search progresses. The decrease in this probability is analogous to the process of gradually decreasing the temperature in the physical process of annealing a metal with the goal of obtaining a low-energy state in the metal (hence the name of the method).

For the search algorithm, higher temperatures correspond to a greater probability of large moves in the search space, while lower temperatures correspond to greater probability of only small moves that decrease the cost function. Initially, the temperature schedule reduces the temperature to zero, so that the algorithm only moves to candidates that decrease the cost function. Thus, at this stage of the search, the algorithm will inevitably converge to a point at which no further decrease is possible. The hope is that the earlier moves have led the algorithm to the deepest "basin" in the cost function surface. In fact, one of the appeals of this approach is that it can be mathematically proved that (under fairly general conditions) this will happen if one is using the appropriate temperature schedule. In practice, however, there is usually no way to specify the optimal temperature schedule for any specific problem. Thus, the practical application of simulated annealing reduces to (yet another) heuristic search method with its own set of algorithmic parameters that are often chosen in an ad-hoc manner.

In the approximation context, the local search methods described above can be modified in two important aspects. First, if the neighborhood $N(x)$ of the current solution x is large, it may not be practical to test all solutions in $N(x)$. Instead, we may randomly select only a certain number of solutions y in $N(x)$. The second modification, which can be more important, is to accept the degradation of

solutions with a certain probability. Namely, the local search can continue from x to a solution y in $N(x)$, even if y has a worse cost than x. The motivation of this modification is to avoid the situation of being stuck in a poor local optimal solution.

Let Δ be the change in the cost value of y from that x. If $\Delta \leq 0$ (that is, an improvement), the current solution is always switched to y (i.e. $x = y$). On the other hand, if $\Delta > 0$, the switch from x to y takes place with probability $e^{-\Delta/T}$, where T is a given positive parameter called "temperature." The search then starts again from the new solution. Parameter T is initially set to a relatively large value so that the switch from x to y occurs more frequently, and then it is gradually decreased as the search proceeds. When T becomes sufficiently small and the solution x does not change for many iterations, T is concluded to be "frozen" and the best solution available by then is reported as the computed approximate solution. In order to obtain high performance, it is essential to tune carefully these parameters according to the problem we want to solve. The general scheme is described by the following steps (*input:* an approximate solution x; and *output:* an improved approximate solution x), where some details are not specified; for example, how to determine in SA1 an initial temperature, how to conclude that the procedure is in equilibrium (SA4) or is frozen (SA5), and how to reduce temperature T in SA5. Of course, other details, which are problem specific, such as how to define the cost and neighborhood, need also be provided.

SA1 Determine an initial temperature T

SA2 Given an x, pick randomly a solution y in $N(x)$, and let Δ be the change in the cost value of y from that of x

SA3 If $\Delta \leq 0$ (that is, improved), then let $x = y$. Otherwise, let $x = y$ with probability $e^{-\Delta/T}$

SA4 If it is concluded that a sufficient number of trials have been made with the current T (that is, in equilibrium), then go to *SA5*. Otherwise return to *SA2* with the current x

SA5 If the current T is concluded to be sufficiently small (that is, frozen), then go to *SA6*. Otherwise reduce the temperature T (for example, $T = f * T$, where f is a factor satisfying $0 < f < 1$) and return to *SA2*

SA6 Halt after outputting the best solution obtained so far as the computed approximate solution x

In Papadias et al. (1999), based on ideas of Ioannidis and Kang (1990), the *Configuration Similarity Simulated Annealing* (CSSA) for structural queries was proposed. These queries belong to a kind of content-based queries where similarity is not defined on visual properties such as color or texture, but on the relation of objects in space. Simulated annealing follows a procedure similar to

local search, but it also accepts uphill moves with some probability. This probability is gradually decreased with time and finally the algorithm accepts only downhill moves, leading to a good local minimum. The intuition behind accepting uphill moves is led by the fact that some local minima may be close to each other, separated by a small number of uphill moves. If only downhill moves were accepted (as in local search), the algorithm would stop at the first local minimum visited, missing a subsequent (and possibly better) one. By using these ideas, we can adapt the ILS algorithm and obtain the *Indexed Simulated Annealing* (ISA). Its pseudo-code is given in the following and it can be easily adapted for DBQs in spatial databases.

```
S = S0 /* S is initialized to a random solution */
T = T0
bestSolution = S
while not (Time threshold) {
    while not (frozen) { /* stopping criterion */
        while not (equilibrium) {
            determine an R-treei
            S' = find_best_solution(Root of R-treei, S)
            Δ = cost(S') – cost(S)
            if Δ ≤ 0 then S = S'
            if Δ > 0 then S = S' with probability e^(–Δ/T)
            if cost(S) < cost(bestSolution) then bestSolution = S
        } /* end while not equilibrium */
        T = reduce(T)
    } /* end while not frozen */
} /* end while not Time threshold */
```

The initial temperature T_0 corresponds to a (usually high) probability of accepting an uphill move. The algorithm tries a number of moves (inner iterations) for each temperature value T. T is gradually decreased, allowing the acceptance uphill moves less frequently. When the temperature is small enough, the probability of accepting an uphill move converges to zero and the algorithm behaves like local search. Then, the "system" is in a state of "frozen" and the algorithm terminates.

We must cite an alternative suggested in Ioannidis and Kang (1990) that is a combination of local search and simulated annealing in the same algorithm (called *Two Phase Optimization*, 2PO). 2PO can be divided in two phases: (1)

the local search algorithm is run for a small period of time (that is, a few local optimizations are performed). The output of this phase (the best local minimum found) is the initial state of the next phase; (2) a simulated annealing algorithm is run with a low initial temperature. Intuitively, the algorithm chooses a local minimum and then searches the area around it, still being able to move in and out of local minima, but practically unable to climb up very high hills.

Genetic Algorithms

Genetic Algorithms (genetic search) (Goldberg, 1989), just as simulated annealing is motivated by ideas from physics, are a general set of heuristic search techniques based on ideas of evolutionary biology. The essential idea is to represent candidates as *chromosomes* (often encoded as binary arrays) and to "evolve" a *population* of such chromosomes by selectively pairing chromosomes to create new offspring. Chromosomes are paired based on their "fitting level" (their similarity level) to encourage the better-fitting chromosomes to survive from one generation to the next (only a limited number of chromosomes are allowed to survive from one generation to the next). There are many variations on this general theme, but the key ideas of *genetic search* are:

- *Maintenance* of a set of chromosomes rather than just a single chromosome, allowing the search algorithm to explore different parts of the search space simultaneously.
- *Creating* new chromosomes for exploration, based on combinations of existing ones, allowing, in effect, the algorithm to "jump" to different parts of the search space.

Genetic search can be viewed as a specific type of heuristic, so it may work well on some problems and less well on others. It is not always clear that it provides better performance on specific problems than a simpler method such as *local search* with random restarts. A practical drawback of this approach is that there are usually many algorithmic parameters (such as the number of chromosomes, specification of how chromosomes are combined, and so forth) that must be specified and it may not be clear what the ideal settings are for these parameters for a given problem.

In Papadias and Arkoumanis (2002), the *Spatial Evolutionary Algorithm* (SEA) was proposed for approximate processing of multiway spatial joins that takes advantage of spatial indexes and the problem structure to improve solutions. This algorithm can be easily adapted to DBQs. In this algorithm, three genetic operations (selection, crossover and mutation) are used. These opera-

tions are repeatedly applied in order to obtain a population (that is, a new set of solutions) with better characteristics, and they are presented in the following.

- **Selection mechanism.** This operation consists of two parts: *evaluation* and *offspring allocation*. Evaluation is performed by measuring the similarity of every solution; offspring generation then allocates to each solution a number of offspring items proportional to its similarity. Techniques for offspring allocation include *ranking*, *proportional selection*, *stochastic remainder*, *tournament*, and so forth. For example, in the tournament technique, each solution S_i competes with a set of T random solutions in the generation. Among the T+1 solutions, the one with the highest similarity replaces S_i. After offspring allocation, the population contains multiple copies of the best solutions, while the worst ones are likely to disappear.
- **Crossover mechanism.** It is the driving force of exploration in genetic algorithms. In the simplest approach, pairs of solutions are selected randomly from the population. For each pair, a *crossover point* is defined randomly, and the solutions beyond it are mutually exchanged with probability μ_c *(crossover rate)*, producing two new solutions. The rationale is that after the exchange of genetic materials, the two newly generated solutions are likely to possess the good characteristics of their parents (building-block hypothesis)(Goldberg, 1989). For DBQs, randomly swapping assignments will most probably generate multiple condition violations. Especially for the later generations, where solutions may have reached high similarities, random crossover may lead to the removal of good solutions. In order to limit the number of violations, a variable *crossover point c* is proposed, which is initially 1 and increases every g_c generations. When a solution S_i is chosen for crossover, *c* terms will retain their current assignments, while the remaining will get the assignments of another solution.
- **Mutation mechanism.** Although it is not a primary search operation and sometimes it is omitted, mutation is very important for *SEA* and the only operation that uses the indexes. At each generation, mutation is applied to every solution in the population with probability μ_m, called the *mutation rate*. The process is similar to ILS, getting a new solution using the procedure *find_best_solution*.

In general, the SEA algorithm first computes the similarity of the solutions (evolution) and then performs offspring allocation (using, for example, the tournament approach), crossover and mutation (following the methods described above). During the initial generations, crossover plays an important role in the formation of new solutions. As time passes, its role gradually diminishes and the algorithm behaves increasingly like ILS, since mutation becomes the main

operation that alters solutions. The pseudo-code of SEA is given in the following and it can be easily adapted for DBQs (Papadias et al., 1999) in spatial databases as ILS, GILS and ISA.

```
P = generate initial set of solutions {S_1, S_2, …, S_p}
while not (Time threshold) {
        compute crossover point c /* increase c every g_c generations */
        for each S_i in P /*evaluation */
                evaluate S_i
                if S_i is the best solution found so far then keep S_i
        for each S_i in P /* offspring allocation */
                compare S_i with T other random solutions in P
                replace S_i with the best among the T +1 solutions
        for each S_i in P /*crossover*/
                with probability μ_c change S_i as follows
                determine set of c variables to keep their current values
                re-instantiate the remaining variables using their values in
                another solution S_j (S_j ∈ P)
        for each S_i in P /* mutation */
                with probability μ_m change S_i as follows
                determine an R-tree_j
                S_i = find_best_solution(Root of R-tree_j, S_i)
} /* end while not Time threshold */
```

In the SEA algorithm, if the best *K* solutions are required, the *k* distinct chromosomes are extracted ($k < K$, the algorithm is executed repeatedly with different initial populations)(Papadias et al., 1999). The main problem of SEA is that it involves numerous parameters; namely, the number T of solutions participating in the tournament (offspring allocation), the crossover (μ_c) and mutation (μ_m) rates, the number of generations g_c during which the crossover point remains constant and the number P of solutions in each population. Furthermore, these parameters are inter-related in the sense that the optimal value for one of them depends on the rest. Careful tuning of these parameters is essential for the good performance of SEA and genetic algorithms in general.

Conclusion and Future Trends

This chapter is a review of the most important approximation techniques and the related algorithms that have appeared in literature for processing DBQs. These techniques and the resulting algorithms aim at minimizing the response time and the number of I/O operations over tree-like structures, while reporting sufficiently good results. There are many future research possibilities in this area. A number of them are summarized in the following.

(1) Recently, in Papadias et al. (2003), an interesting architecture that integrates spatial networks and Euclidean distance, capturing pragmatic constraints, has been proposed, and a framework for processing the most representative spatial queries has been developed. The approximation techniques presented in this chapter could be applied within this framework for reporting sufficiently good results quickly.

(2) The tradeoff between cost and accuracy of the search space reduction techniques presented in that section has been studied by Corral et al. (2002a, 2002b), using the notion of Average Relative Distance Error (ARDE). It would be interesting to also examine other measures, such as Match And Compare (MAC) error (Ioannidis & Poosala, 1999) and Earth Mover's Distance (EMD) error (Rubner et al., 1998) and compare the accuracy, as well as the efficiency of all the techniques presented in this chapter, by performing extensive experimentation for several dimensionalities, different DBQs and artificial and real datasets of various cardinalities.

(3) The application of each approximation technique that we have reviewed in this chapter to other DBQs (for example, Iceberg distance joins) and distance functions could be investigated.

(4) Moving object applications are an example of spatio-temporal applications that require specialized indexing techniques for supporting efficient processing of spatio-temporal queries based on distances (Papadias et al., 2001; Tao & Papadias, 2003). Within this context, the reviewed approximation techniques could be employed to obtain good adequate solutions quickly.

(5) The idea of approximation as a tool against the curse of dimensionality (the variance of distances of points becomes very small as the number of dimensions increases) has been applied for the creation of index structures that can be used for computing of exact solutions also. The vector approximation file (VA-file) (Weber, Schek, & Blott, 1998) and the VA+-file (Ferhatosmanoglu, Tuncel, Agrawal, & El Abbadi, 2000) are two such structures that store approximations of points. The A-tree (Sakurai,

Yoshikawa, Uemura, & Kojima, 2000) is another such structure where the representation of MBRs and data objects is based on relative approximations of their parent MBRs. These structures have been used for the exact computation of nearest-neighbor search queries in high-dimensional spaces outperforming R-tree based techniques. It would be challenging to examine the use of these and other analogous structures for computing approximate solutions of DBQs, achieving even higher savings in the computation cost.

(6) Random Sampling has been used for the approximate computation of aggregate queries in spatial databases (Olken & Rotem, 1993; Vassilakopoulos & Manolopoulos, 1997; Lang & Singh, 2001; Nanopoulos, Theodoridis, & Manolopoulos, 2002), taking advantage of spatial indexes (Region Quadtrees and R-trees). It would be interesting to extend such techniques for the approximate computation of DBQs and compare its performance to the methods presented in this chapter.

References

Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., & Wu, A.Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, *45*(6), 891-923.

Beckmann, N., Kriegel, H.P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD 1990)*, (pp. 322-331).

Berchtold, S., Böhm, C., & Kriegel, H.P. (1998). The pyramid-technique: Towards breaking the curse of dimensionality. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, (pp. 142-153).

Berchtold, S., Kiem, D., & Kriegel, H.P. (1996). The X-tree: An index structure for high-dimensional data. *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB 1996)*, (pp. 28-39).

Böhm, C., & Krebs, F. (2002). High performance data mining using the nearest neighbor join. *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, (pp. 43-50).

Brown, P. (2001). *Object-relational database development: A plumber's guide*. Menlo Park, CA: Prentice Hall.

Chakrabarti, K., Garofalakis, M.N., Rastogi, R., & Shim, K. (2000). Approximate query processing using wavelets. *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*, (pp. 111-122).

Chan, E.P. (2001). Evaluation of buffer queries in spatial databases. *Proceedings of the Seventh International Symposium on Spatial and Temporal Databases (SSTD 2001)*, (pp. 197-216).

Ciaccia, P., & Patella, M. (2000). PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. *Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE 2000)*, (pp. 244-255).

Clarkson, K.L. (1994). An algorithm for approximate closest-point queries. *Proceedings of the 10th ACM Symposium on Computational Geometry (SCG 1994)*, (pp. 160-164).

Corral, A., Cañadas, J., & Vassilakopoulos, M. (2002a). Approximate algorithms for distance-based queries in high-dimensional data spaces using R-trees. *Proceedings of the Advances in Databases and Information Systems (ADBIS 2002)*, (pp. 163-176).

Corral, A., Cañadas, J., & Vassilakopoulos, M. (2002b). Improvements of approximate algorithms for distance-based queries. *Proceedings of the First Hellenic Data Management Symposium (HDMS 2002)*, (pp. 83-102). In */www.dbnet.ece.ntua.gr/HDMS02-proceedings/HDMS02_files/vassilakopoulos.pdf*

Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2000). Closest pair queries in spatial databases. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, (pp. 189-200).

Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2003). Distance join queries of multiple inputs in spatial databases. *Proceedings of the Advances in Databases and Information Systems (ADBIS 2003)*. Lecture notes in *Computer Science 2798*, 323-338.

Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., & El Abbadi, A. (2000). Vector approximation based indexing for non-uniform high dimensional data sets. *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management (CIKM 2000)*, (pp. 202-209).

Gaede, V., & Günther, O. (1998). Multidimensional Access Methods. *ACM Computing Surveys*, *30*(2), 170-231.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley.

Hjaltason, G.R., & Samet, H. (1998). Incremental distance join algorithms for spatial databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SGIMOD 1998)*, (pp. 237-248).

Ioannidis, Y., & Kang, Y. (1990). Randomized algorithms for optimizing large join queries. *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD 1990)*, (pp. 312-321).

Ioannidis, Y., & Poosala, V. (1999). Histogram-based approximation of set-valued query answers. *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB 1999)*, (pp. 174-185).

Kirkpatrick, S., Gelat, C., & Vecchi, M. (1983). Optimization by simulated annealing, *Science, 220*(4598), 671-680.

Lang, C., & Singh, A. (2001). Modeling high-dimensional index structures using sampling. *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, (pp. 389-400).

Laurini, R., & Thomson, D. (1992). *Fundamentals of spatial information systems*. London: Academic Press.

Nanopoulos, A., Theodoridis, Y., & Manolopoulos, Y. (2002). An efficient and effective algorithm for density biased sampling. *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management (CIKM 2002)*, (pp. 398-404).

Olken, F., & Rotem, D. (1993). Sampling from spatial databases. *Proceedings of the Ninth IEEE International Conference on Data Engineering (ICDE 1993)*, (pp. 199-208).

Oracle Technology Network (2001). Oracle Spatial User's Guide and Reference. Retrieved November 7, 2001 from *http://technet.oracle.com/doc/Oracle8i_816/inter.816/a77132.pdf*

Papadias, D., & Arkoumanis, D. (2002). Approximate processing of multiway spatial joins in very large databases. *Proceedings of the Eighth International Conference on Extending Database Technology (EDBT 2002)*, (pp. 179-196). Lecture notes in *Computer Science 2287*.

Papadias, D., Mamoulis, N., & Delis, V. (2001). Approximate spatio-temporal retrieval. *ACM Transactions on Information Systems, 19*(1), 53-96.

Papadias, D., Mantzourogiannis, M., Kalnis, P., Mamoulis, N., & Ahmad, I. (1999). Content-based retrieval using heuristic search. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, (pp. 168-175).

Papadias, D., Zhang, J., Mamoulis, N., & Tao, Y. (2003). Query processing in spatial network databases. *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, (pp. 802-813).

Rubner, Y., Tomasi, C., & Guibas, L. (1998). A metric for distributions with applications to image databases. *Proceedings of the Sixth International Conference on Computer Vision (ICCV 1998)*, (pp. 56-66).

Sakurai, Y., Yoshikawa, M., Uemura, S., & Kojima, H. (2000). The A-tree: An index structure for high-dimensional spaces using relative approximation. *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*, (pp. 516-526).

Shou, Y., Mamoulis, N., Cao, H., Papadias, D., & Cheung, D.W. (2003). Evaluation of Iceberg distance joins. *Proceedings of the Eighth International Symposium on Spatial and Temporal Databases (SSTD 2003)* (pp. 270-288). Lecture notes in *Computer Science 2750*.

Tao, Y., & Papadias, D. (2003). Spatial queries in dynamic environments. *ACM Transactions on Database Systems*, *28*(2), 101-139.

Vassilakopoulos, M., & Manolopoulos, Y. (1997). On sampling regional data. *Data Knowledge Engineering*, *22*(3), 309-318.

Weber, R., Schek, H.J., & Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB 1998)*, (pp. 194-205).

Zezula, P., Savino, P., Amato, G., & Rabitti, F. (1998). Approximate similarity retrieval with M-trees. *VLDB Journal*, *7*(4), 275-293.

Endnotes

[1] Partially supported by the ARCHIMEDES project «Management of Moving Objects and the WWW» of the Technological Educational Institute of Thessaloniki (EPEAEK II).

Chapter VII

Spatial Joins: Algorithms, Cost Models and Optimization

Nikos Mamoulis, University of Hong Kong, Hong Kong

Yannis Theodoridis, University of Piraeus, Greece

Dimitris Papadias,
Hong Kong University of Science and Technology, Hong Kong

Abstract

This chapter describes algorithms, cost models and optimization techniques for spatial joins. Joins are among the most common queries in Spatial Database Management Systems. Due to their importance and high processing cost, a number of algorithms have been proposed covering all possible cases of indexed and non-indexed inputs. We first describe some popular methods for processing binary spatial joins and provide models for selectivity and cost estimation. Then, we discuss evaluation of multiway

spatial joins by integrating binary algorithms and synchronous tree traversal. Going one step further, we show how analytical models can be used to combine the various join operators in optimal evaluation plans. The chapter can serve as a comprehensive reference text to the researcher who wants to learn about this important spatial query operator and to the developer who wants to include spatial query processing modules in a Database System.

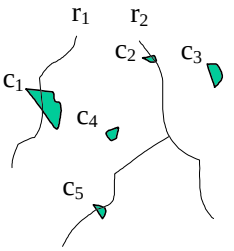
Introduction

Spatial database systems (Güting, 1994) manage large collections of multidimensional data which, apart from conventional features, include special characteristics such as position and extent. That there is no total ordering of objects in space that preserves proximity renders conventional indexes, such as B^+-trees, inapplicable to spatial databases. As a result, a number of *spatial access methods* have been proposed (Gaede & Günther, 1998). A very popular method, used in several commercial systems (for example, Informix and Oracle), is the R-tree (Guttman, 1994), which can be thought of as an extension of B^+-tree in multi-dimensional space. R-trees index object approximations, usually minimum bounding rectangles (MBRs), providing a fast *filter step* that excludes all objects that cannot satisfy a query. A subsequent *refinement step* uses the geometry of the candidate objects (that is, the output of the filter step) to dismiss false hits and retrieve the actual solutions. The R-tree and its variations have been applied to efficiently answer several query types, including spatial selections, nearest neighbors and spatial joins.

As in relational databases, joins play an important role in effective spatial query processing. A *binary* (that is, *pairwise*) spatial join combines two datasets with respect to a spatial predicate (usually *overlap*/*intersect*). A typical example is "find all pairs of cities and rivers that intersect." For instance, in Figure 1 the result of the join between the set of cities $\{c_1, c_2, c_3, c_4, c_5\}$ and rivers $\{r_1, r_2\}$, is $\{(r_1, c_1), (r_2, c_2), (r_2, c_5)\}$.

The query in this example is a spatial *intersection* join. In the general case, the join predicate could be a combination of *topological*, *directional* and *distance* spatial relations. Apart from the intersection join, variants of the *distance* join have received considerable attention because they find application in data analysis tasks (for example, data mining and clustering). Given two sets R and S of spatial objects (or multidimensional points) and a distance function *dist*(), the *ε-distance* join (or else *similarity* join) (Koudas & Sevcik, 2000) returns the

Figure 1. Graphical example of a spatial intersection join

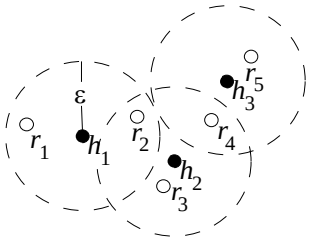


pairs of objects $\{(r, s): r \in R, s \in S, dist(r, s) \leq \varepsilon\}$. A *closest pairs query* (Corral, Manolopoulos, Theodoridis, Vassilakopoulos, 2000) returns the set of closest pairs $CP=\{(r, s): r \in R, s \in S\}$, such that $dist(r, s) \leq dist(r', s')$, for all $r' \in R$, $s' \in S$: $(r', s') \notin CP$. A similar (non-commutative) operator is the *all k-nearest neighbors query* (Böhm & Krebs, 2002), which returns for each object from R its k nearest neighbors in S. Finally, given two datasets R and S, a real number ε and an integer t, the *iceberg distance join* (Shou, Mamoulis, Cao, Papadias, Cheung, 2003) retrieves all pairs of objects from R and S such that: (i) the pairs are within distance ε, and (ii) an object of R appears at least t times in the result (for example, find all regions of R that are within distance 1 km from at least 10 regions of S).

As an example for the spatial join variants, consider Figure 2, which illustrates a set of hotels $\{h_1, h_2, h_3\}$ and a set of restaurants $\{r_1, r_2, r_3, r_4, r_5\}$. The ε-distance join between these two sets returns seven pairs (h_1, r_1), (h_1, r_2), (h_2, r_2), (h_2, r_3), (h_2, r_4), (h_3, r_4) and (h_3, r_5). The 3-closest pairs are (h_2, r_3), (h_3, r_5) and (h_3, r_4). The *all 1-nearest neighbor* operator (for the hotels) returns (h_1, r_2), (h_2, r_3) and (h_3, r_5). Note that ε is not involved in closest pairs and all k-nearest neighbors operations. Finally, the iceberg distance join for t=3 returns (h_2, r_2), (h_2, r_3) and (h_2, r_4). Observe that h_2 is the only hotel with at least 3 nearby restaurants.

In this chapter, we focus on intersection joins by reviewing evaluation algorithms and cost models, as well as techniques for optimizing and integrating them in a spatial database query engine. Other variants of spatial joins can be processed by (trivially or non-trivially) extending algorithms for intersection joins. The interested reader should check Koudas and Sevcik (2000), Corral et al. (2000), Böhm and Krebs (2002) and Shou et al. (2003) for details.

Figure 2. Example of distance join variants



Binary Spatial Joins

Most early spatial join algorithms apply a transformation of objects in order to overcome the difficulties raised by their spatial extent and dimensionality. The first known algorithm (Orenstein, 1986) uses a grid to regularly divide the multidimensional space into small blocks, called *pixels*, and employs a space-filling curve (z-ordering) to order them. Each object is then approximated by the set of pixels intersected by its MBR, that is, a set of z-values. Since z-values are 1-dimensional, the objects can be dynamically indexed using relational index structures, like the B^+-tree, and the spatial join can be performed in a sort-merge join fashion. The performance depends on the granularity of the grid; larger grids can lead to finer object approximations, but also increase the space requirements. Rotem (1991) proposes an algorithm based on a spatial join index similar to the relational join index, which partially pre-computes the join result and employs grid files to index the objects in space.

The most influential algorithm for joining two datasets indexed by R-trees is the *R-tree join* (RJ) (Brinkhoff, Kriegel, Seeger, 1993), due to its efficiency and the popularity of R-trees. RJ synchronously traverses both trees, according to the paradigm of Günther (1993), following entry pairs that overlap; non-intersecting pairs cannot lead to solutions at the lower levels. After RJ, most research efforts focused on spatial join processing for non-indexed inputs. Non-indexed inputs are usually intermediate results of a preceding operator. Consider, for instance, the query "find all *cities* with *population over* 5,000 which are crossed by a *river*." If there are only a few large cities and an index on population, it may be preferable to process the selection part of the query before the spatial join. In such an execution plan, even if there is a spatial index on *cities*, it is not employed by the spatial join algorithm.

The simplest method to process a pairwise join in the presence of one index is by applying a window query to the existing R-tree for each object in the non-

indexed dataset (*index nested loops*). Due to its computational burden, this method is used only when the joined datasets are relatively small. Another approach is to build an R-tree for the non-indexed input using bulk loading (Patel & DeWitt, 1996; Papadopoulos, Rigaux, Scholl, 1999) and then employ RJ to match the trees (*build and match*). Lo and Ravishankhar (1994) use the existing R-tree as a skeleton to build a *seeded tree* for the non-indexed input. The *sort and match* (SaM) algorithm (Papadopoulos et al., 1999) spatially sorts the non-indexed objects but, instead of building the packed tree, it matches each in-memory created leaf node with the leaf nodes of the existing tree that intersect it. Finally, the *slot index spatial join* (SISJ) (Mamoulis & Papadias, 1999, 2003) applies hash-join, using the structure of the existing R-tree to determine the extents of the spatial partitions.

If no indexes exist, both inputs have to be preprocessed in order to facilitate join evaluation. Arge, Procopiuc, Ramaswamy, Suel and Vitter (1998) propose *scalable sweeping-based spatial join* (SSSJ) that employs a combination of *plane sweep* (Preparata & Shamos, 1985) and space partitioning to join the datasets. However, the algorithm cannot avoid external sorting of both datasets, which may lead to large I/O overhead. Patel and DeWitt (1996) describe *partition based spatial merge join* (PBSM) that regularly partitions the space using a rectangular grid, and hashes both inputs into the partitions. It then joins groups of partitions that cover the same area using plane-sweep to produce the join results. Some objects from both sets may be assigned in more than one partition, so the algorithm needs to sort the results in order to remove the duplicate pairs. Another algorithm based on regular space decomposition is the *size separation spatial join* (S^3J) (Koudas & Sevcik, 1997). S^3J avoids replication of objects during the partitioning phase by introducing more than one partition layer. Each object is assigned in a single partition, but one partition may be joined with many upper layers. The number of layers is usually small enough for one partition from each layer to fit in memory; thus, multiple scans during the join phase are not needed. *Spatial hash-join* (SHJ) (Lo & Ravishankar, 1996) avoids duplicate results by performing an irregular decomposition of space based on the data distribution of the build input.

Table 1 summarizes the existing algorithms of all three classes. In general, indexing facilitates efficiency in spatial join processing; an algorithm that uses existing indexes is expected to be more efficient than one that does not consider them. The relative performance of algorithms in the same class depends on the problem characteristics. Günther (1993) suggests that spatial join indices perform best for low join selectivity, while in other cases RJ is the best choice. Among the algorithms in the second class (one indexed input), SISJ and SaM outperform the other methods because they avoid the expensive R-tree construction (Mamoulis & Papadias, 2003). There is no conclusive experimental evaluation for the algorithms in the third class (non-indexed inputs). S^3J is

preferable when the datasets contain relatively large rectangles and extensive replication occurs in SHJ and PBSM. SHJ and PBSM have similar performance when the refinement step is performed exactly after the filter step. In this case, both algorithms sort their output in order to minimize random I/Os and PBSM combines the removal of duplicate pairs with sorting. However, in complex queries (for example, multiway spatial joins) and when the refinement step is postponed after the filter steps of all operators, PBSM may be more expensive, because it can produce larger intermediate results (due to the existence of duplicates). S^3J requires sorting of both datasets to be joined, and therefore it does not favor pipelining and parallelism of spatial joins. On the other hand, the fact that PBSM uses partitions with fixed extents makes it suitable for processing multiple joins in parallel. In the following paragraphs, we review in detail one representative algorithm from each of the three classes, namely the RJ, SHJ and SISJ.

The R-Tree join

The *RJ* (Brinkhoff et al., 1993) is based on the *enclosure property* of R-trees: if two nodes do not intersect, there can be no MBRs below them that intersect. Following this observation, RJ starts from the roots of the trees to be joined and finds pairs of overlapping entries. For each such pair, the algorithm is recursively called until the leaf levels where overlapping pairs constitute solutions. Figure 3 illustrates the pseudo-code for RJ assuming that the trees are of equal height; the extension to different heights is straightforward.

Figure 4 illustrates two datasets indexed by R-trees. Initially, RJ receives the two tree roots as parameters. The qualifying entry pairs at the root level are (A_1, B_1) and (A_2, B_2). Notice that since A_1 does not intersect B_2, there can be no object pairs under these entries that intersect. RJ is recursively called for the nodes pointed by the qualifying entries until the leaf level is reached, where the intersecting pairs (a_1, b_1) and (a_2, b_2) are output.

Table 1. Classification of spatial join methods

Both inputs are indexed	One input is indexed	Neither input is indexed
• transformation to z-values (Orenstein, 1986) • spatial join index (Rotem, 1991) • tree matching (Günther, 1993, Brinkhoff et al., 1993)	• index nested loops • seeded tree join (Lo & Ravishankar, 1994) • build and match (Patel & DeWitt, 1996, Papadopoulos et al., 1999) • sort and match (Papadopoulos et al., 1999) • slot index spatial join (Mamoulis & Papadias, 2003)	• spatial hash join (Lo & Ravishankar, 1996) • partition based spatial merge join (Patel & DeWitt, 1996) • size separation spatial join (Koudas & Sevcik, 1997) • scalable sweeping-based spatial join (Arge et al., 1998)

Two optimization techniques can be used to improve the CPU speed of RJ (Brinkhoff et al., 1993). The first, *search space restriction*, reduces the quadratic number of pairs to be evaluated when two nodes n_i, n_j are joined. If an entry $e_{i,x} \in n_i$ does not intersect the MBR of n_j (that is, the MBR of all entries contained in n_j), then there can be no entry $e_{j,y} \in n_j$, such that $e_{i,x}$ and $e_{j,y}$ overlap. Using this fact, space restriction performs two linear scans in the entries of both nodes before RJ and prunes out from each node the entries that do not intersect the MBR of the other node. The second technique, based on the *plane sweep paradigm*, applies sorting in one dimension in order to reduce the cost of computing overlapping pairs between the nodes to be joined. Plane sweep also saves I/Os compared to nested loops because consecutive computed pairs overlap with high probability. Brinkhoff et al. (1994) discuss multi-step processing of RJ using several approximations (instead of conventional MBRs). Huang, Jing and Rundensteiner (1997a) propose a breadth-first optimized version of RJ that sorts the output at each level in order to reduce the number of page accesses.

Spatial Hash Join

SHJ (Lo & Ravishankar, 1996) (based on the relational hash-join paradigm) computes the spatial join of two non-indexed datasets *R (build* input*)* and *S (probe* input*)*. Set *R* is partitioned into *K* buckets, where *K* is decided by the system parameters. The initial extents of the buckets are points determined by sampling. Each object is inserted into the bucket that is enlarged the least. Set *S* is hashed into buckets with the same extent as *R*'s buckets, but with a different insertion policy: An object is inserted into all buckets that intersect it. Thus, some objects may be assigned to multiple buckets (*replication*) and some may not be inserted at all (*filtering*). The algorithm does not ensure equal-sized partitions for *R* (that is, with the same number of objects in them), as sampling cannot guarantee the best possible bucket extents. Equal-sized partitions for *S* cannot

Figure 3. R-tree-based spatial join

```
RJ(Rtree_Node n_i, RTNode n_j)
    for each entry e_{j,y} ∈ n_j do {
        for each entry e_{i,x} ∈ n_i with e_{i,x} ∩ e_{j,y} ≠ ∅ do {
        if n_i is a leaf node    /* n_j is also a leaf node */
            then Output (e_{i,x}, e_{j,y});
            else  { /* intermediate nodes */
                ReadPage(e_{i,x}.ref); ReadPage(e_{j,y}.ref);
                RJ(e_{i,x}.ref, e_{j,y}.ref); }       }
    } /* end for */
```

be guaranteed in any case, as the distribution of the objects in the two datasets may be different.

Figure 5 shows an example of two datasets, partitioned using the SHJ algorithm. After hashing set *S*, the two bucket sets are joined; each bucket from *R* is matched with only one bucket from *S*, thus requiring a single scan of both files, unless for some pair neither bucket fits in memory. In this case, an R-tree is built for one of them, and the bucket-to-bucket join is executed in an index nested loop fashion.

Slot Index Spatial Join

SISJ (Mamoulis & Papadias, 2003) is applicable when there is an R-tree for one of the inputs (*R*). The algorithm is similar to SHJ, but uses the R-tree on *R* in order to determine the bucket extents. If *K* is the desired number of partitions, SISJ will find the topmost level of the tree such that the number of entries is larger than or equal to *K*. These entries are then grouped into *K* (possibly overlapping) partitions called *slots*. Each slot contains the MBR of the indexed R-tree entries, along with a list of pointers to these entries. Figure 6 illustrates a 3-level R-tree (the leaf level is not shown) and a slot index built over it. If $K = 9$, the root level contains too few entries to be used as partition buckets. As the number of entries in the next level is over *K*, we partition them in 9 slots (for this example). The grouping policy of SISJ starts with a single empty slot and inserts entries into the slot that is enlarged the least. When the maximum capacity of a slot is reached (determined by *K* and the total number of entries), either some entries are deleted and reinserted or the slot is split according to the R*-tree splitting policy (Beckmann, Kriegel, Schneider & Seeger, 1990).

After building the slot index, the second dataset *S* is hashed into buckets with the same extents as the slots. If an object from *S* does not intersect any bucket, it

Figure 4. Two datasets indexed by R-trees

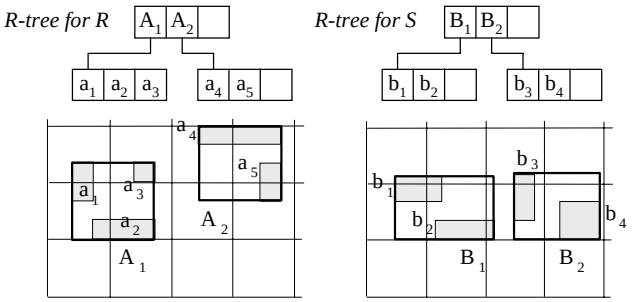
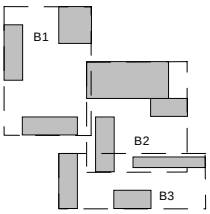
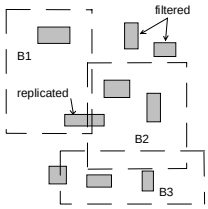


Figure 5. The partitioning phase of SHJ algorithm



(a) Objects from set *R* (build input) in three partition buckets



(b) Filtering and replication of objects from set *S* (probe input)

is filtered; if it intersects more than one bucket, it is replicated. The join phase of SISJ is also similar to the corresponding phase of SHJ. All data from the R-tree of *R* indexed by a slot are loaded and joined with the corresponding hash-bucket from *S* using plane sweep. If the data to be joined do not fit in memory, they can be joined using the algorithm of Arge et al. (1998), which employs external sorting and then plane sweep. Another alternative is index nested loop join (using as a root of the R-tree the corresponding slot). These methods can be expensive when the partitions are much larger than the buffer. In such cases SISJ is applied recursively, in a similar way to recursive hash-join. During the join phase of SISJ, when no data from *S* is inserted into a bucket, the sub-tree data under the corresponding slot is not loaded (slot filtering).

Selectivity and Cost Estimation for Spatial Joins

Estimating the cost and the output size of a spatial join is an important and difficult problem. Accurate cost models are necessary for the query optimizer to identify a good execution plan that accelerates retrieval and minimizes the usage of system resources. The output size of a spatial join between datasets *R* and *S* depends on three factors:

- The *cardinalities* $|R|$ and $|S|$ of the datasets. The join may produce up to $|R|\times|S|$ tuples (that is, the Cartesian product).
- The *density* of the datasets. The density of a dataset is formally defined as the sum of areas of all objects in it divided by the area of the workspace[1]. In other words, it is the expected number of objects that intersect a random

Figure 6. An R-tree and a slot index built over it

(a) level 2 (root) entries (b) level 1 entries (c) slot index over level 1

point in the workspace. Datasets with high density have objects with large average area, and produce numerous intersections when joined.

- The *distribution* of the MBRs. Skewed datasets may produce arbitrary few or many join pairs. Data skew is the most difficult factor to estimate, since in many cases the distribution is not known, and even if known, its characteristics are very difficult to capture.

The I/O cost of the refinement step is determined by the selectivity of the filter step, since for each candidate object (or object pair) a random access that retrieves its exact geometry is required. However, the selectivity of the refinement step is hard to estimate because the arbitrary extents of the actual objects do not allow for the easy computation of quantities like density and complicate the probabilistic analysis of overlapping regions. Although this estimate does not affect the cost of the spatial operator, it can be crucial for the cost estimate of operators that succeed it. For example, for a complex query, where three datasets are joined, the selectivity of the first join determines the input size of the second. Estimating the selectivity of a spatial query after the refinement step is a challenging issue, and to the best of our knowledge, no previous work sufficiently solves this problem. Existing studies focus on the filter step, often assuming that the data are uniformly distributed in the workspace (*uniformity assumption*). Several of these studies are based on selectivity and cost estimation formulae for window queries.

Selectivity and Cost Estimation for Window Queries

Given a spatial dataset R consisting of $|R|$ δ-dimensional uniformly distributed rectangles in a rectangular area u (workspace universe), the number of rectangles that intersect a window query w (*output cardinality* - *OC*) is estimated by the following formula,

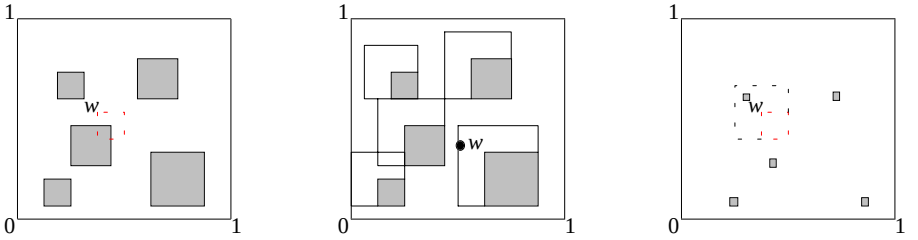
$$OC(R,w) = |R| \cdot \prod_{d=1}^{\delta} \min\left(1, \frac{\overline{r_d} + \overline{w_d}}{\overline{u_d}}\right), \quad (1)$$

where $\overline{r_d}$ is the average length of the projection of a rectangle $r \in R$ at dimension d, and $\overline{w_d}$, $\overline{u_d}$ are the corresponding projections of w, u respectively. The product in Equation 1, called Minkowski sum, depends on the probability that a random rectangle from R intersects w. A graphical example is given in Figure 7. In particular, Figure 7a depicts a dataset R and a window query w. We can think of w as a point query on a dataset that contains rectangles of average projection $\overline{r_d} + \overline{w_d}$ (Figure 7b), in which case the goal is to retrieve all rectangles that contain the query point. Alternatively, the query can be transformed to a rectangle with average side $\overline{r_d} + \overline{w_d}$ on $|R|$ points (Figure 7c), in which case the goal is to retrieve the data points falling in the query window. The *min* function in Equation 1 avoids boundary effects when $\overline{r_d} + \overline{w_d} > 1$ for some dimension d.

The output size for non-uniform data can be estimated by maintaining a histogram that partitions the data space into a set of *buckets*, and assuming that object distribution in each bucket is (almost) uniform. Specifically, each bucket b contains the number $b.num$ of objects whose centroids fall in b, and the average extent $b.len$ of such objects. Figure 8 illustrates an example in the 2D space, where the gray area corresponds to the intersection between b and the extended query region, obtained by enlarging each edge of q with distance $b.len/2$. Following the analysis on uniform data, the expected number of qualifying objects in b approximates $b.num \bullet I.area/b.area$, where $I.area$ and b.area are the areas of the intersection region and b, respectively (Acharya, Poosala, & Ramaswamy, 1999). The total number of objects intersecting q is predicted by summing the results of all buckets. Evidently, satisfactory estimation accuracy depends on the degree of uniformity of objects' distributions in the buckets. This can be maximized using various algorithms (Muralikrishna & DeWitt, 1988; Poosala & Ioannidis, 1997; Acharya et al., 1999), which differ in the way that buckets are structured. For example, in Muralikrishna & DeWitt (1988), buckets have similar sizes (that is, "equi-width") or cover approximately the same number of objects (that is, "equi-depth"), while in Poosala and Ioannidis (1997) and Acharya et al. (1999) bucket extents minimize the so-called "spatial skew."

When the dataset is not indexed, the cost of a window query (in terms of disk accesses) is equal to the cost of sequentially scanning the entire dataset (that is,

Figure 7. Output size estimation for window queries



(a) a dataset and a window query

(b) a point query on rectangles of average side $\overline{r_d}$ $\overline{w_d}$ in d={1,2}

(c) a window query of side $\overline{r_d}$ $\overline{w_d}$ on points

it is independent of the selectivity). On the other hand, the existence of an R-tree can significantly reduce the query cost. The number of R-tree pages accessed when processing a window query is equal to the expected number of non-leaf node entries that intersect the query window plus the access of the R-tree root. Let L be the number of R-tree levels and N_l ($\overline{r_{d,l}}$) be the number of entries (average entry projection length) at level l and dimension d (0 is the leaf level and L-1 the root level). The cost of a window query is then given by the following formula (Kamel & Faloutsos, 1993; Theodoridis & Sellis, 1996):

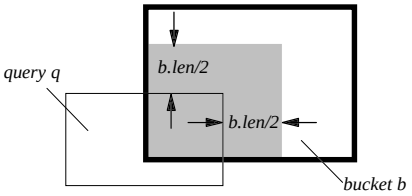
$$Cost(R, w) = 1 + \sum_{l=1}^{L-1} N_l \cdot \prod_{d=1}^{\delta} \left(\min\{1, \frac{\overline{r_{d,l}} + \overline{w_d}}{\overline{u_d}}\}\right) \qquad (2)$$

Theodoridis & Sellis (1996) present formulae for the estimation of $\overline{r_{d,l}}$ for each R-tree level that are based solely on the cardinality of the dataset $|R|$, the average object MBR projection length and the page size, which determines the capacity of the nodes and the height of the tree. The cost for non-uniform datasets can be computed with the aid of histograms (similar to the selectivity case).

Selectivity and Cost Estimation for Spatial Joins

The output size (and selectivity) of a spatial join can be estimated by extending Equation 1 (and Equation 2) in a straightforward way. Let R, S be the joined datasets. The output of the join is the expected number of rectangles retrieved

Figure 8. Estimation using histograms



from R when applying $|S|$ window queries of projection length $\overline{s_d}$. Conversely, it is the expected number of rectangles retrieved from S when applying $|R|$ window queries of projection length $\overline{r_d}$. In either case the output size is:

$$OC(R,S) = |R| \cdot |S| \cdot \prod_{d=1}^{k} \min\left(1, \frac{\overline{r_d} + \overline{s_d}}{\overline{u_d}}\right) \quad (3)$$

For other data distributions where the uniformity assumption does not hold, one can use 2D histograms that divide the space into buckets and summarize local statistics for each bucket (Theodoridis et al., 1998; Mamoulis & Papadias, 1999). The uniformity assumption can then be applied to each region to accumulate the estimate for the join output. An et al. (An et al., 2001) extend this method by maintaining, for each bucket, statistics about the objects' edges and corners that intersect it. Parametric approaches for distance joins, based on the observation that distances between spatial objects follow power laws, were proposed in (Belussi & Faloutsos, 1998, Faloutsos et al., 2000). Approximating the distribution (or object distances) in a real spatial dataset using histograms (or functions) cannot provide worst-case guarantees for query selectivity estimation. As a result, the effectiveness of most of the above methods is evaluated experimentally. An, Yang and Sivasubramaniam (2001) show by experimentation that their method is more accurate compared to the techniques used in Theodoridis et al. (1998)and in Mamoulis and Papadias (1999) for joins between sets of MBRs. Belussi and Faloutsos (1998) and Faloutsos, Seeger, Traina and Traina (2000) provide experimental evidence for the accuracy of their models, though they are incomparable with An et al. (2001), since they are only applicable for distance joins between point sets.

Cost of R-Tree Join

Theodoridis et al. (1998) studied the cost of RJ in terms of R-tree node accesses. Let R and S be two datasets indexed by R-trees and assume that the two R-trees have (at level l) average entry side $\overline{r_{d,l}}$, $\overline{s_{d,l}}$ and number of entries $N_{R,l}$, $N_{S,l}$, respectively. The number of node accesses during their spatial join can be estimated by the following formula:

$$Cost_{NA}(\text{RJ}, R, S) = 2 + 2 \cdot \sum_{l=1}^{L-1} N_{R,l} \cdot N_{S,l} \cdot \prod_{d=1}^{\delta} \left(\min\{1, \frac{\overline{r_{d,l}} + \overline{s_{d,l}}}{\overline{u_d}}\}\right) \quad (4)$$

Equation 4 expresses that every pair of intersecting entries at level l is responsible for two node accesses at level l-1, if l>0. Therefore, the sum of the expected number of intersecting entry pairs at the high levels of the trees, plus the two accesses of the tree roots, give an estimate of the total number of node accesses. Nevertheless, this quantity can be considered as an upper bound only, since it does not reflect the actual number of I/Os under the existence of an LRU buffer. When an intersecting pair of entries needs to be loaded, there is a high probability that these pages will be in the system buffer if the buffer is large and if they have been requested before. Huang et al. (1997b) provide an analysis of RJ based on this observation, according to which, the I/O cost of joining R and S in the presence of an LRU buffer is given by the following formula:

$$Cost(\text{RJ}, R, S) = T_R + T_S + (Cost_{NA}(\text{RJ}, R, S) - T_R - T_S) \times Prob(\text{node}, M), \quad (5)$$

where T_R, T_S are the number of nodes in the R-trees for R and S, respectively, and $Prob$(node, M) is the probability that a requested R-tree node will not be in the buffer (of size M), resulting in a page fault. This probability falls exponentially with M, and its estimation is based on an empirical analysis.

Cost of Spatial Hash Join

The I/O cost of SHJ depends on the size of the joined datasets and the filtering and replication that occur in set S. Initially, a small number of pages $Cost_{sam}$ is loaded to determine the initial hash buckets. Then both sets are read and hashed

into buckets. Let P_R, P_S be the number of pages of the two datasets (stored in sequential files) and rep_S, fil_S be the replication and filtering ratios of S. The partitioning cost of SHJ is given by the following formula:

$$Cost_{part}(\text{SHJ}, R, S) = Cost_{sam} + 2\ P_R + (2 + rep_S - fil_S) \times P_S \quad (6)$$

Next, the algorithm will join the contents of the buckets from both sets. In typical cases, where the buffer is large enough for at least one partition to fit in memory, the join cost of SHJ is,

$$Cost_{join}(\text{SHJ}, R, S) = P_R + (1 + rep_S - fil_S) \times P_S, \quad (7)$$

considering that the join output is not written to disk. Summing up, from Equations 6 and 7, the total cost of SHJ is:

$$Cost(\text{SHJ}, R, S) = Cost_{sam} + 3\ P_R + (3 + 2rep_S - 2fil_S) \times P_S \quad (8)$$

Cost of Slot Index Spatial Join

SISJ joins a dataset R indexed by an R-tree with a non-indexed file S. Let T_R be the number of R-tree nodes of R, and P_S the number of pages in S. Initially, the slots have to be determined from R. This requires loading the top K levels of R's R-tree, in order to find the appropriate slot level. Let $frac_R$ be the fraction of tree nodes from the root until K. The slot index is built in memory, without additional I/Os. Set S is then hashed into the slots requiring P_S accesses for reading, and $P_S + rep_S P_S - fil_S P_S$ accesses for writing, where rep_S, fil_S are the replication and filtering ratios of S. Thus, the cost of SISJ partition phase is:

$$Cost_{part}(\text{SISJ}, R, S) = frac_R \times T_R + (2 + rep_S - fil_S) \times P_S \quad (9)$$

For the join phase of SISJ, we make the same assumptions as for SHJ; that is, for each joined pair at least one bucket fits in memory. The pages from set R that have to be fetched for the join phase are the remaining $(1\text{-}frac_R) \times T_R$, since the pointers to the slot entries are kept in the slot index and need not be loaded again from the top levels of the R-tree. The number of I/O accesses required for the join phase is:

$$Cost_{join}(\text{SISJ}, R, S) = (1 - frac_R) \times T_R + (1 + rep_S - fil_S) \times P_S \quad (10)$$

Summarizing, the overall cost of SISJ is:

$$Cost(\text{SISJ}, R, S) = T_R + (3 + 2rep_S - 2fil_S) \times P_S \quad (11)$$

We can qualitatively compare the three algorithms from Equations 5, 8 and 11. Given a large enough memory buffer, the cost of RJ is not much higher than $T_R + T_S$, since we expect that every requested R-tree node that has been loaded will remain in the memory buffer with high probability, due to the locality of accessed pages. This assertion is experimentally verified in several studies (for example, Huang et al., 1997b; Mamoulis & Papadias, 1999). Given that in typical R-tree structures nodes have around 67% average utilization (Beckmann et al., 1990), and that the non-leaf R-tree nodes are very few compared to the leaves (due to the large fanouts, 100-200 in practice), the I/O cost of RJ is roughly the cost of reading 150% of the total number of pages occupied by the rectangles (that is, $Cost(\text{SISJ}, R, S) \approx T_R + T_S \approx 1.5(P_R + P_S)$). The cost of SISJ is affected by the filtering and replication ratios, which cannot be easily predicted. From empirical studies on real datasets (Mamoulis & Papadias, 2003), it has been observed that in practice, $rep_S \approx 0.3$ and $fil_S \approx 0$. Considering this and the discussion on average R-tree node occupancy, we can approximate the cost of SISJ with $1.5P_R + 3.6P_S$. With similar assumptions on the filtering and replication ratios (and assuming a negligible sampling cost), the cost of SHJ is reduced to $3P_R + 3.6P_S$. Based on these numbers, we can conclude that RJ is more efficient than SISJ, which is more efficient than SHJ, under usual problem settings. Of course, the application of RJ (SISJ) presumes the existence of two (one) R-trees.

In the next sections, we discuss how the cost estimation formulae for RJ, SHJ and SISJ can be used in combination with the selectivity estimation models discussed earlier to optimize complex queries that include spatial join operators. The experimental study of Mamoulis and Papadias (2001) suggests that these estimates are indeed accurate and usually lead to optimal plans.

Multiway Spatial Joins

Multiway spatial joins involve an arbitrary number of spatial inputs. Such queries are important in several applications, including Geographical Information Systems (for example, "find all cities *adjacent to* forests, which are *intersected* by a river") and VLSI (for example, "find all sub-circuits that formulate a

specific topological configuration"). Formally, a multiway spatial join can be expressed as follows: Given n datasets R_1, R_2, ... R_n and a *query* Q, where Q_{ij} is the spatial predicate that should hold between R_i and R_j, retrieve all n-tuples $\{(r_{1,w},..,r_{i,x},..,r_{j,y},.., r_{n,z}) \mid \forall\, i,j : r_{i,x} \in R_i, r_{j,y} \in R_j$ and $r_{i,x}\ Q_{ij}\ r_{j,y}\}$. The query can be represented by a graph, where nodes correspond to datasets and edges to join predicates. Equivalently, the graph can be viewed as a spatial *constraint network*, where the nodes correspond to problem variables and edges to binary spatial constraints. In the sequel we use the terms variable/dataset and constraint/join condition interchangeably.

We consider that all datasets are indexed by R-trees (on MBRs) and we deal with the filter step, assuming that *overlap* is the default join condition; that is, if Q_{ij} = True, then the rectangles from the corresponding inputs i,j should overlap. The loosest query is the one that corresponds to an acyclic (*tree*) graph (for example, the one illustrated in Figure 9a), while the most constrained consists of a complete (*clique*) graph (for example, the one in Figure 9c). For each type of query, Figure 9 illustrates a solution; that is, a configuration of rectangles $r_{i,1} \in R_i$ that satisfies the join conditions. We do not consider non-connected query graphs, as these can be processed by solving connected sub-graphs and then computing their Cartesian product.

Patel and DeWitt (1996) apply PBSM in a distributed, multi-processor environment to process cascading joins. Spatial datasets are regularly partitioned in space (*spatial declustering*), and the physical resources (disks, processors) are distributed according to the partitions. Papadopoulos et al. (1999) perform a two-join case study to evaluate the performance of four spatial join algorithms. Mamoulis and Papadias (1999) propose a *pairwise joins method* (PJM) that combines binary join algorithms in a processing tree where the leaves are input relations indexed by R-trees and the intermediate nodes are join operators.

Processing multiway joins by integration of pairwise join algorithms is the standard approach in relational databases where the join conditions usually relate different attributes. In spatial joins, however, the conditions refer to a single spatial attribute for all inputs; that is, all object sets are joined with respect to their spatial features. Motivated by this fact, *synchronous traversal* (ST) traverses top-down[2] all the R-trees involved in the query, excluding combinations of intermediate nodes that do not satisfy the join conditions. The first general application of ST to an arbitrary number of inputs appeared in Papadias et al. (1998) for retrieval of database images matching some input configuration. The employment of the method in multi-way spatial join processing is discussed in Papadias et al. (1999) and in Mamoulis and Papadias (2001), together with formulae for selectivity (in uniform datasets) and cost estimation (in terms of node accesses). Next, we present in detail PJM and ST. Finally, we discuss the optimization of processing multiway spatial joins based on dynamic programming.

Integration of Pairwise Join Algorithms for Processing Multiple Inputs

As in the case of relational joins, multiway spatial joins can be processed by combining pairwise join algorithms. PJM considers a join order that is expected to result in the minimum cost (in terms of page accesses). Each join order corresponds to a single execution plan, where:

(i) RJ is applied when the inputs are leaves; that is, datasets indexed by R-trees,

(ii) SISJ is employed when only one input is indexed by an R-tree,

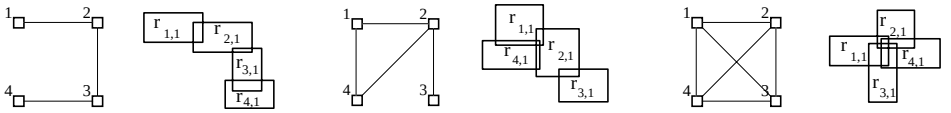
(iii) SHJ is used when both inputs are intermediate results.

As an example of PJM, consider the query in Figure 9a and the plans of Figure 10. Figure 10a involves the execution of RJ for determining $R_3 \bowtie R_4$. The intermediate result, which is not indexed, is joined with R_2 and finally with R_1 using SISJ. On the other hand, the plan of Figure 10b applies RJ for $R_1 \bowtie R_2$ and $R_3 \bowtie R_4$, and SHJ to join the intermediate results.

Queries with cycles can be executed by transforming them to tree expressions using the most selective edges of the graph and filtering the results with respect to the other relations in memory. For instance, consider the cycle (R_1 *overlap* R_2), (R_2 *overlap* R_3), (R_3 *overlap* R_1) and the query execution plan $R_1 \bowtie (R_2 \bowtie R_3)$. When joining the tuples of ($R_2 \bowtie R_3$) with R_1 we can use either the predicate (R_2 *overlap* R_1), or (R_3 *overlap* R_1) as the join condition. If (R_2 *overlap* R_1) is the most selective one (that is, results in the minimum cost), it is applied for the join, and the qualifying tuples are filtered with respect to (R_3 *overlap* R_1).

PJM uses Equations 5, 8 and 11 to estimate the join cost of the three algorithms. The expected output size of a pairwise join determines the execution cost of an upper operator and therefore is crucial for optimization. Selectivity estimation for a pairwise join has already been discussed. Optimization of multiway spatial joins

Figure 9. Multiway join examples



requires selectivity estimation for each possible decomposition of the query graph (that is, for each allowable sub-plan). The generalized formula for the output size of a query (sub) graph Q with n inputs is:

$$OC(R_1,R_2, \ldots, R_n, Q) = \#(possible\ tuples) \times Prob(a\ tuple\ is\ a\ solution) \quad (12)$$

The first part of the product equals the cardinality of the Cartesian product of the n domains, while the second part corresponds to *multiway join selectivity*. In case of acyclic graphs, the pairwise probabilities of the join edges are independent and selectivity is the product of pairwise join selectivities (Papadias et al., 1999):

$$Prob(a\ tuple\ is\ a\ solution) = \prod_{\forall i,j:Q_{ij}=TRUE} \prod_{d=1}^{\delta} \min\left(1, \frac{\overline{r_{i,d}} + \overline{r_{j,d}}}{\overline{u_d}}\right) \quad (13)$$

From Equations 12 and 13, the total number of query solutions is:

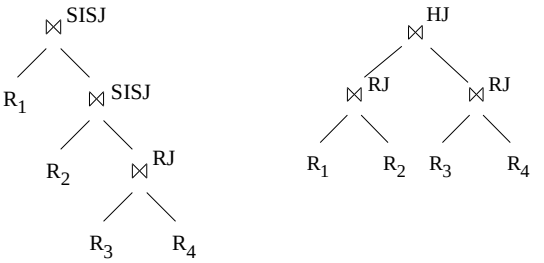
$$OC(R_1,\ldots,R_n,Q) = \prod_{i=1}^{n} |R_i| \cdot \prod_{\forall i,j:Q_{ij}=TRUE} \prod_{d=1}^{\delta} \min\left(1, \frac{\overline{r_{i,d}} + \overline{r_{j,d}}}{\overline{u_d}}\right) \quad (14)$$

When the query graph contains cycles, the pairwise selectivities are no longer independent and Equation 14 is not accurate. For cliques, it is possible to provide a formula for multiway join selectivity based on the fact that if a set of rectangles mutually overlap, then they must share a common area. Given a random n-tuple of rectangles, the probability that all rectangles mutually overlap is (Papadias et al., 1999):

$$Prob(a\ tuple\ is\ a\ solution) = \prod_{d=1}^{\delta} \frac{1}{(n-1)\cdot \overline{u_d}} \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} \overline{r_{j,d}} \quad (15)$$

Thus, in case of clique queries Q the number of solutions is:

Figure 10. Alternative plans using pairwise join algorithms



(a) right-deep plan (b) bushy plan

$$OC(R_1,...,R_n,Q) = \prod_{i=1}^{n} |R_i| \cdot \prod_{d=1}^{\delta} \frac{1}{(n-1)\cdot \overline{u_d}} \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} \overline{r_{j,d}} \qquad (16)$$

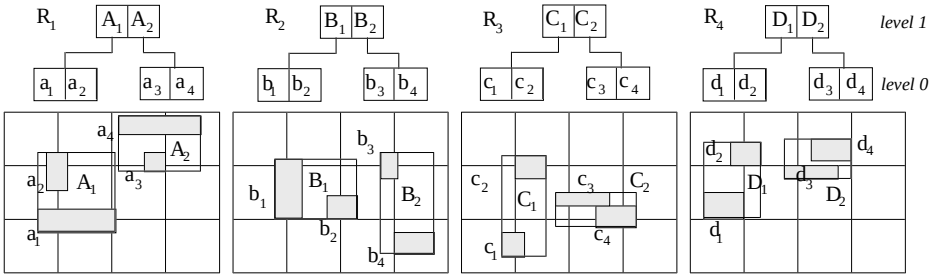
The above formulae are applicable for queries that can be decomposed to acyclic and clique graphs (for example, Figure 9b). The optimal execution plan can be computed from the estimated output size and the costs of the algorithms involved. Selectivity estimation for real datasets can be performed using histograms. Next we describe ST, an alternative to PJM for processing multiway spatial joins.

Synchronous Traversal

ST processes the indexes of all joined datasets, following combinations of nodes that satisfy the query constraints. Consider the four R-trees of Figure 11 and the clique query of Figure 9c. The query asks for the set of 4-tuples (a_w, b_x, c_y, d_z), such that the four objects mutually overlap (for example, (a_2, b_1, c_2, d_2)). ST starts from the roots of the R-trees, searching for entries that satisfy the join conditions. In this example, out of the 16 combinations of root entries (that is, (A_1, B_1, C_1, D_1), (A_1, B_1, C_1, D_2), .., (A_2, B_2, C_2, D_2)), only (A_1, B_1, C_1, D_1) may lead to actual solutions. For instance, the combination (A_2, B_1, C_1, D_1) does not satisfy the query constraints because A_2 does not intersect C_1 (or D_1); therefore, there cannot be any pair of overlapping objects (a_w, c_y), a_w pointed by A_2 and c_y pointed by C_1. As in the case of RJ, for each intermediate level solution, the algorithm is called for the pointed R-tree nodes, recursively, until the leaves, where solutions are output.

In the worst case, the total number of combinations of data MBRs that have to be checked for the satisfaction of the join conditions is $|R|^n$, where n is the number

Figure 11. Example of four R-trees



of inputs and $|R|$ the cardinality of the datasets (assumed to be the same). ST takes advantage of the hierarchical decomposition of space preserved by R-trees to break the problem in smaller *local* ones at each tree level. A local problem has to check C^n combinations in the worst case (C is the R-tree node capacity), and can be defined by:

- A set of n variables, v_1, v_2,.., v_n, each corresponding to a dataset.
- For each variable v_i, a domain Δ_i consisting of the entries $\{e_{i,1},.., e_{i,Ci}\}$ of a node n_i (in tree R_i).
- Each pair of variables (v_i, v_j) is constrained by *overlap*, if Q_{ij} is True.

A binary assignment $\{v_i \leftarrow e_{i,x}, v_j \leftarrow e_{j,y}\}$ is *consistent* if Q_{ij} = True $\Rightarrow e_{i,x}$ overlaps $e_{j,y}$. A solution of a local problem is a n-tuple t = $(e_{1,w},.., e_{i,x},.., e_{j,y},..,e_{n,z})$ such that $\forall i,j$, $\{v_i \leftarrow e_{i,x}, v_j \leftarrow e_{j,y}\}$ is consistent. The goal is to find all *solutions*; that is, assignments of entries to variables such that all constraints are satisfied. In the previous example (clique query of Figure 9c), there exist four variables $v_1,..,v_4$, and for each (v_i,v_j), $i \neq j$, the constraint is *overlap*. At level 1 the domains of the variables are $\Delta_1=\{A_1,A_2\}$, $\Delta_2=\{B_1,B_2\}$, $\Delta_3=\{C_1,C_2\}$ and $\Delta_4=\{D_1,D_2\}$. Once the root level solution (A_1,B_1,C_1,D_1) is found, ST will recursively search for qualifying tuples at the lower level, where the domains of $v_1,..,v_4$ consist of the entries under $A_1,..,D_1$, respectively; that is, $\Delta_1=\{a_1,a_2\}$, $\Delta_2=\{b_1,b_2\}$, $\Delta_3=\{c_1,c_2\}$ and $\Delta_4=\{d_1,d_2\}$. Notice that an intermediate-level solution does not necessarily lead to an actual one. Since a part of the node area corresponds to "dead space" (space not covered by object MBRs), many high-level solutions are *false hits*. The pseudo-code for ST, assuming R-trees of equal height, is presented in Figure 12.

For each Δ_i, *space-restriction* prunes all entries that do not intersect the MBR of some n_j, where Q_{ij} = True. Consider the chain query of Figure 9a and the top-level solution (A_2, B_1, C_1, D_1). At the next level ST is called with $\Delta_1 = \{a_3, a_4\}$, $\Delta_2 = \{b_1, b_2\}$, $\Delta_3 = \{c_1, c_2\}$ and $\Delta_4 = \{d_1, d_2\}$. Although A_2 intersects B_1, none of

entries (a_3, a_4) do and these entries can be safely eliminated from Δ_1. Since Δ_1 becomes empty, (A_2, B_1, C_1, D_1) cannot lead to an actual solution and the search is abandoned without loading the nodes pointed by B_1, C_1 and D_1. *Find-combinations* is the "heart" of ST; that is, the search algorithm that finds tuples $t \in \Delta_1 \times \Delta_2 \times ... \times \Delta_n$, that satisfy Q. In order to avoid exhaustive search of all combinations, several backtracking algorithms applied for constraint satisfaction problems can be used. The implementation of Mamoulis and Papadias (2001) uses *forward checking* (FC) (Haralick & Elliott, 1981), which accelerates search by progressively assigning values to variables and pruning the domains of future (non-instantiated) variables. Given a specific order of the problem's variables v_1, v_2,…, v_n, when v_i is instantiated, the domains of all future variables v_j, $j > i$, such that Q_{ij} = True, are revised to contain only rectangles that intersect the current instantiation of v_j (*check forward*). If during this procedure some domain is eliminated, a new value is tried for v_i until the end of D_i is reached. Then FC *backtracks* to v_{i-1}, trying a new value for this variable.

Figure 12. Synchronous R-tree traversal

```
ST(Query Q[][], RTNode n[])
    for i:=1 to n do {  /*prune domains*/
            Δi := space-restriction(Q, n[], i);
            if  Δi = ∅ then return;  /*no qualifying tuples exist for this combination of nodes*/}
    for each τ ∈ find-combinations(Q, Δ) do { /* for each solution at the current level */
            if n[] are leaf nodes then /*qualifying tuple is at leaf level*/
                    Output(τ);
            else /*qualifying tuple is at intermediate level*/
                    ST(Q, τ.ref[]); /* recursive call to lower level */}

Domain space-restriction(Query Q[][], RTNode n[], int i)
    read ni; /* read node from disk */
    Δi := ∅;
    for each entry ei,x ∈ ni  do {
            valid := True; /*mark ei,x as valid */
            for each node nj such that Qij = True do {/*an edge exists between ni and nj*/
                    if  ei,x ∩ nj.MBR = ∅ then { /* ei,x does not intersect the MBR of node ni */
                            valid := false; /* ei,x is pruned */
                            break;}
        if valid = True then /*ei,x is consistent with all node MBRs*/
                Δi := Δi ∪ ei,x;
    }
    return Δi;
```

ST Cost Estimation

ST starts from the top level *L-1* (where *L* is the height of the trees), and solves one local problem in order to find solutions at the roots. Each solution generates one problem at the next level until it reaches the leaves where solutions are output. Thus, the total number of local problems is,

$$N_{PROBLEMS} = 1 + \sum_{l=1}^{L-1} \#solutions(Q,l) , \quad (17)$$

where *#solutions(Q, l)* is the number of qualifying entry combinations at level *l*. An experimental study in Mamoulis and Papadias (2001) suggests that ST is CPU bound, due to the huge number of local problems and the fact that tree nodes are visited with high locality; thus, the LRU buffer serves the majority of I/O requests. Therefore, it is crucial to estimate the CPU cost of the algorithm. This depends on the cost of the local problems, all of which have the same characteristics (that is, number of variables, constraints and domain size); therefore, it is reasonable to assume that they all have approximately the same cost ($C_{PROBLEM}$). Consequently, the total CPU cost ($Cost_{CPU}$) equals the number of local problems times the cost of each problem:

$$Cost_{CPU}(\text{ST}, Q) = N_{PROBLEMS} \times C_{PROBLEM} \quad (18)$$

$N_{PROBLEMS}$ can be estimated by Equation 17 using Equation 12 for the number of solutions at each level of the tree. The only difference is that instead of object MBRs, intermediate nodes are used in Equations 14 and 16. The remaining factor is the cost $C_{PROBLEM}$. Although in the worst case (for example, extremely large intermediate nodes) each local problem is exponential ($O(C^n)$), the average $C_{PROBLEM}$ for typical situations is much lower (actually, it increases linearly with *n* and page size). Unfortunately, the nature of backtracking-based search algorithms (including forward checking) does not permit theoretical average case analysis (Kondrak & van Beek, 1997). Therefore, an empirical analysis was conducted in Mamoulis and Papadias (2001) to isolate this cost. The result of this analysis is that *the CPU-time for each local problem is linear to the number of variables n and the page size p, independently of the domain density or the structure of the graph*, and we can define:

$$C_{PROBLEM} = F \times n \times p , \quad (19)$$

Table 2. Iterator functions

Iterator	***Open***	***Next***	***Close***
ST (RJ for two inputs)	– open tree files	– return next tuple	– close tree files
SISJ (assuming that left input is the R-tree input)	– open left tree file; – construct slot index; – *open* right (probe) input; – call *next* on right input and hash results into slots; – *close* right input	– perform hash-join; – return next tuple	– close tree file; – de-allocate slot index; – hash buckets
SHJ (assuming that left input is the build input and right input the probe input)	– *open* left input; – call *next* on left and write the results into intermediate file while determining extents of hash buckets; – *close* left input; – hash results from intermediate file into buckets; – *open* right input; – call *next* on right and hash all results into right buckets; – *close* right input	– perform hash-join; – return next tuple	– de-allocate hash buckets

where F is a factor that depends on the algorithm for ST and the CPU speed and can be estimated by Equations 17, 18, 19 and the actual cost of a multiway join. The experiments of Mamoulis and Papadias (2001) suggest that this method has low average error (below 15%) for various multiway joins on synthetic datasets.

Combining ST with Pairwise Join Algorithms

Since ST is essentially a generalization of RJ, it easily can be integrated with other pairwise join algorithms to effectively process complex spatial queries. Table 2 shows how ST, SISJ and SHJ can be implemented as *iterator functions* (Graefe, 1993) in an execution engine running on a centralized, uni-processor environment that applies pipelining.

ST (RJ for two inputs) executes the join and passes the results to the upper operator. SISJ first constructs the slot index, then hashes the results of the probe (right) input into the corresponding buckets and finally performs the join, passing the results to the upper operator. SHJ does not have knowledge about the initial buckets where the results of the left join will be hashed; thus, it cannot avoid writing the results of its left input to disk. At the same time it performs sampling to determine the initial extents of the buckets. Then, the intermediate file is read and hashed to the buckets. The results of the probe input are immediately hashed to buckets. Notice that in this implementation, the system buffer is shared between at most two operators and *next* functions never run concurrently; when

Table 3. Number of plans and optimization cost parameters for different query graphs

	clique	chain	star
$comb_k$	$\binom{n}{k}$	$n\text{-}k\text{+}1$	$\begin{cases} n, k=1 \\ \binom{n-1}{k-1}, \text{otherwise} \end{cases}$
$decomp_k$	$\begin{cases} 0, 1 \le k \le 2 \\ k + \sum_{2 \le i < k-1} \binom{k}{i}, \text{otherwise} \end{cases}$	$\begin{cases} 0, 1 \le k \le 2 \\ 2, \text{otherwise} \end{cases}$	$\begin{cases} 0, 1 \le k \le 2 \\ k-1, \text{otherwise} \end{cases}$

join is executed at one operator, only hashing is performed at the upper one. Thus, given a memory buffer of M pages, the operator that is currently performing a join uses $M - K$ pages and the upper operator, which performs hashing, uses K pages, where K is the number of slots/buckets. In this way, the utilization of the memory buffer is maximized.

Optimization of Multiway Spatial Joins

Given a set of binary (for example, SISJ, SHJ) and *n*-ary (for example, ST) join operators, and the corresponding selectivity/cost estimation formulae, the spatial query optimizer aims at finding a fast execution plan. *Dynamic programming* (DP), the standard technique for relational query optimization, can also be applied for multiway spatial joins. The optimal plan for a query is computed in a bottom-up fashion from its sub-graphs. At step *i*, for each connected sub-graph Q_i with *i* nodes, DP (Figure 13) finds the best decomposition of Q_i to two connected components, based on the optimal cost of executing these components and their sizes. We assume that all join inputs are indexed by R-trees. When a component consists of a single node, SISJ is considered as the join execution algorithm, whereas if both parts have at least two nodes, SHJ is used. The output size is estimated using the size of the plans that formulate the decomposition. DP compares the cost of the optimal decomposition with the cost of processing the whole sub-graph using ST, and sets as optimal plan of the sub-graph the best alternative. Since pairwise algorithms are I/O bound and ST is CPU-bound, when estimating the cost for a query sub-plan, DP takes under consideration the dominant factor in each case.

At the end of the algorithm, *Q*.plan will be the optimal plan, and *Q*.cost and *Q*.size will hold its expected cost and size. The execution cost of dynamic programming depends on: (i) the number of relations *n*, (ii) the number of valid node

Figure 13. Dynamic programming for optimization of multiway spatial joins

```
DP(Query Q, int n) /*n = number of inputs*/
  for each connected sub-graph Ri-Rj Q2 ∈ Q of size 2 do {
      Q2.cost := Cost(RJ, Ri, Rj); /*Equation 5*/
      Q2.size := OC(Ri, Rj); /*Equation 3*/     }
  for i:=3 to n do
      for each connected sub-graph Qi ∈ Q with i nodes do {/*Find optimal plan for Qi*/
          Qi.plan := ST; Qi.cost := Cost_CPU(ST, Qi); /*Equation 18*/
          for each decomposition Qi → {Qk, Qi-k}, such that Qk, Qi-k connected do {
              if (k=1) then /*Qk is a single node; SISJ will be used*/
                  {Qk, Qi-k}.cost := Qi-k.cost + Cost(SISJ, Qk, Qi-k); /*Equation 11*/
              else /*both components are sub-plans; SHJ will be used*/
                  {Qk, Qi-k}.cost := Qk.cost + Qi-k.cost + Cost(SHJ, Qk, Qi-k); /*Equation8*/
              if {Qk,Qi-k}.cost<Qi.cost then { /*better than former optimal*/
                  Qi.plan := {Qk, Qi-k}; /*mark decomposition. as Qi's optimal plan*/
                  Qi.cost := {Qk, Qi-k}.cost; /*mark so far optimal cost of Qi*/}
          } /*decomposition*/
          /*Estimate Qi's output size from optimal decomposition*/
          Qi.size := OC(Qi.plan);
      }}
```

combinations $comb_k$ (that formulate a connected sub-graph) for each value of n, and (iii) the number of decompositions $decomp_k$ of a specific combination. Table 3 illustrates the above parameters for three special cases of join graphs. Note that combinations of 2 nodes do not have valid decompositions because they can be processed only by RJ.

The running cost of the optimization algorithm is the number of input combinations for each value of n times the number of valid decompositions plus 1 for the cost of ST:

$$Cost_{CPU}(\text{DP}, Q) = \sum_{1 \le k \le n} comb_k \cdot (1 + decomp_k) \qquad (20)$$

Equation 20 suggests that DP can be too expensive for joins with a large (for example, >10) number of inputs. For such cases, randomized algorithms can find a good (but sub-optimal) plan within limited time (Mamoulis & Papadias, 2001).

Summary

In this chapter we review some of the most significant research results related to spatial join processing. In particular, we describe: (i) binary algorithms that can be used in different cases, depending on whether the joined inputs are indexed or not; (ii) selectivity and cost estimation models; and (iii) techniques for the efficient processing of multiway joins based on integration of binary algorithms and synchronous traversal. Although we attempted to provide an extensive coverage of the literature, several issues related to spatial joins — for example, parallel join processing (Brinkhoff et al., 1996; Luo, Naughton, & Ellman, 2002) and join variants (Koudas & Sevcik, 2000; Corral et al., 2000; Böhm & Krebs, 2002; Shou et al., 2003) — were omitted due to space constraints.

There are several issues related to spatial joins that still need to be addressed. First, it is a common belief that intersection join algorithms can be straightforwardly applied for other types, like distance joins. However, practice (for example, see Corral et al., 2000; Shou et al., 2003) has already shown that direct extensions (usually of RJ) may be inefficient, and several optimizations can potentially enhanceperformance. Thus the application and optimization of different intersection algorithms to other join variants is an interesting topic of future work. Furthermore, although current systems only consider the standard "first filter, then refinement step" strategy, a spatial query processor should allow the interleaving of filter and refinement steps. For example, consider the query "find all cities *adjacent to* forests, which are *intersected* by a river" and assume that we know there are only a few rivers that intersect cities, although there are numerous such MBR pairs. Then, it would be preferable to execute the refinement step after the first join before we proceed to the next one. However, this knowledge presumes that we have accurate selectivity formulae for the refinement step, which is a difficult, open problem for future work.

References

Acharya, S., Poosala, V., & Ramaswamy, S. (1999). Selectivity estimation in spatial databases. *Proceedings of the ACM SIGMOD Conference*, (pp. 13-24).

An, N., Yang, Z., & Sivasubramaniam, A. (2001). Selectivity estimation for spatial joins. *Proceedings of the IEEE ICDE Conference*, (pp. 368-375).

Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., & Vitter, J.S. (1998). Scalable sweeping-based spatial join. *Proceedings of the VLDB Conference*, (pp. 570-581).

Beckmann, N., Kriegel, H.P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. *Proceedings of the ACM SIGMOD Conference*, (pp. 322-331).

Belussi, A., & Faloutsos, C. (1998). Self-spatial join selectivity estimating using fractal concepts. *ACM TOIS, 16*(2), 161-201.

Böhm, C., & Krebs, F. (2002). High performance data mining using the nearest neighbor join. *Proceedings of the IEEE International Conference on Data Mining*, (pp. 43-50).

Brinkhoff, T., Kriegel, H.P., & Seeger, B. (1993). Efficient processing of spatial joins using R-trees. *Proceedings of the ACM SIGMOD Conference*, (pp. 237-246).

Brinkhoff, T., Kriegel, H.P., & Seeger, B. (1996). Parallel processing of spatial joins using R-trees. *Proceedings of the ICDE Conference*, (pp. 258-265).

Brinkhoff, T., Kriegel, H.P., Schneider, R., & Seeger, B. (1994). Multi-step processing of spatial joins. *Proceedings of the ACM SIGMOD Conference*, (pp. 197-208).

Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2000). Closest pair queries in spatial databases. *Proceedings of the ACM SIGMOD Conference*, (pp. 189-200).

Faloutsos, C., Seeger, B., Traina, A., & Traina, C. (2000). Spatial join selectivity using power laws. *Proceedings of the ACM SIGMOD Conference*, (pp. 177-188).

Gaede, V. & Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys, 30*(2), 123-169.

Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys, 25*(2), 73-170.

Günther, O. (1993) Efficient computation of spatial joins. *Proceedings of the ICDE Conference*, (pp. 50-59).

Güting, R.H. (1994). An introduction to spatial database systems. *VLDB Journal, 3*(4), 357-399.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of the ACM SIGMOD Conference*, (pp. 47-57).

Haralick, R., & Elliott, G. (1981). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence, 14*, 263-313.

Huang, Y.W., Jing, N., & Rundensteiner, E. (1997a). Spatial joins using R-trees: Breadth first traversal with global optimizations. *Proceedings of the VLDB Conference*, (pp. 395-405).

Huang, Y.W., Jing N., & Rundensteiner, E. (1997b). A cost model for estimating the performance of spatial joins using R-trees. *Proceedings of the SSDBM Conference*, (pp. 30-38).

Kamel, I., & Faloutsos, C. (1993). On packing R-trees. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, (pp. 490-499).

Koudas, N., & Sevcik, K. (1997). Size separation spatial join. *Proceedings of the ACM SIGMOD Conference*, (pp. 324-335).

Koudas, N., & Sevcik, K. (2000). High dimensional similarity joins: Algorithms and performance evaluation. *IEEE Transactions in Knowledge and Data Engineering*, *12*(1), 3-18.

Kondrak Q., & van Beek, P. (1997). A theoretical evaluation of selected backtracking algorithms. *Artificial Intelligence*, *89*, 365-387.

Lo, M-L., & Ravishankar, C.V. (1994). Spatial joins using seeded trees. *Proceedings of the ACM SIGMOD Conference*, (pp. 209-220).

Lo, M-L., & Ravishankar, C.V. (1996). Spatial hash-joins. *Proceedings of the ACM SIGMOD Conference*, (pp. 247-258).

Luo, G., Naughton, J., & Ellman, C. (2002). A non-blocking parallel spatial join algorithm. *Proceedings of the ICDE Conference*, (pp. 697-705).

Mamoulis, N., & Papadias, D. (1999). Integration of spatial join algorithms for processing multiple inputs. *Proceedings of the ACM SIGMOD Conference*, (pp. 1-12).

Mamoulis, N., & Papadias, D. (2001). Multiway spatial joins. *ACM Transactions on Database Systems (TODS)*, *26*(4), 424-475.

Mamoulis, N., & Papadias, D. (2003). Slot index spatial join. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, *15*(1), 211-231.

Muralikrishna, M., & DeWitt, D. (1988). Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. *Proceedings of the ACM SIGMOD Conference*, (pp. 28-36).

Orenstein, J. (1986). Spatial query processing in an object-oriented database system. *Proceedings of the ACM SIGMOD Conference*, (pp. 326-336).

Papadias, D., Mamoulis, N., & Delis, V. (1998). Algorithms for querying by spatial structure. *Proceedings of the VLDB Conference*, (pp. 546-557).

Papadias, D., Mamoulis, N., & Theodoridis, Y. (1999) Processing and optimization of multiway spatial joins using R-trees. *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, (pp. 44-55).

Papadopoulos, A.N., Rigaux, P., & Scholl, M. (1999). A performance evaluation of spatial join processing strategies. *Proceedings of the Symposium on Large Spatial Databases (SSD)*, (pp. 286-307).

Patel, J.M., & DeWitt, D.J. (1996). Partition based spatial-merge join. *Proceedings of the ACM SIGMOD Conference*, (pp. 259-270).

Poosala, Y., & Ioannidis, Y. (1997) Selectivity estimation without the attribute value independence assumption. *Proceedings of the VLDB Conference*, (pp. 486-495).

Preparata, F., & Shamos, M. (1985). *Computational geometry*. New York: Springer.

Rotem, D. (1991). Spatial join indices. *Proceedings of the International Conference on Data Engineering (ICDE)*, (pp. 500-509).

Shou, Y., Mamoulis, N., Cao, H., Papadias, D., & Cheung, D.W. (2003). Evaluation of Iceberg Distance Joins. *Proceedings of the Eighth International Symposium on Spatial and Temporal Databases, (SSTD)*, (pp. 270-288).

Theodoridis, Y., & Sellis, T. (1996). A model for the prediction of R-tree performance. *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, (pp. 161-171).

Theodoridis, Y., Stefanakis, E., & Sellis, T. (1998). Cost models for join queries in spatial databases. *Proceedings of the ICDE Conference*, (pp. 476-483).

Endnotes

* supported by grant HKU 7149/03E from Hong Kong RGC

** also with the Data and Knowledge Engineering Group, Computer Technology Institute, Greece [http://dke.cti.gr]

*** supported by grant HKUST 6180/03E from Hong Kong RGC

1 Given a series of different layers of the same region (for example, rivers, streets, forests), its *workspace* is defined as the total area covered by all layers (not necessarily rectangular) including holes, if any.

2 RJ can be thought of as a special case of ST involving two inputs.

Section IV

Moving Objects

Chapter VIII

Applications of Moving Objects Databases

Ouri Wolfson, University of Illinois, USA

Eduardo Mena, University of Zaragoza, Spain

Abstract

Miniaturization of computing devices and advances in wireless communication and sensor technology are some of the forces propagating computing from the stationary desktop to the mobile outdoors. Some important classes of new applications that will be enabled by this revolutionary development include location-based services, tourist services, mobile electronic commerce and digital battlefield. Some existing application classes that will benefit from the development include transportation and air traffic control, weather forecasting, emergency response, mobile resource management and mobile workforce. Location management, that is, the management of transient location information, is an enabling technology for all these applications. Location management is also a fundamental component of other technologies, such as fly-through visualization, context awareness, augmented reality, cellular communication and dynamic resource discovery. Moving Objects Databases (MODs) store and manage the

location as well as other dynamic information about moving objects. In this chapter we will present the applications of MODs and their functionality. The target readership is researchers and engineers working in databases and mobile computing.

Background

In 1996, the Federal Communications Commission (FCC) mandated that all wireless carriers offer a 911 service with the ability to pinpoint the location of callers making emergency requests. This requirement is forcing wireless operators to roll out costly new infrastructure that provides location data about mobile devices. In part to facilitate the rollout of these services, in May 2000, the U.S. government stopped jamming the signals from global positioning system (GPS) satellites for use in civilian applications, dramatically improving the accuracy of GPS-based location data to 5-50 meters.

As prices of basic enabling equipment like smart cell phones, handheld devices, wireless modems and GPS devices continue to drop rapidly, the number of wireless subscribers worldwide will soar. Spurred by the combination of expensive new location-based infrastructure and an enormous market of mobile users, companies will roll out new wireless applications to recoup their technology investments and increase customer loyalty and switching costs. These applications are collectively called *location-based services.*

Emerging commercial location-based services include Mobile Resource Management (MRM) applications, such as systems for mobile workforce management, automatic vehicle location, fleet management, logistics, transportation management and support (including air traffic control). These systems use location data combined with route schedules to track and manage service personnel or transportation systems. Call centers and dispatch operators can use these applications to notify customers of accurate arrival times, optimize personnel utilization, handle emergency requests and adjust for external conditions like weather and traffic. Another example of location-based service is Location-aware Content Delivery, which uses location data to tailor the information delivered to the mobile user in order to increase relevance; for instance, delivering accurate driving directions, instant coupons to customers nearing a store or nearest resource information like local restaurants, hospitals, ATM machines or gas stations.

In addition to commercial systems, management of moving objects in location-based systems arises in the military in the context of the digital battlefield. In a

military application one would like to ask queries such as "retrieve the helicopters that are scheduled to enter region R within the next 10 minutes."

MODs, which include the management of transient location information, are an enabling technology for all the above applications. MODs are also a fundamental component of other technologies, such as fly-through visualization (the visualized terrain changes continuously with the location of the user), context awareness (the location of the user determines the content, format or timing of information delivered), augmented reality (the location of both the viewer and the viewed object determines the type of information delivered to viewer) and cellular communication.

Location management has been studied extensively in the cellular architecture context. The problem is as follows: In order to complete the connection to a cellular user *u*, the network has to know the cell id of *u*. Thus the network maintains a database of location records (*key, cell-id*), and it needs to support two types of operations: (1) Point query, when a cellular user needs to be located in order to complete a call or send a message, for example, "find the current location (cell) of the moving object with key 707-476-2276," and (2) Point update, when a cellular user moves beyond the boundary of its current cell, for example, "update the current location (cell) of the moving object with key 707-476-2276." The question addressed in the literature is how to distribute, replicate and cache the database of location records such that the two types of operations are executed as efficiently as possible. Related questions are how frequently to update, and how to search the database. Many papers have addressed this question, and two good surveys of the subject are Bhattacharya and Das (1999) and Pitoura and Samaras (2001).

However, the location management problem addressed by MODs is much broader. The main limitations of the cellular work are that the only relevant operations are point queries and updates that pertain to the current time, and they are only concerned with cell-resolution locations. For the applications we discussed, queries are often set oriented, location of a finer resolution is necessary, queries may pertain to the future or the past, and triggers are often more important than queries. Some examples of queries and triggers supported by MODs are: "during the past year, how many times was bus #5 late by more than 10 minutes at some station" (past query); "show me the taxi cabs within 1 mile of my location" (set oriented present query); "retrieve the estimated location of truck #56 tomorrow at 8 a.m." (future query); "retrieve the trucks that will reach their destination within the next 20 minutes" (set oriented future query); "send me a message when a helicopter is in a given geographic area" (trigger).

In terms of location-based-services software development, the current approach is to build a separate, location-independent management component for each application. However, this results in significant complexity and duplication of

efforts, in the same sense that data management functionality was duplicated before the development of Database Management Systems (DBMS). To continue the analogy, we need to develop location management technology that addresses the common requirements and serves as a development platform in the same sense that DBMS technology extracted concurrency control, recovery, query language and query processing, and serves as a platform for inventory and personnel application development.

In this chapter we describe the approach in building a general-purpose location management system, that is, a MOD. Such a database serves as the repository that stores and manages location as well as other dynamic information about moving objects. The main topics that will be discussed are: location technologies and applications, location modeling/management and MOD architecture and functionality.

MOD Applications

In this section we discuss the kind of applications that can be built on top of MOD technology.

- **Geographic resource discovery:** A mobile user provides its (partial) future trajectory to a service provider, and expects the answer to such queries/triggers as: "notify me when I am two miles away from a motel (in my travel direction) that has rooms available for under $100 per night." The service provider uses a MOD to store the location information of its customers and answer their queries/triggers.
- **Digital battlefield:** The dynamic location and status of the moving objects (tanks, helicopters, soldiers) in the battlefield is stored in a MOD that must answer queries and process triggers of various degrees of complexity (for example, "How many friendly tanks are in region X?").
- **Transportation (taxi, courier, emergency response, municipal transportation, traffic control, supply chain management, logistics):** In these applications the MOD stores the trajectories of the moving objects and answers such queries as: "which taxi cab is expected to be closest to 320 State Street half an hour from now" (when presumably service is requested at that address); "When will the bus arrive at the State and Oak station?" "How many times during the last month was bus #25 late at some station by more than 10 minutes?"
- **Location (or mobile) e-commerce and marketing:** In these applications, coupons and other location-sensitive marketing information are fed to

a mobile device (that presumably screens it based on the user profile and displays it selectively).

- **Mobile workforce management:** Utilities and residential/commercial service providers track their service engineers and the MOD answers such queries as: "Which service crew is closest to the emergency at 232 Hill Street?"
- **Context-awareness, augmented-reality, fly-through visualization:** In these applications the service provider feeds, in real time, the relevant information to the current location of a moving user. For example, a geologist driving through a terrain can use its handheld device to view the area she sees with the naked eye, but with additional information superimposed. The additional information is fed by the server and may include seismographic charts, images of the terrain taken at another season or notes made by other geologists about each landmark in the viewable terrain.
- **Air traffic control:** Currently commercial flights take "highways in the sky," but when free-flight (FAA, 2004) is instituted, a typical trigger to the air-traffic control MOD may be: "Retrieve the pair of aircraft that are on a collision course, that is, are expected to be less than a mile apart at some point."
- **Dynamic allocation of bandwidth in cellular network:** Cellular service providers may track their customers and may dynamically change the bandwidth allocation to various cells to satisfy changing customer density.
- **Querying in mobile environments:** A Mobile Ad-hoc Network (MANET) is a system of mobile computers equipped with wireless broadcast transmitters and receivers used for communicating within the system. Such networks provide an attractive and inexpensive alternative to the cellular infrastructures when this infrastructure is unavailable (for example, in remote and disaster areas), inefficient or too expensive to use (Haas, 1998). Knowing the location of the destination computer enables better and more reliable routing of messages. Thus, maintaining the trajectories of mobile computers in a MOD is an attractive alternative. However, in this case, the MOD is distributed among the moving objects themselves, since a centralized solution defeats the MANET purpose. Currently, commercial MOD products provide a very limited set of capabilities, and they focus on transportation, particularly fleet management systems. Companies marketing such systems include Mobitrac (MOBITRAC Inc., 2002), Qualcomm (Qualcomm Inc., 2002) and @Road (At Road Inc., 2004).

Location Technologies

Location sensing methods fall into three categories: triangulation, proximity and scene analysis. In triangulation, several signals originating from known sources are correlated by a processor, to determine the location of this processor. GPS receivers are the best known implementation of this method (Leick, 2003). Such a receiver is a special-purpose computer chip that costs less than $100 and is as small as one cm. It receives and triangulates signals from 24 satellites at 20,000 KM, and computes latitude and longitude with tennis-court-size precision. A Differential GPS is assisted by ground stations and achieves twp- to three-foot precision.

The second location sensing method is proximity, where the location of a moving object is determined to be within a small distance from a sensor. For example, RFID (AIM, 2004) tags transmit a digital response when contacted by radio signals from nearby scanning devices. Then the location of the tag is known to be very close to that of the scanning device.

The last method is scene analysis, where the location of an object with known dimensions can be determined by analyzing its image produced by a camera with a known location.

Modeling Based on Point Location Management

A fundamental capability of location management is modeling of transient location information, particularly the location of mobile devices such as cell phones, personal digital assistants, laptops, and so forth. These devices are carried by people or are mounted on moving objects such as vehicles, aircraft or vessels. The location information is updated by positioning technologies. In this section we describe a point location modeling technique; an alternative location modeling technique based on trajectory management can be found in the next section.

A straightforward approach used by existing industrial applications such as fleet management and Automatic Vehicle Location (AVL) is to model the location as follows. For each moving object, a location-time point of the form (l, t) is generated periodically, indicating that the object is at location l at time t. l may be a coordinate pair (x,y) or a cell-id. The point is stored in a database managed by a DBMS, and SQL is used to retrieve the location information.

This method is called point-location management, and it has several critical drawbacks. First, the method does not enable interpolation or extrapolation. For example, assume that a user needs to know which police officers were within one mile from the location of an emergency that occurred at 3 p.m. This information can only be retrieved for the moving objects that happened to generate a location update at 3 p.m. If an object did not generate an update at 3 p.m., then its whereabouts at that time are unknown. The problem is even more severe for extrapolation (that is, if a future location is requested); for example, "which field service employees will be closest to a customer location at 5 p.m.?" This query cannot be answered by the point-location method, even though the future location of the service personnel can be estimated by being based on current work schedules.

The second problem of the point-location method is that it leads to a critical precision/resource trade-off. An accurate picture of the precise location of moving objects would require frequent location updates that consume precious resources such as bandwidth and processing power.

Finally, a third problem of this method is that it leads to cumbersome and inefficient software development. Specifically, location based services will require the development of a vast array of new software applications. Doing so on top of existing DBMS technology has several drawbacks. First, existing DBMS' are not well-equipped to handle continuously changing data, such as the location of moving objects. The reason for this is that, in databases' areas, data are assumed to be constant unless they are explicitly modified. For example, if the salary field is $30,000, then this salary is assumed to hold (that is, $30,000 is returned in response to queries) until explicitly updated. This constant-until-modified assumption does not hold for the location of moving objects, which changes continuously. The second drawback is that location based services applications need to manage space and time information, whereas SQL is not designed and optimized for this type of queries and triggers. For example, the query "retrieve the vehicles that are inside region R always between 4 p.m. and 5 p.m." would be very difficult to express in SQL. Finally, the location of a moving object is inherently imprecise because the database location of the object (that is, the object-location stored in the database) cannot always be identical to the actual location of the object. This inherent uncertainty has various implications for database modeling, querying and indexing. For example, there can be two different kinds of answers to queries; that is, the set of objects that "may" satisfy the query, and the set that "must" satisfy the query. SQL semantics cannot account for this difference.

An interesting observation is that the point location management is used for two different cases: one in which the route of the moving object is known a priori (for example, trucking fleet management, municipal transit), and the other in which

such information is not available. For example, in location-aware advertising consumers usually cannot be assumed to provide their destination, and this is also the case for the enemy in digital battlefield applications. In other words, the information available a priori is not utilized for tracking, and it is not updated as a result of tracking.

Modeling Based on Trajectory Location Management

In this section we outline Databases fOr MovINg Objects (DOMINO's) model (Wolfson, 1999) of a trajectoryand explain how to construct it and how it solves the problems associated with point location management. Let us observe that there exist alternatives to the approach here (see, for example, Guting, Bohlen, Erwig, Jensen, Lorentzos, Schneider, & Vazirgiannis, 2000) and Sistla, Wolfson, Chamberlain, & Dao, 1997). If possible, we make use of a priori or inferred information about the destination of an object. For example, the destination can be inferred based on a motion pattern (for instance, the person travels to the office between 8 a.m. and 9 a.m.) or by accessing auxiliary information (for instance, a calendar may indicate a meeting at a given time and address).

The method proposed is called trajectory location management. In this method we first obtain or estimate the source and destination of the moving object. For example, the object starts in New York City at the intersection of 57th Street and 8th Avenue at 7 a.m. and heads for Chicago at the intersection of Oak and State Streets. Then, by using an electronic map geocoded with distance and travel-time information for every road section, a trajectory is constructed.

Electronic Maps

Before defining the trajectory, let us define the format of an electronic map. An electronic map is a relation where each tuple in the relation represents a city block; that is, the road section between two intersections, with the following attributes:

- **Polyline:** the block polyline given by a sequence of 2D *x*,*y* coordinates: $(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$. Usually the block is a straight line segment, that is, given by two (x,y) coordinates.
- **Fid:** The block id number.

The following attributes are used for geocoding; that is, translating between an *(x,y)* coordinate and an address such as "1030 North State Street" (let us assume that the even range of numbers on the block is 1000-1100, and the odd range is 997-1103):

- **L_f_add:** Left-side-from street number (in the example,1000)
- **L_t_add:** Left-side-to street number (in the example, 1100)
- **R_f_add:** Right-side-from street number (in the example, 997)
- **R_t_add:** Right-side-to street number (in the example, 1103)
- **Name:** street name
- **Type:** ST or AVE
- **Zipl:** Left side Zip code
- **Zipr:** Right side Zip code
- **Speed:** speed limit on this city block
- **One way:** a Boolean One way flag.

The following attributes are used for computing travel time and travel distance.

- **Meters:** Length of the block in meters
- **Drive Time:** Typical drive time from the one end of the block to the other, in minutes

Such maps are provided by, among others, Geographic Data Technology Co. (GDT, 2004). An intersection of two streets is the endpoint of the four block-polylines. Thus, each map is an undirected graph, with the tuples representing edges of the graph.

Dealing with Trajectories

The route of a moving object O is specified by giving the starting address or *(x,y)* coordinate (start_point), the starting time and the ending address or *(x,y)* coordinate (end_point). An external routine available in most existing Geographic Information Systems, and which we assume is given a priori, computes the shortest cost (distance or travel time) path in the map graph. This path, denoted P(O), is given as a sequence of blocks (edges); that is, tuples of the map. Since P(O) is a path in the map graph, the endpoint of one block polyline is the beginning point of the next block polyline. Thus the whole route represented by P(O) is a polyline denoted L(O). For the purpose of processing spatiotemporal

range queries, the only relevant attributes of the tuples in P(O) are Polyline and Drive Time.

Given that the trip has a starting time, for each straight line segment on L(O), we can compute the time at which the object O will arrive to the point at the beginning of the segment (using the Drive-Time attribute). This is the certain-trajectory, or c-trajectory. Intuitively, the c-trajectory gives the route of a moving object, along with the time at which the object will be at each point on the route. More formally, a c-trajectory is a sequence of straight-line segments (x_1, y_1, t_1), (x_2, y_2, t_2), ... ,(x_n, y_n, t_n) in 3-dimensional space. The c-trajectory means that when the object starts at a location having coordinates (x_1, y_1) at time t_1, it will move on a straight line at constant speed and will reach location (x_2, y_2) at time t_2, and then it will move on a straight line at constant speed and will reach location (x_3, y_3) at time t_3, and so forth. The c-trajectory is an approximation of the expected motion of the object in space and time. The reason it is only an approximation is that the object does not move in straight lines at constant speed. However, given enough straight lines, the approximation can be accurate up to an arbitrary precision. The number of line segments on the trajectory has an important implication on the performance and precision of queries and triggers. Specifically, the performance increases and the precision decreases as the number of line segments decreases. We adjust and fine-tune the number of line segments on each trajectory by using a method that has been studied in computer graphics, namely line simplification (Agarwal & Varadarajan, 2000; Douglas & Peucker, 1973).

The c-trajectory is stored in the server database and in a computer carried by the moving object. At any point in time t between t_i and t_{i+1} the server can compute the expected location of the moving object at time t. Observe that this technique solves the first problem associated with point location management, namely, trajectory location management enables both location interpolation and extrapolation. The server can compute the expected location of the moving object at any point in time between the start and end times of the trip. For example, if it is known that the object is at location (x_5, y_5) at 5 p.m. and at location (x_6, y_6) at 6 p.m., and it moves in a straight line at constant speed between the two locations, then the location at 5:16 p.m. can be computed anytime, that is, before 5:16 (extrapolation) or after (interpolation).

Finally, the trajectory (or the uncertain trajectory) is obtained by associating an uncertainty threshold u_i with the i^{th} line segment on the c-trajectory. The line segment, together with the uncertainty threshold, constitutes an "agreement" between the moving object and the server. The agreement specifies the following: The moving object will update the server if and only if it deviates from its expected location according to the trajectory by u_i or more. How does the moving object compute the deviation at any point in time? Its computer receives

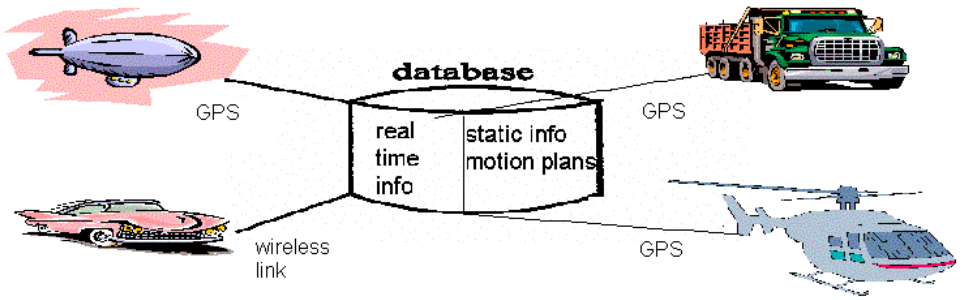
a GPS update every two seconds, so it knows its actual location at any point in time. It has the trajectory, so by interpolation it can compute its expected location at any point in time. The deviation is simply the distance between the actual and the expected location. More formally, a trajectory is a polyline $(x_1, y_1, t_1, u_1), (x_2, y_2, t_2, u_2), \ldots, (x_n, y_n, t_n, u_n)$ in 4-dimensional space.

At the server, the trajectory is maintained by revising it according to location updates from the moving object, and according to real-time traffic conditions obtained from traffic Web sites. We have developed a traffic incident model and a method of identifying the trajectories affected by a traffic incident. Observe that determining whether a trajectory is affected by a traffic incident is not a simple matter, and requires prediction capabilities. For example, suppose that according to the current information, Joe's van is scheduled to pass through highway section *x* 20 minutes from now, and suppose that a Web site currently reports a traffic jam on highway section *x*. Will Joe's expected arrival time at his destination be affected by this? Clearly it depends on whether the jam will clear by the time Joe arrives at highway section *x*. One can use historical information and various traffic models to make this prediction.

Observe that the agreement (namely the trajectory plus the uncertainty threshold) between the moving object and the server solves the second problem of point location management, namely, the tradeoff between resource/bandwidth consumption and precision has been broken. In trajectory location management, the location of a moving object can be computed with a high degree of precision using a small number of location updates or no updates at all. In particular, if the moving object is "on schedule", that is, it does not deviate from its prescribed trajectory by more than the uncertainty threshold, then no resources are consumed for updates.

Finally, let us observe that a trajectory can be constructed by being based on past motion in which an object used the point location management. Namely, the

Figure 1. Moving objects database technology



trajectory can be constructed from a set of 3D points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)$ that were transmitted by a moving object using the point location management method. One can simply connect the points along the shortest path on the map, and then associate an uncertainty u_i with line segment i. The uncertainty u_i can be bounded given the maximum speed of the object and the known times of the two GPS points immediately preceding and succeeding the i^{th} line (Pfoser & Jensen, 2000). Or, the uncertainty u_i can represent the maximum error of the GPS receiver.

MOD Architecture and Functionality

A MOD consists of static geo-spatial and temporal information, some of which is updated in real time (see Figure 1). The static information includes maps, profile information about moving objects and motion plans (for example, vehicle *v* starts at address *a* and will make deliveries at addresses *b*, *c* and *d*, in this order).

The real time updates include current location and other information fed in by sensors. This is a conceptual model. In practice, location data may be incomplete; that is, only partial trajectories may be available. Also, the database may be distributed rather than centralized. Moreover, the distribution may be among the moving objects themselves, with the extreme case being that each moving object stores its own location data.

Another generalization includes sensor data associated with the (time, location) information. For example, fuel level, images of the environment, captured information from sensors in the instrumented infrastructure (for instance, availability of a parking slot in the neighborhood broadcast to the immediate vicinity by the slot), an accident in the vicinity indicated by a deployed air-bag sensor, and so forth.

MOD Architecture

A MOD stores and manages the location as well as other dynamic information about moving objects. It can be described as a three-layer architecture, from the top to the bottom:

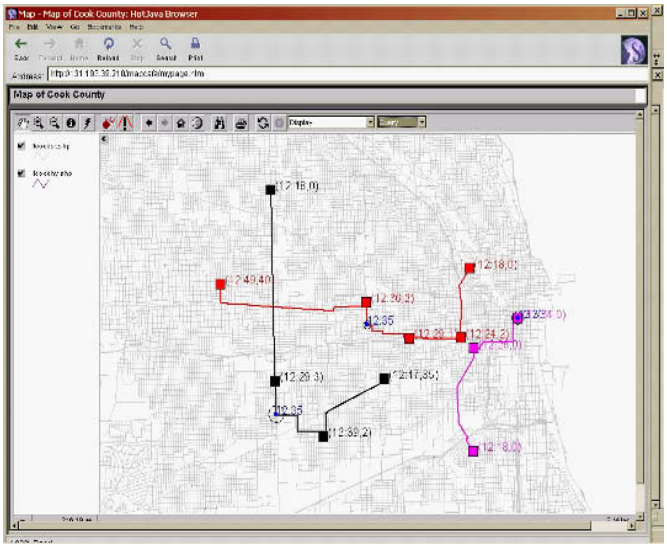
1. A software envelope that manages the dynamic aspects of the system
2. A Geographic Information System (GIS) that provides different functionalities to deal with geo-spatial information
3. A DBMS that actually stores and manages the data

Thus, we wrap a DBMS and a GIS to build a platform for location-based services application development.

Location data and other dynamic attribute values flow from moving objects to where the MOD resides (in general, in proxies or other elements of the wireless network). In the previous discussion we assumed a centralized architecture.

However, it is not realistic to think of a centralized proxy where the interesting data is sent from *all* the moving objects in the scenario (which could be crowded with moving objects!). In the same way as wireless networks are composed of a network of elements (proxies) that provide moving objects under its area with connectivity, we could think about a distributed approach to deal with location queries that involve moving objects highly distributed geographically. Thus, queries should be answered in an incremental way (area by area) and the different answers joined recursively in order to obtain a final answer. Agent technology (Milojisic et al, 1998) can be a good choice to implement an approach like the one described here: Mobile agents will carry data (subqueries and their answers) to the right place, deal with disconnections and network delays, and, therefore, fit the dynamic nature of wireless networks quite well. Some work using this technology can be found in Ilarri, Mena and Illarramendi (2002). The described distributed approach scales well, which is a very important feature in dynamic environments like wireless and *ad hoc* networks.

Figure 2. Query result showing all vehicles' locations at 12:35



MOD Functionality

Here we demonstrate the query language and user interface for the DOMINO system (Wolfson, 1999). Figure 2 illustrates the answer to the query "Where are the vehicles at 12:35?"; the screenshot shows three trajectories on a map of Chicago, each with the 12:35 location indicated by a circle. The circle is centered at the expected location of the vehicle, with the radius of the circle being equal to the uncertainty threshold. The squares of each trajectory indicate the stops of the trip. The label of each square indicates the time at which the moving object is (expected to be) at that location, at the time the moving object is stationary at that location. For example, the leftmost north-south trajectory starts at 12:18, arrives at the second stop at 12:29 and stays there for three minutes, arrives at the next stop at 12:39, and stays there for two minutes.

Figure 3 illustrates the query "Which moving object is closest to the star between 12:35 and 12:50?" The trajectories are displayed by the system, and the user marks the star on the map and enters the time interval in the pull-down menu.

The answer of the query is illustrated in Figure 4. Notice that, although the black route (left north-south route) runs closer to the given point (the star), the answer to the query issued in Figure 3 is a vehicle (indicated in Figure 4 as a circle) on the red route (east-west route). The reason is that although vehicles following

Figure 3. Which vehicle will be closest to the "star" between 12:35 and 12:50?

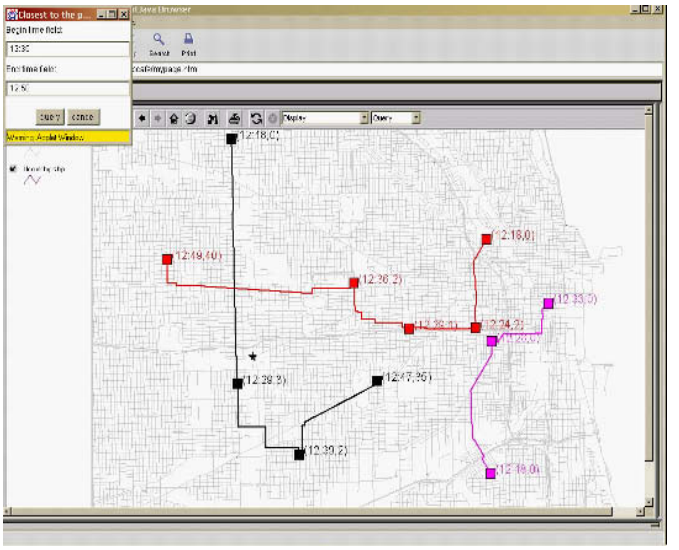


Figure 4. Answer: The vehicle (see the circle) on the red route

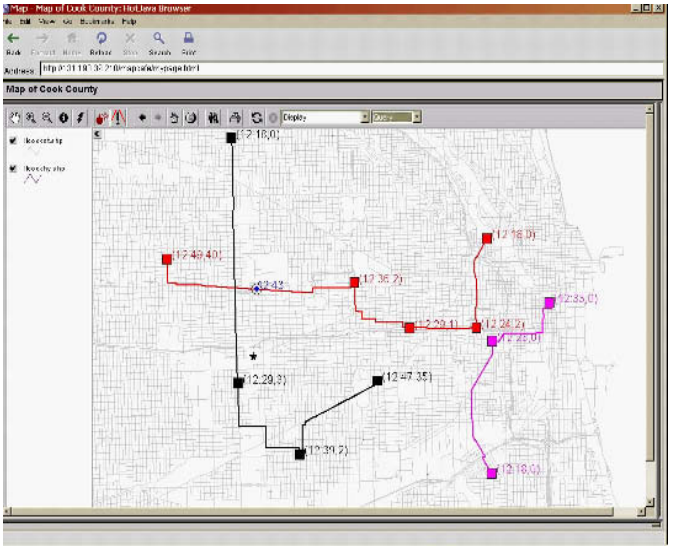
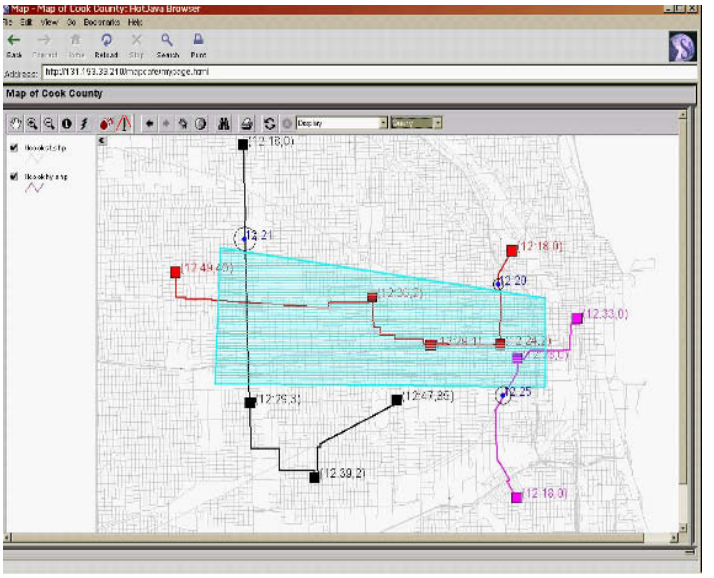


Figure 5. When will each vehicle enter the specified sector (shown as shaded area)?



the black trajectory run closest to the star, they do not do so during the specified time interval.

In Figure 5 we show another sample location query, this time is a set-oriented query that requests the time at which the vehicles in the scenario enter a given area. The screenshot illustrates the query in which the user draws the area (shaded polygon) on the screen, and the system answers by showing the circles on each trajectory. Each circle touches the polygon and is labeled with the time at which the vehicle is at the expected location.

Conclusions

We believe that the pervasive, wireless, mobile computing is a revolutionary development, and location-based services and mobile resource management are some of the initial manifestations of this revolution. In this chapter we focused on location management, which is an enabling technology for a variety of applications and technologies related to this revolution. We discussed the research issues that need to be addressed in order to make location management a plug-in component in these applications and technologies. The first of these issues is location modeling. We discussed the drawbacks of existing approaches and proposed the trajectory as a four-dimensional piece-wise linear function that captures the essential aspects of the moving object location. These aspects are two-dimensional space, time and uncertainty.

We also discussed MOD architecture and functionality, pointing out the need for considering a distributed approach for the location query processing. Then, we demonstrated the query language and user interface for the DOMINO system.

Future Trends

Concerning future work, we believe that MODs will become increasingly important and that DBMS' should be made the platform for developing moving objects applications. For this purpose, much remains to be done in terms of spatio-temporal query languages, support for rapidly changing real-time data, indexing, and distributed/mobile query and trigger processing with incomplete/imprecise location information. More specific open issues are enumerated in the following:

A comparison of the existing indexing methods (for moving objects and queries) should be made in order to choose the most appropriate for each situation. The use of these index structures for join queries should be studied as well.

- Extend the present work to handle uncertainty for moving objects that do not report their location; instead, their location could be sensed by possibly unreliable means. This is the case, for example, for enemy forces in a battlefield.
- Data mining techniques can be used to extract interesting information (for instance, trajectories of moving objects) from location data.
- Extensible and visual languages should be defined, because, in this context, the way in which users query the system and how the corresponding answers are presented are very important; textual data are not enough, as both queries and answers refer to data that should be displayed on a map.· Concerning privacy/security, new methods are needed to assure that location data are only accessed by granted applications.
- Distributed approaches must be developed to process location queries in order to fit the natural distribution of mobile scenarios. Scalability becomes an important issue with the growing number of moving objects and queries.

References

Agarwal, P.K., & Varadarajan, K.R. (2000). Efficient algorithms for approximating polygonal chains. *Discrete & Computational. Geometry*, *23*(2), 273-291.

Association for Automatic Identification and Mobility (2004). Retrieved from Radio Frequency Identification (RFID) Homepage at *www.rfid.org*

At Road Inc. (2004). Retrieved from *www.@road.com*

Bhattacharya, A., & Das, S.K. (1999, August). Lezi-Update: An information-theoretic approach to track mobile users in PCS networks. *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM99),* Seattle, WA.

Douglas, D.H., & Peucker, T.K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, *10*(2),112-122.

Federal Aviation Administration (2004). Retrieved from *www.faa.gov/freeflight*

Geographic Data Technology Co. (2004). Retrieved from *www.geographic.com*

Guting, R.H., Bohlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems Journal*, *25*(1), 1-42.

Haas, Z.J. (1998). Panel report on ad hoc networks - MILCOM'97. *Mobile Computing and Communications Review*, *2*(1).

Ilarri, S., Mena, E., & Illarramendi, A. (2002, September). Monitoring continuous location queries using mobile agents. *Proceedings of the 6th East-European Conference on Advances in Databases and Information Systems (ADBIS'02),* Bratislava, Slovakia.

Leick, A. (2003). *GPS satellite surveying*. John Wiley & Sons.

Milojisic, D., Breugst, M., Busse, I., Campbell, J., Covaci, S., Friedman, B., Kosaka, K., Lange, D., Ono, K., Oshima, M., Tham, C., Virdhagiswaran, S. & White, J. (1998, September). MASIF, the OMG Mobile Agent System Interoperability Facility. *Proceedings of Mobile Agents '98.*

MOBITRAC Inc. (2002). Retrieved from *www.mobitrac.com*

Pfoser, D., & Jensen, C.S. (1999). Capturing the uncertainty of moving objects representations. *Proceedings of the International. Symposium on Advances in Spatial Databases (SSD),* Hong Kong, China.

Pitoura, E., & Samaras, G. (2001). Locating objects in mobile computing. *IEEE Transactions on Knowledge and Data Engineering*, *13*(4).

Qualcomm Inc. (2002). Retrieved from *www.qualcomm.com*.

Sistla, A.P., Wolfson, O., Chamberlain, S., & Dao, S. (1997). Modeling and querying moving objects. *Proceedings of the International Conference on Data Engineering,* (pp. 422-432).

Wolfson, O., Sistla, A.P., Xu, B., Zhou, J., Chamberlain, S., Yesha, Y., & Rishe, N. (1999). Tracking moving objects using database technology in DOMINO. *Proceedings of the 4th International Workshop Next Generation Information Technologies and Systems (NGITS'99)*, (pp. 112-119).

Chapter IX

Simple and Incremental Nearest-Neighbor Search in Spatio-Temporal Databases

Katerina Raptopoulou, Aristotle University of Thessaloniki, Greece

Apostolos N. Papadopoulos, Aristotle University of Thessaloniki, Greece

Yannis Manolopoulos, Aristotle University of Thessaloniki, Greece

Abstract

The efficient processing of nearest-neighbor queries in databases of moving objects is considered very important for applications such as fleet management, traffic control, digital battlefields and more. Such applications have been rapidly spread due to the fact that mobile computing and wireless technologies nowadays are ubiquitous. This chapter presents important aspects towards simple and incremental nearest-neighbor search for spatio-temporal databases. More specifically, we describe the algorithms that

have already been proposed for simple and incremental nearest neighbor queries and present a new algorithm regarding that issue. Finally, we study the problem of keeping a query consistent in the presence of insertions, deletions and updates of moving objects.

Introduction

Spatio-temporal database systems aim at combining the spatial and temporal characteristics of data. There are many applications that benefit from efficient processing of spatio-temporal queries, such as: mobile communication systems, traffic control systems (for example, air-traffic monitoring), geographical information systems and multimedia applications (Wolfson, Xu, Chamberlain & Jiang, 1998; Theodoridis, Sellis, Papadopoulos & Manolopoulos, 1998). A dataset of moving objects is composed of objects whose positions change with respect to time (for example, moving vehicles). Examples of basic queries that could be posed to such a dataset include window queries, nearest-neighbor queries and join queries.

Queries that must be evaluated for a time interval $[t_s, t_e]$ are characterized as continuous (Tao, Papadias & Shen, 2002). The research conducted in access methods and query processing techniques for moving object databases are generally categorized in the following areas:

- Query processing techniques for past positions of objects (Nascimento & Silva, 1998; Pfoser, Jensen, & Theodoridis, 2000; Tao & Papadias, 2001, 2002).
- Query processing techniques for present and future positions of objects (Kollios, Gunopoulos & Tsotras, 1999; Agarwal, Arge & Erickson, 2000; Saltenis, Jensen, Leutenegger & Lopez, 2000; Procopiuc, Agarwal & Har-Peled, 2002; Lazaridis, Porkaew & Mehrotra, 2002).

We focus on the second category, where it is assumed that the dataset consists of moving point objects, which are organized by means of a Time-Parameterized R-tree (TPR-tree) (Saltenis et al., 2000). The TPR-tree is an extension of the well-known R-tree (Beckmann, Kriegel & Seeger, 1990), designed to handle object movement.

Among the different types of queries, we focus on the k nearest-neighbor query, which asks for the k closest neighbors to q during a specific time interval $[t_s, t_e]$. An interesting variation of the problem is to compute the $(k+1)$ nearest neighbor, given the result of the k-NN query. This approach requires high computation

costs unless we exploit the result of the *k*-NN query, which has been evaluated previously. Another issue is the possibility of the result of a *k*-NN query to be invalidated due to the occurrence of insertions, deletions and updates.

The ultimate objective of this chapter is to present a new algorithm for *k* nearest-neighbor queries, with the intention of overcoming the essential drawbacks characterizing the existing methods. The result produced by this algorithm contains the nearest neighbors ordered according to the increasing distance from the query.

Additionally, to the best of the authors' knowledge, incremental algorithms have not been proposed yet in the context of mobile objects. Therefore, in the sequel of this chapter we present our ideas toward this goal. Moreover, we study the problem of the consistency of the query result under the occurrence of insertions, deletions and updates.

Background

In spatial and spatio-temporal environments, the nearest-neighbor query has appeared in different forms. These forms depend on whether the query and the objects under consideration move. They are described thoroughly inconsequent sections.

Static Nearest-Neighbor Query for Static Objects

Nearest-neighbor queries were first introduced by Roussopoulos, Kelly and Vincent (1995) for static objects. In this paper the authors suggested methods to calculate static nearest neighbor queries in an R-tree. They also proposed some metrics in order to achieve ordering and pruning during the execution of the query along the R-tree.

An alternative algorithm for nearest-neighbor search was suggested by Berchtold, Ertl, Keim, Kriegel and Seidl (1990). Their method is capable of calculating static nearest-neighbor queries for static objects. The fundamental idea on which their algorithm was based was the use of Voronoi cells.

Moreover, Korn, Sidiropoulos, Faloutsos, Siegel and Protopapas (1996) suggested a multi-step algorithm with a view to answer nearest-neighbor queries for static objects. Such a multi-step search is performed by scanning the dataset several times until the desired distance is reached.

Finally, the previous method was further extended in Siedl and Kriegel (1998). The experiments performed show that the method they proposed is optimal and its performance is significantly better than the previous one.

Moving Nearest-Neighbor Query for Static Objects

The first attempt to use the nearest-neighbor search for moving query and static objects was made by Zheng and Lee (2001). Their idea is applied on an R-tree and is based on Voronoi diagrams. Furthermore, it does not seem to be applicable to other values of *k* and higher dimensions.

Moreover, an alternative idea was proposed by Roussopoulos, Kelly and Vincent (1995). They introduced an algorithm that was implemented by exploiting the method of sampling. Nevertheless, because sampling and its performance depend on the sampling rate, the results were not so satisfactory. More specifically, a low sampling rate increases the performance but may result in incorrect results, whereas a high sampling rate creates computational overhead but decreases the possibility of producing incorrect results.

Tao and Papadias (2002) proposed queries called time parameterized, since their result was time dependent. More precisely, each result contained three components <*R*,*T*,*C*>, where *R* is the set of objects currently satisfying the query, *T* is the time interval after which the result of the query will become invalid and *C* is the set of objects that will affect *R* at *T*. In order to calculate the result during an interval $[t_s,t_e]$, the algorithm recomputes the result of the query n times, where n is the number of split points. Such a recomputation induces prohibitive computational cost.

To avoid the drawbacks of large computational overhead produced by their method, Tao, Papadias and Shen (2002) proposed a completely different algorithm. The main difference between this and the previous one is that the calculation is based on a single query without having to recompute the result several times. The experiments conducted in this paper have shown that their newly proposed algorithm does indeed outperform their previous one.

Moving Nearest-Neighbor Query for Moving Objects

Initially, Benetis, Jensen, Karciauskas and Saltenis (2002) addressed this issue. The queries under consideration were simple, and reverse nearest neighbor. Nevertheless, the algorithm they proposed was restricted to returning only one nearest neighbor per query execution.

A different aspect of moving objects is the use of dual transformation, which was employed by Kolios et al. (1999). The result returned by this specific method determines the one object that came closer to the query during a predefined time interval $[t_s, t_e]$. It does not, however, return the k nearest neighbors of the query at each point of the time interval.

Incremental Nearest-Neighbor Algorithms

There are cases when after the execution of the *k*-NN query, the (*k*+1) neighbor is requested. In such a case, we can either calculate the result for the *k*+1 neighbors from the beginning or exploit the previous result and use an incremental method to determine the new answer. This idea was first addressed in Spatial Databases by Hjaltason and Samet (1995), whose algorithm could rank the objects with respect to their distance from a query object. This algorithm was implemented on a PMR quadtree and used a priority queue. Heinrich (1994) proposed a similar method that can be applied on a LSD-tree (Henrich, 1998), but it used two priority queues instead of one. Afterwards, Hjaltason and Samet (1999) applied the algorithm they presented in 1995 to the R-tree. The authors extended their initial idea and showed that their algorithm is more general and applicable not only to the PMR-tree but to the R-tree as well.

Nearest-Neighbor Query Processing

In our methods we make use of the Euclidean distance. This distance ($D_{q,o}(t)$), between a query q and an object o is given by the following equation:

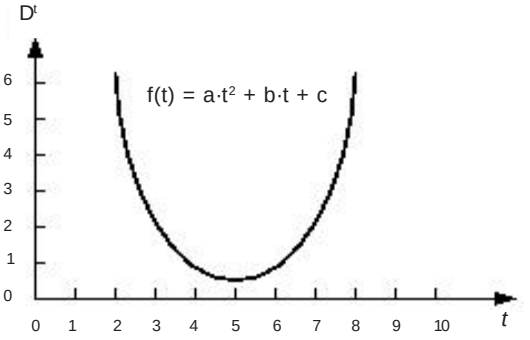
$$D_{q,o}(t) = \sqrt{c_1 \cdot t^2 + c_2 \cdot t + c_3} \qquad (1)$$

where c_1, c_2, c_3 are constants given by:

$$c_1 = (uo_x, - uq_x)^2 + (uo_y - uq_y)^2$$
$$c_2 = 2 \cdot [(o_x, - q_x) \cdot (uo_x, - uq_x) + (o_y - q_y) \cdot (uo_y - uq_y)]$$
$$c_3 = (o_x, - q_x)^2 + (o_y - q_y)^2$$

uo_x, uo_y are the velocities of the object o, uq_x, uq_y are the velocities of the query q in each dimension and (o_x, o_y), (q_x, q_y) are the reference points of the object o

Figure 1. Visualization of the distance between a moving object and a moving query



and the query q respectively. In the sequel, we assume that the distance is given by $(D_{q,o}(t))^2$. The movement of an object with respect to the query is visualized by plotting the function $(D_{q,o}(t))^2$, as illustrated in Figure 1. Assume that we have a set of moving objects O and a moving query q.

By inspecting Figure 2, we obtain the k nearest neighbors of the moving query during the time interval $[t_s, t_e]$. For example, for $k = 2$ the nearest neighbors of q for the time interval are contained in the shaded area of Figure 2. The nearest neighbors of q for various values of k, along with the corresponding time intervals, are depicted in Figure 3.

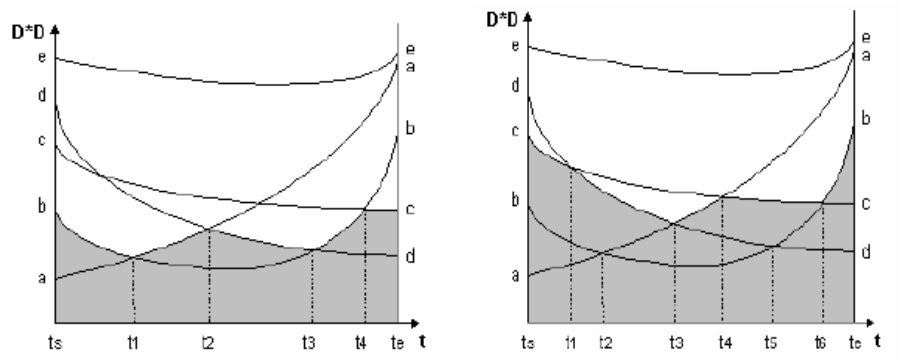
The pair of objects above each time point t_x declare the objects that have an intersection at t_x. These time points when a modification of the result is performed are called split points. Note that not all intersection points are split points. For example, in Figure 2, the intersection of objects a and c is not considered to be a split point for k =2, whereas it is a split point for k=3.

NNS Algorithm

The **NNS** algorithm consists of two parts:

- **NNS-a algorithm:** Given a set of moving objects, a moving query and a time interval, the algorithm returns the k nearest neighbors for the given interval.
- **NNS-b algorithm:** Given the k nearest neighbors, the corresponding time interval and a new moving object, the algorithm computes the new result.

Figure 2. Relative distance of objects with respect to a moving query



(a) k = 2 *(b) k = 3*

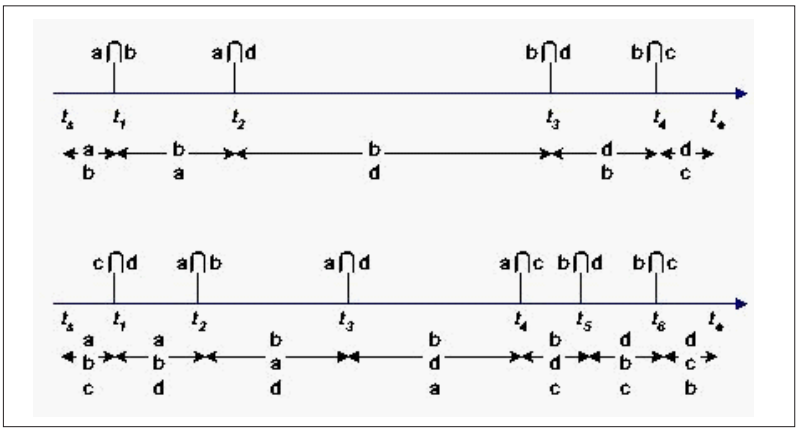
Algorithm NNS-a

We are given a moving query q, a set $\boldsymbol{O}$ of N moving objects and a time interval $[t_s, t_e]$, and the k nearest neighbors of q are requested. The target is to partition the time interval into one or more subintervals in which the list of nearest neighbors remains unchanged. Each time subinterval is defined by two time split points declaring the beginning and the end of the subinterval. During the calculation, the set $\boldsymbol{O}$ is partitioned into three subsets:

- the set $\boldsymbol{K}$, which always contains k objects that are currently the nearest neighbors of q,
- the set $\boldsymbol{C}$, which contains objects that are possible candidates for subsequent time points, and
- the set $\boldsymbol{R}$, which contains rejected objects whose contribution to the answer is impossible for the given time interval $[t_s, t_e]$.

Initially, $\boldsymbol{K} = \varnothing$, $\boldsymbol{C} = \boldsymbol{O}$, and $\boldsymbol{R} = \varnothing$. The first step is to determine the k nearest neighbors for time point t_s. By inspecting Figure 2 for $k=2$ we get that these objects are a and b. Therefore, $\boldsymbol{K} = \{a,b\}$, $\boldsymbol{C} = \{c,d,e\}$ and $\boldsymbol{R} = \varnothing$. Next, for each $o \in \boldsymbol{K}$ the intersections with objects in $\boldsymbol{K} + \boldsymbol{C}$ are determined. If there are any objects in $\boldsymbol{C}$ that do not intersect any objects in $\boldsymbol{K}$, they are removed from $\boldsymbol{C}$ and put in $\boldsymbol{R}$, meaning that they will not be considered again (Proposition 1). In our example, object e is removed from $\boldsymbol{C}$ and we have $\boldsymbol{K} = \{a,b\}$, $\boldsymbol{C} = \{c,d\}$ and $\boldsymbol{R} = \{e\}$. The currently determined intersections are kept in an ordered list in

Figure 3. Nearest neighbors of the moving query for k = 2 (top) and k=3 (bottom)



increasing time order. Each intersection is represented as (t_x, {u, v}), where t_x is the time point of the intersection and {u,v} is the objects that intersect at t_x.

Proposition 1. Moving objects that do not intersect the k nearest neighbors of the query at time t_s can be rejected.

Each intersection is defined by two objects [1] u and v. The currently determined intersection points comprise the current list of time split points. According to the example, the split point list has as follows: (t_1, {a,b}), (t_2, {a,d}), (t_x, {a,c}), (t_3,{b,d}), (t_4,{b,c}). For each intersection we distinguish between two cases:

- $u \in \boldsymbol{K}$ and $v \in \boldsymbol{K}$
- $u \in \boldsymbol{K}$ and $v \in \boldsymbol{C}$ (or $u \in \boldsymbol{C}$ and $v \in \boldsymbol{K}$)

In the first case, the current set of nearest neighbors does not change. However, the order of the currently determined objects changes, since two objects in $\boldsymbol{K}$ intersect, and therefore they exchange their position in the ordered list of nearest neighbors. Therefore, objects u and v exchange their position. In the second case, object v is inserted into $\boldsymbol{K}$ and therefore the list of nearest neighbors must be updated accordingly (Proposition 2).

Proposition 2. Let us consider a split point at time t_x at which objects o_1 and o_2 intersect. If $o_1 \in \boldsymbol{K}$ and $o_2 \in \boldsymbol{C}$, then at t_x, o_1 is the k nearest neighbor of the query.

According to the currently determined split points, the first split point is t_1, where objects *a* and *b* intersect. Since both objects are contained in ***K***, no new objects are inserted into ***K***, and objects *a* and *b* simply exchange their position. Up to this point concerning the subinterval $[t_s, t_1)$ the nearest neighbors of *q* are *a* and *b*.

We now are ready to check the next split point, which is t_2, where objects *a* and *d* intersect. Since $a \in \boldsymbol{K}$ and $d \in \boldsymbol{C}$, object *a* is removed from ***K*** and inserted into ***C***. On the other hand, object *d* is removed from ***C*** and inserted into ***K***, taking the position of *a*. Up to this point, another part of the answer has been determined, since in the subinterval $[t_1, t_2)$ the nearest neighbors of *q* are *b* and *a*. Moving to the next intersection, t_x, we see that this intersection is caused by objects *a* and *c*. However, neither of these objects is contained in ***K***. Therefore, we ignore t_x and remove it from the list of time split points. Since a new object *d* has been inserted into ***K***, we check for new intersections between *d* and objects in ***K*** and ***C***. No new intersections are discovered, and therefore we move to the next split point t_3. Currently, for the time subinterval $[t_2, t_3)$ the nearest neighbors of *q* are *b* and *d*. At t_3 objects *b* and *d* intersect, and this causes a position exchange. We move to the next split point t_4, where objects *b* and *c* intersect. Therefore, object *b* is removed from ***K*** and inserted into ***C***, whereas object *c* is removed from ***C*** and inserted into ***K***. Since *c* does not have any other intersections with objects in ***K*** and ***C***, the algorithm terminates. The final result is depicted in Figure 3, along with the corresponding result for *k=3*.

Each object $o \in \boldsymbol{K}$ is responsible for a number of potential time split points defined by the intersections of *o* and the objects contained in ***C***. Therefore, each time an object is inserted into ***K***, intersection checks must be performed with the objects in ***C***. In order to reduce the number of intersection tests, if an object was previously inserted into ***K*** and now is reinserted, it is not necessary to recompute the intersections. Moreover, according to Proposition 3, intersections at time points prior to the currently examined split point can be safely ignored.

Proposition 3. If there is a split point at time t_x, where $o_1 \in \boldsymbol{K}$ and $o_2 \in \boldsymbol{C}$ intersect, all intersections of o_2 with the other objects in ***K*** that occur at a time before t_x are not considered as split points.

Let the square of the Euclidean distance between *q* and the objects be described by the functions $(D_{u,q}(t))^2 = u_1 \cdot t^2 + u_2 \cdot t + u_3$ and $(D_{v,q}(t))^2 = v_1 \cdot t^2 + v \cdot t + v_3$ respectively. In order for the two object to have an intersection in $[t_s, t_e]$ there must be at least one value t_x, $t_s <= t_x <= t_e$ such that:

$$(u_1 - v_1) \cdot t_x^2 + (u_2 - v_2) \cdot t_x + (u_3 - v_3) = 0.$$

If $(u_2\text{-}v_2)^2 - 4 \cdot (u_1\text{-}v_1) \cdot (u_3\text{-}v_3) < 0$, then there is no intersection between u and v. If $(u_2\text{-}v_2)^2 - 4 \cdot (u_1\text{-}v_1) \cdot (u_3\text{-}v_3) = 0$ then the two objects intersect at:

$$t_x = -(u_2\text{-}v_2) \;/\; 2 \cdot (u_1\text{-}v_1).$$

Otherwise the objects intersect at two points t_x and t_y given by:

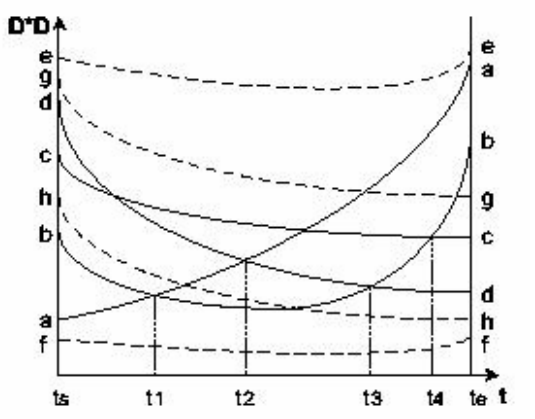
$$t_x = (-(u_2 - v_2) + \sqrt{(u_2 - v_2)^2 - 4(u_1 - v_1)(u_3 - v_3)}) / 2(u_1 - v_1)$$
$$t_y = (-(u_2 - v_2) - \sqrt{(u_2 - v_2)^2 - 4(u_1 - v_1)(u_3 - v_3)}) / 2(u_1 - v_1)$$

Algorithm NNS-b

After the execution of **NNS-a** algorithm, the *CNN-list* is formulated, which contains elements of the form: $([t_1, t_2], o_1, o_2,..., o_k)$, where $o_1,..., o_k$ are the nearest neighbors of q from t_1 to t_2, in increasing distance order. Let $\boldsymbol{S}$ be the set containing the nearest neighbors of q at any given time between t_s and t_e. Clearly, $k < |\boldsymbol{S}| < |\boldsymbol{O}|$. Assume now that we have to consider another object w, which was not known during the execution of **NNS-a**. We distinguish among the following cases, which describe the relation of w to the current answer.

- **Case 1:** w does not intersect any of the objects in $\boldsymbol{S}$ between t_s and t_e, and it is above the area of relevance. In this case w is ignored, since it cannot contribute to the nearest neighbors. The number of split points remains the same.
- **Case 2:** w does not intersect any of the objects in $\boldsymbol{S}$ between t_s and t_e, and it is completely inside the area of relevance. In this case w must be taken into account, since it affects the answer from t_s to t_e (Proposition 4). The number of split points may be reduced.
- **Case 3:** w intersects at least one object $v \in \boldsymbol{S}$ at time $t_s < t_x < t_e$, but at time t_x v is not contained in the set of nearest neighbors. In this case w is ignored, since this intersection cannot be considered as a split point, because the answer is not affected. Therefore, no new split points are generated.
- **Case 4:** w intersects at least one object $v \in \boldsymbol{S}$ at time $t_s <= t_x <= t_e$, and object v is contained in the set of nearest neighbors at time t_x. In this case w must be considered because at least one new split point is generated. We note, however, that some of the old split points may be discarded.

Figure 4. Four different cases that show the relation of a new object to the current set of nearest neighbors



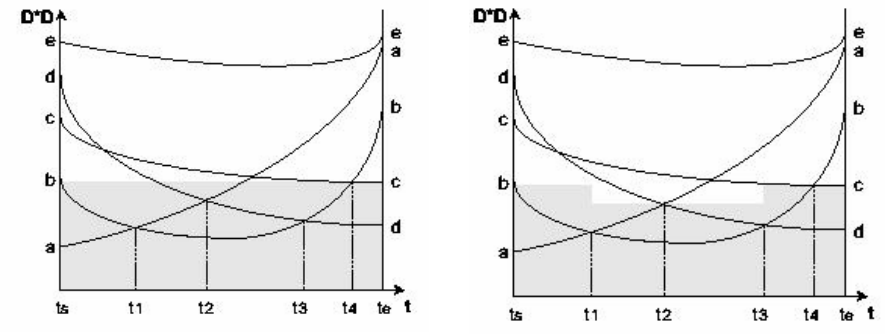
Proposition 4. Assume that a new object *w* does not intersect any of the nearest neighbors from t_s to t_e. If at time t_s its position among the *k* nearest neighbors is pos_w, then it maintains this position throughout the query duration.

The aforementioned cases are depicted in Figure 4. Object *e* corresponds to *case 1*, since it is above the area of interest. Object *f* corresponds to *case 2*, because it is completely covered by the relevant area. Although object *g* intersects some objects, the time of these intersections are irrelevant to the answer, and therefore the situation corresponds to *Case 3*. Finally, object *h* intersects a number of objects at time points that are critical to the answer and therefore corresponds to *Case 4*.

Query Processing with TPR-Trees

Let *T* be a TPR-tree built to index the underlying data. Starting from the root node of *T* the tree is searched in a Depth-First Search manner (DFS)[2]. The first phase of the algorithm is completed when $m >= k$ objects have been collected from the dataset. Tree branches are selected for descendants according to the *mindist* metric (Roussopoulos, 1995) (see Definition 1) between the moving query and bounding rectangles at time t_s. These *m* moving objects are used as input to the **NNS-a** algorithm in order to determine the result from t_s to t_e. Therefore, up to now we have a first version of the *split-list* and the *CNN-list*. However, other relevant objects may reside in leaf nodes of *T* that are not yet examined.

Figure 5. Pruning techniques



(a) one bounding rectangle *(b) many bounding rectangles*

Definition 1. Given a point p at $(p_1\ p_2,\ldots,\ p_n)$ and a rectangle r whose lower-left and upper-right corners are $(s_1\ s_2,\ldots,\ s_n)$ and $(t_1,\ t_2,\ldots,t_n)$, the distance mindist(p,r) is defined as follows:

$$\text{mindist}(p,r) = \sqrt{\sum_{j=1}^{n} \left| p_j - r_j \right|^2} \qquad \text{where } r_j = \begin{cases} s_j, & p_j < s_j \\ t_j, & p_j > t_j \\ p_j, & \text{otherwise} \end{cases}$$

In the second phase of the algorithm, the DFS continues to search the tree by selecting *possibly relevant tree branches* and discarding non-relevant ones. Every time a *possibly relevant moving object* is reached, algorithm **NNS-b** is called in order to update the *split-list* and the *CNN-list* of the result. The algorithm terminates when there are no relevant branches to examine.

Figure 5 illustrates two possible pruning techniques that can be used to determine *relevant and non-relevant tree branches and moving objects*.

Pruning Technique 1 (PT1)

In this technique we keep track of the maximum distance D_{max} between the query and the current set of nearest neighbors. In Figure 5(a) this distance is defined between the query and object b at time $t_{\{start\}}$. We formulate a *moving bounding rectangle* R centered at q with extends D_{max} in each dimension and moving with the same velocity vector as q. If R intersects a bounding rectangle E in an internal

node, the corresponding tree branch may contain objects that contribute to the answer and therefore must be examined further. Otherwise, it can be safely rejected since it is impossible to contain relevant objects. In the same manner, if a moving object o_x found in a leaf node intersects R, it may contribute to the answer, otherwise it is rejected.

Pruning Technique 2 (PT2)

Instead of using only one bounding rectangle, a set of bounding rectangles is defined according to the currently determined split points. Note that it is not necessary to consider all split points, but only those that are defined by the k nearest neighbor in each time interval. An example set of moving bounding rectangles is illustrated in Figure 5(b). Each internal bounding rectangle and moving object is checked for intersection with the whole set of moving bounding rectangles, and it is considered relevant only if it intersects at least one of them.

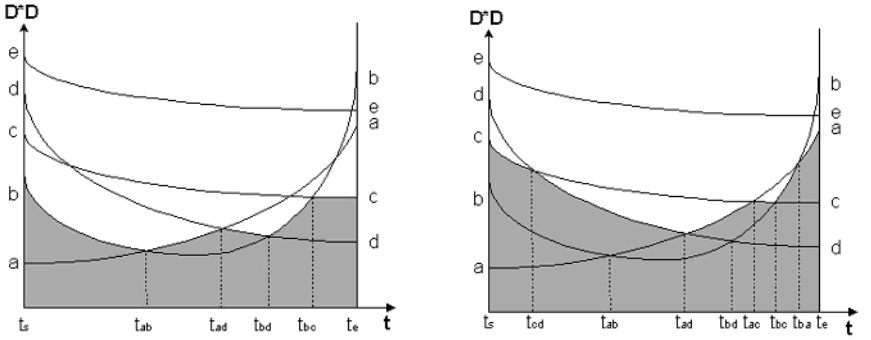
Other pruning techniques can also be determined by grouping split points in order to keep the balance. It is anticipated that PT1 will be more efficient with respect to CPU time but less efficient concerning I/O time, because the *empty space* will cause unnecessary disk accesses. On the other hand, PT2 seems to incur more CPU overhead but less I/O time, owing to the detailed pruning performed. Therefore, we define the **NNS-CON** algorithm, which can be used with either of the two pruning techniques. Thorough experimentation has been performed (Raptopoulou, Papadopoulos & Manolopoulos, 2003) for both the **NNS-CON** algorithm enabled by *PT1* described in the previous section and the **NNS-REP** algorithm, proposed by Tao et al. (2002). The main conclusion was that the proposed algorithm significantly outperforms the repetitive approach.

Incremental Nearest-Neighbor Algorithms

Figure 6 depicts two nearest-neighbor queries for k=2 and k=3 (left and right respectively). According to the way split points are defined, there are two types:

- internal split points, which are defined by an intersection of two objects, o_1 and o_2, both participating in the result, and
- external split points, which are defined by an intersection between the current k nearest neighbor and another object that is not participating in the result.

Figure 6. Visualization of a k-NN query result



(a) k = 2 *(b) k = 3*

As an example, split points at t_{ab} and t_{bd} in Figure 6(a) are internal, whereas split-points at t_{ad} and t_{bc} are external.

The challenge is, given this information, to determine the $k+1$ nearest neighbor by avoiding the query re-execution from the beginning. First, we assume that there is at least one external split-point in the split-list. The case where all split-points are internal will be discussed later. By observing Figure 6(a), it is evident that at time $t_{ad} + dt$ (where dt is a sufficiently small time interval), the $(k+1)$ neighbor is object *a*. Similarly, at time point $t_{bc} + dt$ the $(k+1)$ neighbor is object *b*. Therefore, for the time instances that correspond to external split-points, the $(k+1)$ neighbor can be determined directly from the split-list. However, this does not provide an answer for the whole interval $[t_s, t_e]$. For example, the intersection of objects *a* and *c* (Figure 6(a)) has not been recorded in the split-list.

To avoid this, if t_x is an external split-point that intersects o_1 and o_2, then we can assume that the k nearest neighbor at time $t_x + dt$ is o_2 and at time $t_x - dt$ is o_1. The time interval $[t_s, t_e]$ is therefore partitioned into two subintervals, $[t_s, t_x)$ and $(t_x, t_e]$. Note that at t_x the $(k+1)$ NN is already known (either o_1 or o_2). The method proceeds as follows:

1. New external split-points are determined for the subinterval $[t_s, t_x)$.
2. New external split-points are determined for the subinterval $(t_x, t_e]$.
3. A new split-list is generated by combining the split-points of the k-NN result with the new set of split-points.

Generating New Split-Points

There are two alternatives for the generation of new split-points. The first one determines split-points by continuously querying the TPR-tree. The second one

searches the TPR-tree only once, but it is possible to fetch irrelevant objects. We examine each alternative in detail (Figure 6(a)), assuming that the external split-point at t_{ad} is the starting point.

At time t_{ad}, objects *a* and *d* intersect. According to the first alternative, new split-points are determined one by one for the two subintervals $[t_s, t_{ad})$ and $(t_{ad}, t_e]$. Using object *a*, the TPR-tree is searched for the next possible intersection between *a* and other objects. If no intersection is found, then we deduce that object *a* is the $(k+1)$ nearest neighbor during $(t_{ad}, t_e]$. In our case, an intersection between *a* and *c* is determined at time t_{ac}. A new split-point is generated, and the same method is applied for object *c*. At time t_{cb}, objects *c* and *b* intersect, and therefore another split-point is generated. Finally, object *b* does not intersect any other object before t_e and thus no more split-points can be generated for the subinterval $(t_2, t_e]$. The method for the subinterval $[t_s, t_{ad})$ is the same. There is only one new split-point for this subinterval at time t_{cd}, denoting the intersection between objects *c* and *d*.

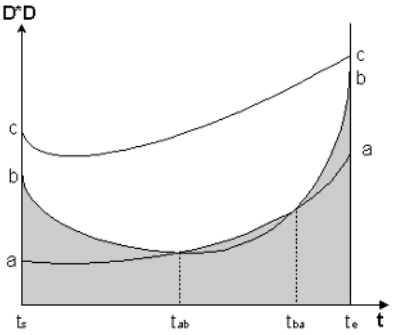
In the second alternative, we only issue two queries (one for each subinterval), and some of the objects may be discarded because they may not contribute to the result. After searching the TPR-tree for both subintervals, the algorithm returns the possible candidates for the $(k+1)$ nearest neighbor. In our example, these objects are *b*, *c* and *e*, and are participating for the $(k+1)$ nearest neighbor. Using the set of candidates, a new split-list is generated for the $(k+1)$ nearest neighbor.

Absence of External Split-Points

In the previous section we assumed that there is at least one external split-point in the split-list of the *k*-NN query result. However, we must cover the case where all split-points are internal. An example of this situation is depicted in Figure 7. In such a case, a starting point for the $(k+1)$ NN investigation cannot be determined directly from the split-list, since it is empty.

In other words, the set of objects that correspond to the *k* nearest neighbors at t_s are also the nearest neighbors at t_e. Therefore, we need at least one object that does not belong to the set of nearest neighbors. This object is determined by computing the $(k+1)$ nearest neighbor at any time instance in $[t_s, t_e]$. Then, this object is used as a candidate and either **INCNN-REP** or **INCNN-2P** can be used to provide the complete answer (Raptopoulou, Papadopoulos & Manolopoulos 2004).

Figure 7. Query result with no external split-points



Managing Insertions, Deletions, and Updates

When insertions, deletions and updates take place, all the objects participating in them must be examined to determine if they affect any of the queries. Let *w* be a moving object that is inserted, deleted or updated. There are two steps in the process of query update:

1. determination of the queries that may be affected by *w*, and
2. update of the query result.

The first step can be handled by applying indexing mechanisms to the queries. The second step involves the refinement of the set of queries determined and the update of the query result if this is necessary. In the sequel we focus on the efficient update of the query results.

Let *w* be a new moving object inserted in the database, *q* be a *k*-NN query and T a time interval $[t_s, t_e]$. Also, let *S* be the set of objects that participates in the result of *q*. Note that set *S* must contain at least *k* objects. The following cases describe the relation of *w* to the current answer:

- **Case 1:** *w* does not intersect any of the objects in ***S*** between t_s and t_e, and lies above the area of relevance. In this case, *w* is ignored, since it cannot contribute to the NNs. The number of split-points remains the same.
- **Case 2:** *w* does not intersect any of the objects in ***S*** between t_s and t_e, and lies completely inside the area of relevance. In this case, *w* must be taken

into account, since it affects the answer from t_s to t_e (Proposition 4). The number of split-points may be reduced.

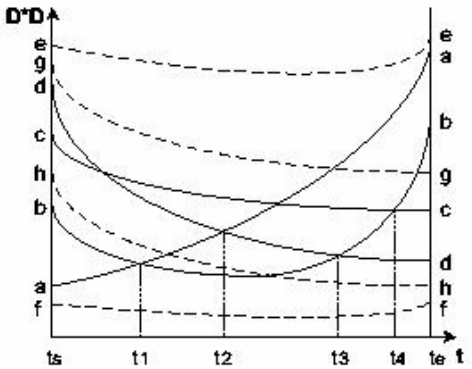
- **Case 3:** *w* intersects at least one object $v \in \boldsymbol{S}$ at time $t_s <= t_x <= t_e$, but at time t_x *v* is not contained in the set of NNs. In this case, *w* is ignored, since its intersection does not affect the answer. Therefore, no new split-points are generated.
- **Case 4:** *w* intersects at least one object $v \in \boldsymbol{S}$ at time $t_s <= t_x <= t_e$, and object *v* is contained in the set of NNs at time t_x. In this case, *w* must be considered, because at least one new split-point is generated. However, some of the previous split-points may be discarded.

The aforementioned cases are depicted in Figure 8. Object *e* corresponds to *Case 1*, since it is above the area of interest. Object *f* corresponds to *Case 2*, because it is completely covered by the relevant area. Although object *g* intersects some objects, the time of these intersections are irrelevant to the answer, and therefore the situation corresponds to *Case 3*. Finally, object *h* intersects a number of objects at time points that are critical to the answer, and therefore corresponds to *Case 4*.

Let us examine now when an object *w* is deleted and *q* is a *k*-NN query that contains it. Firstly, the split-list is scanned and object *w* is removed from the corresponding time intervals where it participates. Finally, if *w* is an object that changes its speed or direction, then we do the following:

1. Object *w* is deleted and the split-list is updated,
2. The new *k* NN of the query is computed incrementally, and
3. Object *w* is inserted and the split-list is updated.

Figure 8. Four different cases that show the relation of a new object to the current nearest neighbors



Extensive experimental evaluation has been conducted for both the previously proposed algorithms **INCNN-REP** and **INCNN-2P**, and the algorithm **REEXEC** (described in Tao et al., 2002). By observing the results, we can state that, generally, **INCNN-2P** algorithm outperforms **INCNN-REP** and **REEXEC**. Experiments conducted for insertions, deletions and updates have shown that the cost is reduced.

Future Trends

The *k*-NN algorithm that initially was presented in the first section of this chapter, may be extended in order to become more efficient, or may be applied to other data structures. More specifically, future research may focus on:

- Comparing the performance of different pruning techniques,
- Modifying the algorithm to provide the ability for incremental computation of the NNs, as the work in Hjaltason et al. (1995, 1999) suggests for static datasets,
- Providing cost estimates concerning the number of node accesses, intersection checks and distance computations.

At the second part of the chapter, the previously presented algorithm was modified to allow incremental computation. Furthermore, insertions, deletions and updates were studied thoroughly. Future research, as far as this particular aspect is concerned, may include:

- Using a priority queue in a spatio-temporal context. A problem that may arise is that objects and tree nodes may become invalid after insertions, deletions and updates of objects.
- Using algorithm **INCNN-2P** to exploit all the available external split-points.
- sutdying cost estimations for *k*-NN query processing in spatio-temporal databases.

Conclusions

Applications that rely on the combination of spatial and temporal characteristics of objects demand new types of queries and efficient query processing tech-

niques. An important query type in such a case is the *k* nearest-neighbor query for a given time interval $[t_s, t_e]$.

At the first part of this chapter, a study on efficient methods for nearest-neighbor query processing in moving-object databases was performed, and a new algorithm was proposed. The main conclusion is that the proposed algorithm significantly outperforms the repetitive approach for different parameter values.

At the second part of the chapter, we altered the preceding algorithm into incremental. The proposed algorithm **INCNN-2P** outperforms **INCNN-REP**.

Finally, we have shown that the incremental computation method can be used if a deletion or an update affects the query results. For object insertions only, the available split-list and the new object are needed to compute the new result.

References

Agarwal, P.K., Arge, L., & Erickson, J. (2000). Indexing moving points. *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS),* (pp. 175-186).

Beckmann, N., Kriegel, H.P., & Seeger, B. (1990). The R*-tree: An efficient and robust method for points and rectangles. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 322-331).

Benetis, R., Jensen, C.S., Karciauskas, G., & Saltenis, S. (2002). Nearest-neighbor and reverse nearest-neighbor queries for moving objects. *Proceedings International Symposium on Database Engineering & Applications (IDEAS),* (pp. 44-53).

Berchtold, S., Ertl, B., Keim, D.A., P. Kriegel, H.P., & Seidl, T. (1998). Fast nearest neighbor search in high-dimensional space. *Proceedings of the 14th IEEE International Conference on Data Engineering (ICDE),* (pp. 219-218).

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 47-57).

Henrich, A.(1994). A distance-scan algorithm for spatial access structures. *Proceedings of the Second ACM Workshop on Advances in Geographic Information Systems (GIS),* (pp. 136-143).

Henrich, A. (1998). The LSD^h-tree: An access structure for feature vectors. *Proceedings of the 14th IEEE International Conference on Data Engineering (ICDE),* (pp. 362-369).

Hjaltason, G.R., & Samet, H. (1995). Ranking in spatial databases. *Proceedings of the Fourth International Symposium on Spatial Databases (SSD),* (pp. 83-95).

Hjaltason, G.R., & Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems, 24*(2), 265-318.

Kollios, G., Gunopoulos, D., & Tsotras, V. (1999). On indexing mobile objects. *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS),* (pp. 261-272).

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., & Protopapas, Z. (1996). Fast nearest neighbor search in medical image databases. *Proceedings of the 22nd International Conference on Very Large Databases (VLDB),* (pp. 215-226).

Lazaridis, I., Porkaew, I., & Mehrotra, S. (2002). Dynamic queries over mobile objects. *Proceedings of the Eighth International Conference on Extending Database Technology (EDBT),* (pp. 269-286).

Nascimento, M.A., & Silva, J.R.O. (1998). Towards historical R-trees. *Proceedings of the 13th ACM symposium on Applied Computing (SAC),* (pp. 235-240).

Pfoser, D., Jensen, C.S., & Theodoridis, Y. (2000). Novel approaches to the indexing of moving object trajectories. *Proceedings of the 26th International Conference on Very Large Databases* (VLDB), (pp. 395-406).

Procopiuc, C.M., Agarwal, P.K., & Har-Peled, S. (2002). STAR-Tree: An efficient self-adjusting index for moving objects. *Proceedings of the Fourth International Workshop on Algorithm Engineering and Experiments (ALENEX),* (pp. 178-193).

Raptopoulou, K., Papadopoulos, A.N., & Manolopoulos Y. (2003). Fast nearest-neighbor query processing in moving objects databases. *GeoInformatica, 7*(2), 113-137.

Raptopoulou, K., Papadopoulos, A.N., & Manolopoulos, Y. (2004). *Incremental processing of nearest-neighbor queries in moving objects.* Technical Report, Department of Informatics, Aristotle University of Thessaloniki, Greece.

Roussopoulos, N., Kelly, S., & Vincent, F. (1995). Nearest neighbor queries. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 71-79).

Saltenis, S., Jensen, C.S., Leutenegger, S., & Lopez, M. (2000). Indexing the positions of continuously moving objects. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 331-342).

Siedl, T., & Kriegel, H. (1998). Optimal multi-step k-nearest neighbor search. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 154-165).

Tao, Y., & Papadias, D. (2001). Efficient historical R-trees. *Proceedings of the 13th International Conference on Scientific and Statistical Database Management (SSDBM),* (pp. 223-232).

Tao, Y., & Papadias, D. (2001). MV3R-tree – A spatio-temporal access method for timestamp and interval queries. *Proceedings of the 27th International Conference on Very Large Databases* (VLDB), (pp. 431-440).

Tao, Y., & Papadias, D. (2002). Time parameterized queries in spatio-temporal databases. *Proceedings of the ACM International Conference on Management of Data (SIGMOD),* (pp. 334-345).

Tao, Y., Papadias, D., & Shen, Q. (2002). Continuous nearest neighbor search. *Proceedings of the 28th International Conference on Very Large Databases* (VLDB), (pp. 287-298).

Theodoridis, Y., Sellis, T., Papadopoulos, A.N., & Manolopoulos, Y. (1998). Specifications for efficient indexing in spatio-temporal databases. *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM),* (pp. 123-132).

Wolfson, O., Xu, B., Chamberlain, S., & Jiang, L. (1998). Moving objects databases: Issues and solutions. *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM),* (pp. 111-122).

Zheng, B., & Lee, D. (2001). Semantic caching in location-dependent query. *Processing of the Sevent International Symposium on Spatial and Temporal Databases (SSTD),* (pp. 97-116).

Endnotes

1 It is assumed that an intersection is defined by two objects. If three or more objects intersect at the same point t_x the conflict is resolved by evaluating the first derivative for each object at t_x and taking the minimum value.

2 The proposed methods can also be combined with a breadth-first search-based algorithm.

Chapter X

Management of Large Moving Objects Databases:
Indexing, Benchmarking and Uncertainty in Movement Representation

Talel Abdessalem
Ecole Nationale Supérieure des Télécommunications, France

Cédric du Mouza
Conservatoire National des Arts et Métiers, France

José Moreira, Universidade de Aveiro, Portugal

Philippe Rigaux, University of Paris Sud, France

Abstract

This chapter deals with several important issues pertaining to the management of moving objects datasets in databases. The design of representative benchmarks *is closely related to the formal characterization of the properties (that is, distribution, speed, nature of movement) of these datasets;* uncertainty *is another important aspect that conditions the accuracy of the representation and therefore the confidence in query results; finally, efficient* index structures, *along with their compatibility with existing softwares, is a crucial requirement for spatio-temporal databases, as it is for any other kind of data.*

Introduction

A lot of emerging applications (traffic control, mobile computing, vehicles tracking) rely on large datasets of dynamic objects. This proliferation is encouraged by mature technologies (for example, the Global Positioning System, or GPS) that provide online information on mobile devices and enable communication between a centralized system and mobile users. Most of the services that can be provided by the system to a user are based on the location of the latter at a given instant (for instance, the case of company searching for the taxi nearest to a customer calling on his mobile phone, or a tourist in his car looking for his next hotel). But, apart from these so-called location-based services that deal with the present or future (expected) positions of objects, we can also envisage applications that study the past movements within large moving objects datasets (for data-mining purposes, for instance).

These examples illustrate some new requirements that address the core functionalities of Database Management Systems (DBMS). Indeed, we must consider new data models (as any previously proposed model falls short in representing continuous movements), new query languages and new system-level support. In this chapter we focus on the latter aspect. More specifically, we propose a survey of the following issues: benchmarking of operations on large moving objects datasets, uncertainty in trajectories representation and database indexing. Let us be more specific by shortly developing each topic.

Benchmarking

In computing, a benchmark is the result of running a set of standard tests on one component or system to compare its performance and capacity to that of other components or systems. They are designed to simulate a particular type of workload, running actual real-world programs on the system "application benchmarks," or using specially created programs that impose the workload on the component "synthetic benchmarks." Application benchmarks are meant to be representative of real-world applications and potentially give a better measure of real-world performance. On the other hand, synthetic benchmarks offer a sizeable workload of data sets and operations, allowing testing individual components (such as indexing methods or hard disks) and stressing the strengths and weaknesses of each one individually. In the spatio-temporal database context, benchmarks help to experiment with new approaches (for example, new languages or new indexing structures); they can be used to assess the effectiveness of a new system; and finally, they contribute to characterizing the properties of datasets.

Uncertainty

The representation of objects' movement is inherently imprecise (Pfoser & Jensen 1999; Trajcevski, Wolfson, Zhang & Chamberlain, 2002). Imprecision is introduced by the measurement process in the sampling of positions and by the sampling approach itself. The accuracy of measurements depends largely on the instruments and the techniques used (consider the example of the GPS). These devices are only able to capture the movement of an object by sampling its position at discrete time points and, consequently, the exact position of an object between measurements is unknown. This feature, commonly referred to as uncertainty, gives rise to several problems regarding the representation and querying of moving objects. We shall discuss in thischapter the factors that determine the imprecision, and then study (in a database perspective) ways to handle this imprecision.

Indexing

The existence of efficient access methods is one of the most important features of modern database systems. Given the huge amount of data stored in such systems, there is indeed a crucial need for structures that allow filtering of irrelevant data during query processing. B+-tree and hash-based techniques are used quite intensively in the traditional relational DBMS context. It is a well-known fact for database practitioners that a query execution plan, which relies on an index, is several orders of magnitude faster than a plan that merely scans the database. It turns out, however, that these structures fall short in supporting queries over spatial or spatio-temporal data. We shall examine in this chapter the difficult challenges raised by indexing objects whose location changes continuously, describe some representative attempts to solve the problem and discuss research perspectives in this area.

The main objectives of this chapter are to provide an up-to-date panorama of the ongoing research devoted to moving objects management systems, with a strong emphasis on the aspects that determine the efficiency and reliability of such systems. We will successively investigate benchmarking, uncertainty and indexing, giving for each topic some concrete examples, a discussion on the raised problems, some general design guidelines commonly adopted to solve the problem and finally, a presentation of future trends together with the most recent references.

Benchmarking

We begin with a short introduction on the general issue of database benchmarking, and then study some representative benchmarks for spatio-temporal databases.

Background

The two major components of a benchmark are workload specification and measurement specification.

In database or transaction processing environments, the workload specifies the data and query sets. The data sets are used to populate the database. They can be composed of real-world data or produced by a data sets generator according to specific statistical models. The query sets simulate the activity occurring in the database, such as operational and decision support transactions, or batch jobs. A transaction set driver may be used to simulate environments, where a number of users input and manage queries or transactions via a terminal or desktop computer connected to a database, with "thinking" and "keying" times interleaved.

A measurement environment must specify a metric and a reporter. By definition, a metric for a feature is an association of numeric values to feature values in such a way that the general properties of distances are verified. In a benchmarking environment, a metric is required to confer significance to the performance evaluation results. An example of a metric is the number of transactions per second (TPS). The reporter specifies how to collect all relevant traces and logs and computes indices pertinent to the specific metrics. It must provide the detailed information required to make accurate decisions on the performance capability of a system under test.

All testing processes require a well-designed execution plan. Execution plans ensure real-world environments duplication during a benchmark. The results should not depend on foreign factors(such as the hardware and software configurations) that are not related with the components in evaluation. These configurations would be barely reproducible in other environments; therefore, the results obtained would be hardly or not even comparable with the results of a similar test in different settings.

Another important feature of a benchmark is to provide a model that is representative of real-world applications with an extensible workload, made of sizeable data sets and sets of queries with varying complexities. This ensures that the model is useful and yet verifiable. Portability (it should be easy to implement on a broad range of DBMS) and simplicity (it must be understandable), also are important qualities of a benchmark.

Benchmarks for Moving Objects Databases

Benchmarking spatio-temporal systems is a novel issue; so far, emphasis has been on the development of spatio-temporal data sets generators. There are now data sets generators for simulating objects moving freely, with no or few restrictions in the movement of the objects, and generators for simulating the movement of objects for which the movement is constrained by a defined network, such as a road network.

Non-Network-Based Generators

The first spatio-temporal data sets generator for moving objects has been the so-called Generator of SpatioTemporal Data (GSTD) (Theodoridis, Silva & Nascimento, 1999a) and later, a new version was proposed introducing some important new features (Theodoridis et al., 1999b). In its current version, GSTD is a Web-based application and there also are data sets, in XML format, that can be downloaded from the Web.

The GSTD allows the generation of data about points and MBRs to be moving on a rectangular space. The space can be populated by static spatial objects that obstruct the movement of the objects. The objects, points or rectangles, are initially distributed in space according to Uniform, Gaussian or Skewed distributions. The evolution of spatial objects is directed through the definition of a set of parameters that control the duration of an object instance (which involves changes of timestamps between consecutive instances), the shift of an object (which involves changes of spatial location in terms of shift/translation of center point) and, when generating MBRs, the resizing of an object (changes in object size).

The combination of possible different distributions for these parameters allows simulating different scenarios, such as objects moving equally slow or fast and uniformly on the map, having a relatively large number of slow objects moving randomly, or having a set of objects that converge to some area of the workspace or moving to some direction (east, for example). The cardinality of the data sets is assumed to be constant during the generation process. The generated data sets are memory-less, that is, future events do not depend on past states. This framework also defines how to handle objects that fall outside the map. Three alternatives are proposed: the adjustment approach, where coordinates are adjusted to fit the workspace; the toroid approach, where the objects that traverse one edge of the workspace enter back in the opposite edge; and the radar approach, where coordinates remain unchanged, although fall beyond the workspace.

The main goal of previous approaches was to produce data sets that are rich from the statistical point of view, but a question arises of how to generate datasets representative of the behavior of real-world objects.

The Oporto framework (Saglio & Moreira, 2001) presents the specification of a realistic spatio-temporal datasets generator. The motivation for this proposal is that real-world entities do not have a chaotic behavior. They are guided by goals and they do not ignore the environment around them; that is, they are sensitive to agents favoring certain kinds of behavior and to agents inducing them to avoid other kinds of behavior. So, the generator exploits a scenario with harbors (static points), spots (regions with fixed center and changing shape, representing plankton areas or storm areas), fishing ships (moving points) and shoals of fish (fully moving regions). The fishing ships move in the direction of the most attractive shoals of fish while trying to avoid storm areas. The shoals of fish are attracted by plankton areas.

The default generation model parameters are based on information obtained from a real application for monitoring fishing activities. These parameters are organized in three classes: sizing parameters, responsible for the size of the data sets; distribution parameters, responsible for the variations in temporal and spatial distribution of moving points; and miscellaneous parameters, responsible for a few realistic features. It is proposed to use a logarithmic scale for sizing the data sets, simulating 1, 10 and 1,000 weeks of fishing activities, and two different scenarios – inshore and open-sea fishing – for the spatio-temporal distribution of data sets.

Network-Based Generators

Previous approaches do not consider applications where moving objects follow a given network. This issue has been covered by Brinkhoff) (2000a, 2000b). This generator combines real data (the network) with user-defined properties for controlling the functionality of selected object classes. Important aspects are the maximum speed of connections, the influence of other moving objects or external impacts (for example, weather conditions) on the speed, the adequate determination of origination and destination of an object and time-scheduled traffic.

The generation of datasets requires three steps: the preparation of the network, which eventually involves the conversion of files describing the nodes and the edges of the network into the file formats supported by the generator; the definition of functions and parameters according to the environment of the system under test; and the visualization of the generated datasets (they are stored in a text file) using a tool that allows visualizing the motion of the objects.

As it is argued that it would be difficult to establish an environment where all these aspects could be defined by simple user interaction or by predefined parameters, the framework only supports a few standard parameters, and the specification of elaborate behavior for moving objects requires user-defined functions and parameter settings. The implementation of the generator is based on the Java programming language. The classes are predefined; only their functionality must be adapted.

Future Trends

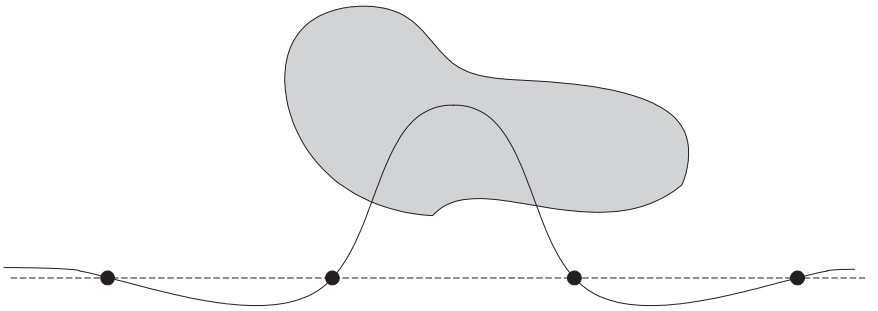
Research on benchmarking moving objects databases systems is a recent issue and, so far, the focus has been on the generation of synthetic data sets. There are now several applications available on the Web that allow generating free (Theodoridis et al., 1999b; Saglio et al., 2001) and network-based (Brinkhoff, 2000a) movements, according to a diversity of rules and control parameters. The generated movement data sets are basically sequences of temporally ordered observations, each one representing the location of a moving object at a certain instant. These data sets can be used to populate a database storing the past movement of objects, or to simulate transactions for updating the last-known location on systems concerned with present and near-future positions of moving objects.

Works on the specification of query sets is quite limited and deserves attention in the future. Apart from the benchmark database queries for location-based services (Theodoridis, 2003), there is no other systematic approach in this area. The metric that has been used in the experiments published so far was the number of disk blocks read for the evaluation of some operation. Notice that, as moving objects database systems are not commercially available yet, the experiments performed have focused exclusively on evaluation of the performance of specific access methods and algorithms for spatio-temporal operations, usually a spatio-temporal windowing or clipping. Authors use their own data and query sets and execution plans; hence, it is very difficult or even impossible to compare the performance of the different methods and techniques that have been proposed. This important issue should be considered in the near future by researchers in this area.

Uncertainty

Let us now turn our attention to the uncertainty of moving objects trajectories. As mentioned in the introduction of this chapter, the history of the objects'

Figure 1. Indeterminacy of the behavior of an object between consecutive observations



movement is inherently imprecise (Pfoser et al., 1999; Trajcevski et al., 2002). Imprecision is introduced by the measurement process in the sampling of positions and by the sampling approach itself. We begin with a short introductory part that illustrates the issue with an example and discusses the factors that determine the imprecision. We then study, in a database perspective, how to handle this imprecision.

The uncertainty of moving objects trajectories Consider the concrete case of a port authority dealing with a spread of toxic waste in the sea and querying a nautical surveillance system to know which ships have crossed the polluted zone for a specified time interval. Imagine that the ship responsible for the waste has actually followed the trajectory represented in Figure 1. The dots represent observations made during the specified time period, the shaded region represents the polluted area and the hatched line a trajectory that might have been inferred from the observations.

The hatched line does not cross the shaded region and, thus, an answer to the query based on this estimation of the trajectory would not include the guilty ship. On the contrary, an answer may include false candidates whose inferred trajectory crosses the area even though they have not actually been there.

Uncertainty of Past, Present and Future Positions

The preceding example focuses on the history of objects' movement. In general, the focus may be put on the past movement or on the future movement of objects, depending on the considered application. Different needs were identified, giving rise to two main approaches.

The first approach (Pfoser et al., 1999), focusing on past movements, addresses the needs of mining applications of spatio-temporal data: traffic mining, environment monitoring, and so forth. In this case, uncertainty is determined using the

observed successive positions of objects and some known constraints on their velocity.

In the second approach (Sistla, Wolfson, Chamberlain & Dao, 1998; Wolfson, Sistla, Xu, Zhou, Chamberlain, Yesha & Rishe 1999c; Trajcevski et al., 2002), the focus is put on the uncertainty about the future movement of objects. This approach addresses the needs of real-time applications and location-based services. Uncertainty, fixed in advance, here is used to avoid frequent updates to the database when the actual object's trajectory deviates from its representation in the database. The database is not updated as long as the object's movement deviation is less than the permitted uncertainty.

Bounding Uncertainty

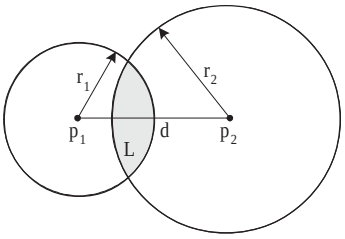
There are physical constraints on the movement of objects, allowing limiting uncertainty of their position. Particularly, the uncertainty interval for an object moving on a road is a section of the road, whereas it is an area in the considered space for an object moving freely. When it comes to future movements in a two-dimensional space, the uncertainty area is a circle centered on the expected location of the object. Each circle bounds for a given instant the permitted deviation of an object. Objects are committed to send a location update when the deviation reaches the bound.

For past movements, since the positions between two consecutive samples are not measured, the best possibility is to limit the possibilities of where the moving object could have been (Pfoser & Tryfona, 2001; Pfoser et al., 1999; Moreira, Saglio & Ribeiro, 1999).

Let us consider a moving object *m*, a time instant *t* in an interval (t_1,t_2) between two consecutive observations (p_1,t_1) and (p_2,t_2) (ee Figure 2). We denote by p_1 and p_2 the positions of the moving object at the observation time instants t_1 and t_2, respectively. At time *t*, the distance between *m* and p_1 is inferior to $r_1 = vv_{max} \times (t - t_1)$ where vv_{max} is a user-defined value standing for the maximum velocity of moving object *m*. Distance between between *m* and p_2 is inferior to $r_2 = vv_{max} \times (t_2 - t)$. So, at *t*, the moving object might be at any location within the area defined by the intersection of the two circles of radius r_1 and r_2. This is a so-called lens area (Pfoser et al., 1999) representing the set of all possible locations for a moving object at a certain time instant.

The set of all locations where a moving object might be between two consecutive observations corresponds to an ellipsis (Figure 3). This ellipsis covers all the possible lens areas between the two consecutive observations. Its parameters *a* and *b* may be computed as follows: $a = vv_{max} \times (t_2 - t_1)/2$, $c = (p_2 - p_1)/2$ and $b^2 = a^2 - c^2$.

Figure 2. A lens area for a time instant



Dealing with uncertainty in ST databases Data collected by sensor systems allow estimating the location of observed objects at any time instant between observations, assuming, for example, that the movement is linear and uniform between two consecutive observations. This semantics is not satisfactory to answer queries where uncertainty is significant. Hence, considering the uncertainty areas as described in the previous section, we propose to combine basic operations on moving objects with different semantics to provide meaningful operations in this context (Moreira et al., 2000).

Possibly Semantics

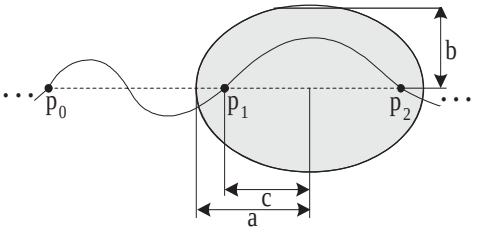
When considering the Possibly semantics, we look for the set of all possible candidates matching a query. In other words, we look for the objects that have a non-null uncertainty to match the query. This may be indicated to the query evaluator by adding a prefix "Possibly" to the chosen operation. Answers to such queries are supersets of the ideal results obtainable in an infinite precision representation. If the question is, "Which are the objects that have been in Area C," the answer includes all objects that actually have been in that area and it may also include some objects that have not. Similarly, if the question is "Give me the movement of object O within Area C," the answer includes all parts of the movement of object O for which it actually was within Area C and it may also include some parts for which the object was not there.

The set complement of this result consists in the values that definitely do not match the query predicate.

Surely Semantics

When considering the Surely semantics, only values that definitively match the query are returned. The prefix "Surely" is associated to query operations in that

Figure 3. Coverage of all possible trajectories between two consecutive observations



case. Answers using such operations are subsets of ideal results. If the question is, “Which are the objects that have been in Area C,” would be a subset of the objects that actually have been in Area C. This means that some objects that actually have been in the area as well may be omitted by the answer. Similarly, the answer to a query like "Give me the movement of object O within Area C," would include only some parts of the movement of object O for which it was actually within Area C. Some parts for which object O also was within Area C may be omitted.

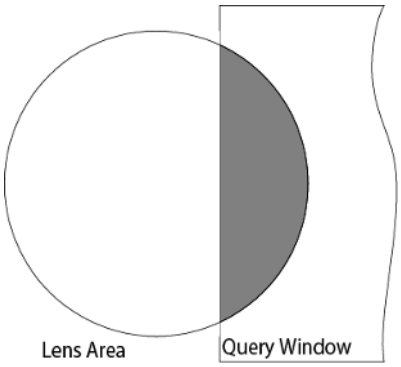
The set difference of the result of an operation using the Possibly semantics with the result of the same basic operation using the Surely semantics returns a set of values for which it is neither possible to assert that they do match the query predicate nor that they do not.

Using Probabilistic Methods

Consider now the use of methods that allow estimating the location for a moving object at any time instant, specifically, a probabilistic distribution method over the uncertainty areas. Query expressions evaluation can be augmented with a probabilistic estimation of the answer. The goal is to be able to answer queries such as, “Which are the objects that have a probability of 0.6 to be inside Area C,” or “Which were the ships in a certain area during a given time interval, with a probably of at least 30%.”

Figure 4 illustrates this query (Pfoser et al., 1999). We consider the case of a future movement in a two-dimensional space. As described in the uncertainty of moving objects trajectories section, the lens area bounds the permitted deviation of the object. If we suppose the probability distribution to be uniform in this lens

Figure 4. Probability of intersection between a lens area and a query window



area, then the object is said within a given area (query window) with a probability of 30%, if at least 30% of its lens area is concentrated within that area.

In the following query examples, prefix "Proba" is used to return the measured probability for the evaluated predicate. When a probabilistic method for the measurement of uncertainty is used, the Possibly and Surely semantics may be seen as particular cases.

Query Examples

We present here some query examples to illustrate the semantic variants proposed above. A more detailed description of relevant query operations may be found in Moreira et al. (2000).

We consider as a case study the MONICAP system for monitoring and control of fishing activities (CCMP, Inesc). The system has been used since 1992 by the Portuguese general authority for fishing activities (IGP). It continuously monitors the position of the vessels and records the history of their courses. Vessels are represented as moving objects. Static objects represent fishing areas, harbors, and so forth.

Consider the following relations:

- *FishingShips(reference:string, name:string, voyages:movement)*
- *ForbiddenAreas(name:string, geometry:polygon)*

Notice that the entire movement of a vessel is represented as an attribute in the relation *FishingShips*. The queries correspond to the kinds of questions that IGP

would like to be able to answer based on their database system. They will be expressed here using an SQL-like syntax.

Q_1: Suppose the authorities want to investigate who was responsible for a spread of waste in the sea and want to know the behavior of all vessels that could have been in the area.

```
SELECT x.name, PossiblyIn(x.voyages, :PollutedArea)
FROM FishingShips x
WHERE notEmpty(PossiblyIn(x.voyages, :PollutedArea));
```

The value *PollutedArea* is a user-defined polygon representing the area where the spread of waste has occurred. The expression *x.voyages* represents the movement of each fishing ship *x*. Operation *PossiblyIn* applied to the couple *(x.voyages, :PollutedArea)* returns, for each ship *x*, the part of its movement that *possibly* occurs inside the given area (that is, where lens areas intersect the *PollutedArea)*. The predicate *notEmpty()* determines whether this argument is empty. Here it allows selecting only the fishing ships that have a nonempty movement inside the *PollutedArea*. Query $\mathbf{Q_1}$ returns pairs of values, where the first is a name of a ship, and the second is the part of its movement that possibly occurred in the considered area.

Q_2: Authorities apply penalties when they are able to guarantee that a fishing ship has been in a forbidden area. The following query returns the name of the ships that have been in the Blue Coast reservation.

```
SELECT x.name
FROM FishingShips x, ForbiddenAreas y
WHERE y.name = “Blue Coast reservation”
AND notEmpty(SurelyIn(x.voyages, y.geometry));
```

Operation *SurelyIn* returns the parts of the movements for which it is possible to assure that a fishing ship was inside the forbidden area. Applying the predicate *notEmpty* to the result of the previous operation allows selecting the tuples of the required fishing ships.

In the following, we assume that a probabilistic method for measuring uncertainty is implemented on the MONICAP system. In that case, the previous two queries may be expressed as follows:

Q_1 :

SELECT x.name, In(x.voyages, :PollutedArea)
FROM FishingShips x
WHERE ProbaIn(x.voyages, :PollutedArea)>0

Q_2 :

SELECT x.name
FROM FishingShips x, ForbiddenAreas y
WHERE y.name = "Blue Coast reservation"
AND ProbaIn(x.voyages, y.geometry)=1

The *ProbaIn* operation, applied to its couple of arguments, returns a value between 0 and 1 corresponding to the measured probability that movement *x.voyages* may occur inside the given area.

Q_3: We search here for the fishing ships that were closer than 0.2 miles from vessel "P01" on May 28, 2000 with a probability greater than 0.6.

SELECT x.name
FROM FishingShips x, FishingShips y
WHERE x.reference = "P01"
AND y.reference ¹ x.reference
AND 0.6 < ProbaWithinDistance(During(x.voyages,á05/28/2000,05/29/2000),
During(y.voyages,á05/28/2000,05/29/2000ñ),0.2));

The first condition allows selecting the fishing ship with reference "P01." The second avoids comparing the distance of "P01" with itself. Finally, the third condition restricts the selection to a measured probability greater than 0.6 from all fishing ships that possibly were at a distance inferior to 0.2 miles from "P01" during May 28, 2000.

Future Trends

In recent years, uncertainty handling emerged as an important issue in moving object database research. Several aspects were investigated. Two complementary models were given: Pfoser et al. (1999) focused on past objects' movement when Wolfson et al. (1999c) and Trajcevski et al. (2002) treated future objects' movement. Wolfson et al. (1999b, 1999a) investigated the communication cost of uncertainty in the case of a real-time application. Pfoser et al. (2001) added fuzziness in object location and considered the case of moving objects that may change their geometry in time.

An important issue of the current research activity in this domain is the design of a probabilistic model of uncertainty. The goal is to handle more realistic (non-uniform) distributions of probability on the location of moving objects, and to be able to measure the validity of the query answers. Recent results (Cheng, Prabhakar & Kalashnikov, 2003b; Cheng, Kalashnikov & Prabhakar, 2003a; Trajcevski et al., 2003) are going toward this goal, even if they just briefly touch upon the possibility of a non-uniform distribution.

Indexing

In this section we investigate the shortcomings of traditional structures with respect to spatio-temporal databases indexing. We present some indexing techniques that have been recently proposed to overcome these limitations, and discuss the perspective of ongoing research.

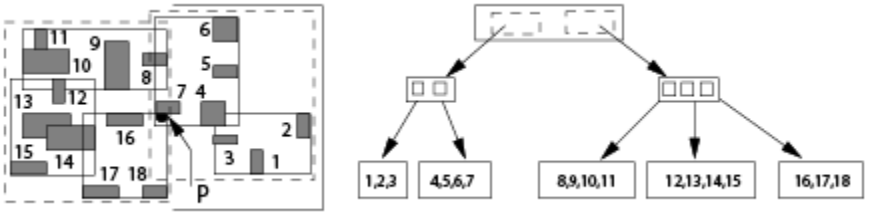
General Issues: Background

Since traditional structures cannot be used for multidimensional data indexing, during the last two decades there have been a lot of works to design efficient and reasonably simple spatial indices, like the R-tree, that can be used in existing DBMSs to support the optimization of spatial queries (see the surveys in Gaede and Guenther, (1998) and Rigaux, Scholl and Voisard, (2001)). R-trees rely on a balanced hierarchical structure in which each tree node, whether internal or leaf, is mapped onto a disk page. The R-tree (and its variants) organize rectangles (which constitute the bounding boxes of the objects in the indexed dataset) according to containment properties.

An example of an R-tree is shown in Figure 5. We assume there that a disk page can store no more than four entries (an entry is a pair composed of a point and a bounding box, and is used to navigate in the tree). Groups of four or less objects are then created and assigned to pages in the structure, based on their proximity relationships. This leads, on our example, to the groups *{1,2,3}, {4,5,6,7},* and so on. Note that grouping objects close in the space aims at minimizing the overlapping of the groups' bounding boxes and helps reduce the disk accesses during search operations. The same grouping process is applied to the bounding boxes of groups, recursively, until one obtains a single disk page, the *root* of the tree.

The R-tree properties are similar to that of the B-tree; that is, the tree is balanced, its size is logarithmic with the size of the indexed data set and its space complexity

Figure 5. R-tree indexing



is linear. It supports point and window queries. A point query, for instance, is performed by a top-down traversal of the tree, exploring at each level the subtrees whose bounding box contains the argument point. R-trees extend B-tree indexing to multi-dimensional data. It relies, however, on the important assumption that these data remain constant once stored in the database, until explicitly updated. In the presence of objects moving in the plane, this assumption is no longer valid, as it would require very continuous updates to the structure.

We chose to focus the rest of this section on a representative proposal, the Time-Parameterized R-tree (TPR-tree), and describe its design, its properties and the queries it supports. We use this first presentation as a basis for a more general discussion devoted to the challenges raised by moving objects indexingand to the issues that remain to be solved by current and future research.

A Detailed Example: TPR-Tree

The TPR-tree (Saltenis, 2000) is an extension of the R-tree that aims at indexing current and future positions (but not the past ones) of moving objects. More precisely, the index handles any object whose position is a tuple of coordinates $(x_1(t),\ldots,x_d(t))$. Each coordinate $xi(t)$ is itself a linear function of time of the form $xi(t)=xi(t_0)+v_i(t-t_0)$, where the instant t_0 defines the reference position of the indexed object and v_i is the speed of the object along the axis i.

Note that using linear function means that we consider only objects with constant speed, which is a reasonable assumption. In the following we restrict the discussion to objects moving in the 2D plane ($d=2$).

Building a TPR-Tree

Given a dataset of objects whose trajectories comply with the above representation, the TPR-tree is an R-tree-like index, built at time t_0 and valid for a time interval U. The basic idea of the structure is to construct an R-tree with time-

evolving bounding boxes. Similar to the classical R-tree, each leaf corresponds to a bounding box that contains a group of objects $\{o_1,\ldots,o_n\}$. But unlike the R-tree, the edges of a bounding box in the TPR-tree "move" so as to enclose as accurately as possible its associated group of objects during all the lifetime of the index, $[t_0, t_0+U]$.

Figure 6, inspired from Saltenis, Jensen, Leutenegger and Lopez, (2000), illustrates this intuition. We consider six moving objects and assume that the capacity of a disk page is two objects. The upper part of the figure shows the positions at time T_1, and the lower one, the positions at T_2 ($T_2>T_1$). An arrow is associated with each object, showing its direction and speed. Each column corresponds to a possible solution for indexing these data with bounding boxes. The left side shows the classical R-tree approach, with three fixed bounding boxes, *r1 ,r2 ,r3,* determined at time T_1. It appears clearly that this approach is not adapted since a bounding box at T_1 is obsolete at T_2 (for instance, r_1 no longer includes any object at T_2). This approach requires updating the bounding boxes so frequently that the maintenance of the index becomes impossible.

Consider now evolving bounding boxes, that is, rectangles whose edges move along the two axes according to a linear function of time. The choice for clustering objects in a box should now take into account not only their spatial proximity at time T_1 but also their future positions. For instance, the central column of Figure 6 shows that grouping objects by merely considering their closeness at time T_1 gives a good result for *r'1,* but not for *r'2* and *r'3*. The index greatly suffers from the increased overlapping.

The rightmost column illustrates a satisfactory grouping of our six objects, which takes in account their proximity along the whole validity interval of the index. This leads to grouping together objects that share more or less the same direction and the same speed. The comparison of the evolving boxes *r"1, r"2, r"3* at T_1 and T_2 shows the superiority of this approach over the previous one.

Figure 7 gives an example of the evolution of the bounding box between t_0 and some $t>t_0$. The position and speed of each object are known at t_0. To find the growing speed of *B* we determine the minimal and maximal speeds on both the *x* and *y* axes. For instancem the right border of the bounding box moves with a speed that corresponds to the maximal values of the projection on the *x* axis of all the objects inside the bounding box, here $v_a^x(t_0)$, and the speed of the left border corresponds to the minimal speed, here $v_b^x(t_0)$. The bounding box *B* of a TPR-tree is minimal at time t_0. Each edge of *B* moves along each axis x_i with a speed v_i, which is equal to the maximal speed along x_i of the objects contained in *B*. It follows that the minimality of the bounding box is not preserved.

In a classical R-tree, the insertion, update and deletion strategies aim at minimizing the bounding box area, the overlapping of the bounding boxes and the perimeters. The TPR-tree maintenance uses a similar approach, using the

Figure 6. Evolving bounding boxes in the TPR-tree

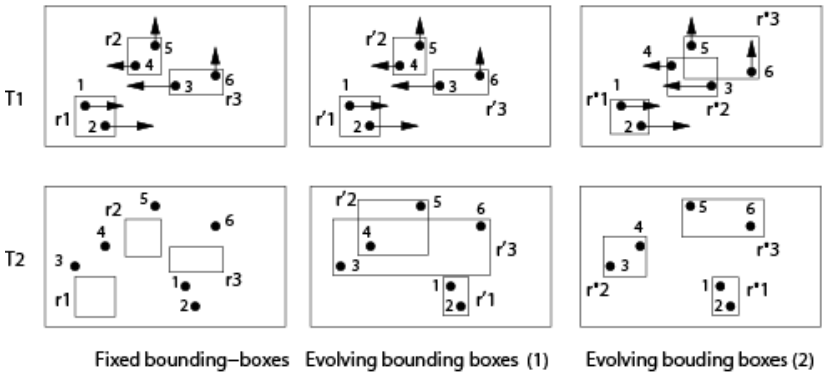
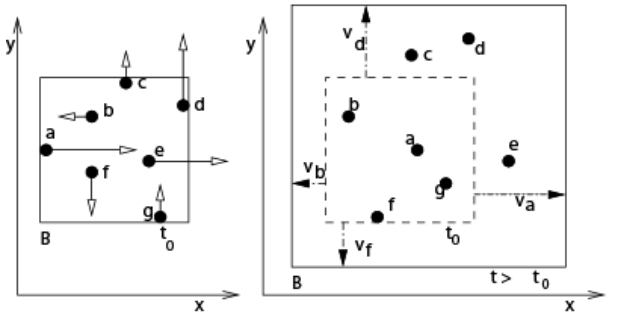


Figure 7. Managing the evolution of a bounding box



integral version of these parameters. Intuitively, the minimization always considers the (continuous) sum of the parameters' values over the validity interval of the tree. Refer to Saltenis et al. (2000) for details.

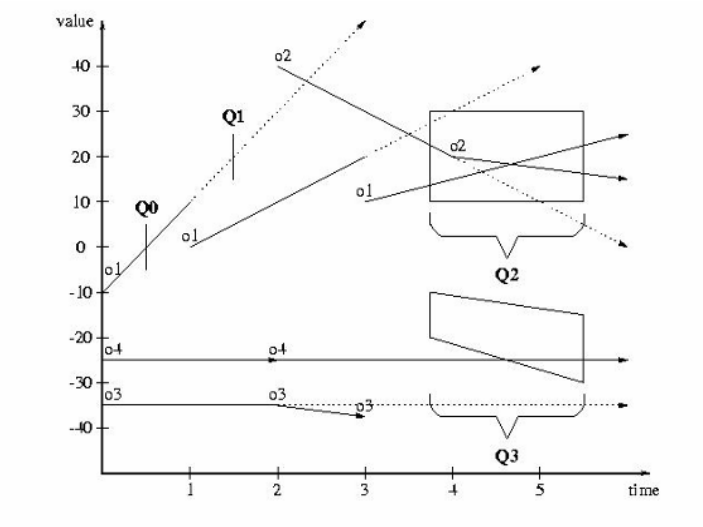
Queries Supported by the TPR-Tree

The index supports three kinds of queries:

- spatial window queries at a given instant \Delta
- spatial window queries for a time interval $[t_{beg}, t_{end}]$
- moving window queries, with values R_{beg} at t_{beg}, and R_{end} at t_{end}

Figure 8 presents examples for these three kinds of queries in a 1D-space. Consequently, a spatial window query is here at a given instant a segment (so are

Figure 8. Examples of queries in a 1D-space



R_{beg} and R_{end}). In this example, o_x denotes a moving object and we assume that $U=1$, that is, a new index is built every time unit.

1. $Q_0=([-5,5],0.5)$ and $Q_1=([15,25],1.5)$ are queries of the first kind; however, two cases appear:
   a) either the query is formulated before t_0 (e.g. Q_0), that is, before the update, and the result is consequently $\{o_1\}$, whose future trajectory is forecasted;
   b) or the answer is the empty set since the new position and speed of o_1 are now considered (for example, Q_1).
2. $Q_2=([10,30],3.75,5.5)$ is a query of the second category. If the query is expressed at a time $t<1$, the result is the empty set, since the expected trajectory for o_1 does not go through the query window, and object o_2 is unknown. If Q_2 is expressed at time $1<t<2$, the answer is o_2 since after the update the planed trajectory intersects the query window. Object o_2 is still ignored. Finally, if Q_2 is expressed at a time $t>2$, the query result is the set $\{o_1, o_2\}$.
3. Q_3 belongs to the third category and its result is the object o_4 at any time.

If we consider the query of the first category, the search with the TPR-tree is close to that of the R-tree: We select a bounding box B if the query window R overlaps *mbb*. For the queries of the second and third categories, defined on a time interval $[t_{beg},t_{end}]$, we have to select the bounding boxes that intersect the

query window between t_{beg} and t_{end}. The algorithms rely on the observation that two moving rectangles intersect if there is an instant $t \in [t_{beg}, t_{end}]$ such that their projections on each dimension intersect.

In summary, the index is valid only for a period U, and its performances deterioriate with time because the bounding boxes grow. A new index must be created whenever the validity period is over. In other words, it is not a fully dynamic structure that can be created once and adapts to the evolutions of the database.

Discussion and Future Trends

Let us now study the characteristics of the TPR-tree and evaluate to what extent it meets the general needs of spatio-temporal indexing. Throughout the discussion we will mention other techniques and recent proposals that apply to other areas, or provide an alternative approach.

Indexing Past, Current and Future Positions

The TPR-tree indexes the current and the future positions of moving objects, and is therefore relevant for the applications that track, in real time, objects equipped with a positioning system. This structure is the state-of-the-art solution for this particular situation. Its performances have been analyzed recently in Tao, Papadias and Sun, (2003), which also describes important improvements to the construction algorithms. Among the other proposals that address the same problem, but are somewhat less satisfactory, we can mention Tayeb, Ulusoy and Wolfson, (1998), which uses PMR-quadtrees to index one-dimensional moving points – (thus, two indices are needed for 2D points), and several theoretical studies, such as Kollios, Gunopolos and Tsotras (1999) and Agarwal, Arg and Erickson (2000), which, unfortunately, do not provide a practical solution.

Another quite relevant area of research concerns the indexing of past locations, which can be of interest to data-mining applications (for example, an application that analyzes the movements of a given population in a given area and for a given period). If we keep the assumption that movements can be decomposed in a finite number of consecutive time intervals, and that for each interval the speed of an object is constant, then the problem is to index a polyline in a 3D space, with "time" as a third dimension.

A straightforward solution is to build a 3D R-tree, as proposed in Theodoridis et al. (1996). Note that this assumes that the bounding boxes are bounded; that is, that each time interval associated to the segments of the polyline are closed.

Otherwise, one of the bounding boxes is of infinite size, and this raises problems. This can be compared with the TPR-tree, which considers only one segment, and bounds the time interval of interest $[t_0, t_0+U]$. Another possibility is to rely on a set of R-trees, each covering a time interval. The approach is first proposed in Nascimento et al.(1998; 1999), with a structure called the HR-tree that maintains an R-tree for each timestamp. The trees of the previous timestamps are never modified. In order to save space, the common branches of consecutive trees are stored only once. The HR-tree performs well for moving objects that frequently update their motion, but the performances are poor in range queries.

Several other proposals are worth mentioning – Tao et al. (2001); Pfoser et al. (2000); Porkaew, Lasaridis and Mehrotta, (2001); Hadjieleftheriou, Kollios, Tsotras and Gunopoulos (2002); and Saltenis et al. (2002), whose common approach is to extend R-trees to handle a polyline in a 3D space, with frequent updates that affect the last segment. In Tao et al. (2001), the authors propose an index, the MV3R-tree, which basically uses both a multi-version R-tree (Becker, Gschwind, Ohler, Seeger & Widmayer, 1996) similar to the HR-tree and a 3D R-tree built on the leaf nodes. The multi-version R-tree is expected to perform better for timeslice or short interval queries, while the 3D R-tree is more adapted for long interval queries. Another interesting structure for indexing the past trajectories of moving objects is described in Pfoser et al. (2000). It still exploits the structure of the R-tree, but tries to group together the segments from the same polyline, which allows to support new types of queries, including the so-called "trajectory queries," with predicates such "enters," "leaves," "crosses," and so forth.

Future Trends

As discussed above, so far the proposed index structures fall in one of two categories: either they index the past position, up to the current time; or they index the present and future positions, but their relevancy degrades with time. There is no structure that supports simultaneously both situations, and no fully dynamic index (that is, no index providing an automatic maintenance policy, avoiding periodic, costly re-creation). In spite of the difficulty of the problem, new research efforts are required to address these limitations.

Recently, some specific applications, with constraints that can help to reduce the complexity of the indexing problem, have attracted the attention of researchers. Among them is worth mentioning the common situation of objects moving on a constrained network such as in Pfoser et al. (2003). For instance, the authors propose to index 3D trajectories with two 2D indices, one that contains the

network (in the 2D space), and one that contains the transformed trajectories (in 1D for space and 1D for time). Another emerging area of research is the main-memory indexing of moving objects, particularly in the context of moving objects servers providing notification services to customer. In Kalashnikov, Prabhakar, Aref and Hambrusch, (2002), a simple partition of space in cells is used to index the set *S* of moving objects and determine, at each instant, and for each query *q* submitted by a user, the subset of *S* that constitutes the answer to *q*. It is argued that the capacity of computers permits to keep all the structure in main memory, and therefore avoids to design complicated mappings of these structures on disks. More generally, this suggests that emerging Web applications providing services on moving objects raise particular challenges that do not necessarily require the traditional database design approaches.

Conclusions

We investigated in this chapter several important issues pertaining to the management of moving objects datasets in databases. The design of representative benchmarks is closely related to the formal characterization of the properties (that is, distribution, speed, nature of movement) of these datasets; uncertainty is another important aspect that conditions the accuracy of the representation and therefore the confidence in query results. Finally, efficient index structures, along with their compatibility with existing software, is a crucial requirement for spatio-temporal databases, as it is for any other kind of data.

The common properties of all the issues considered in this chapter are their strong impact on the representation of data and the way they determine the implementation of both the operations and the data structures that support the evaluation of queries. Indeed, as suggested by the previous discussion, one can envisage many possible applications with quite different features. It is more than likely that the techniques used to manage a database of mobile phone users, a database of cars moving on a road network or a database of airplanes moving freely in a 3D space will strongly or partly differ because of the differents speeds, movement constraints (network-based or not) and behavior. All the aspects (benchmark, uncertainty, indexing) covered, as well as some others (implementation and semantics of database operators, for instance), are affected by these specificities.

We therefore expect in the forthcoming years many other new results, and many improvements to the state-of-the-art solutions that have been established so far.

References

Agarwal, P.K., Arge, L., & Erickson, J. (2000). Indexing moving points. *Proceedings of the ACM Symposium on Principles of Database Systems*, (pp. 175-186).

Becker, B., Gschwind, S., Ohler, T., Seeger, B., & Widmayer, P. (1996). An asymptotically optimal multiversion B-tree. *VLDB Journal*, *5*(4), 264-275.

Brinkhoff, T. (2000a). A framework for generating network-based moving objects. *Proceedings of the International Conference on Scientific and Statistical Databases (SSDMB)*, (pp. 253-255).

Brinkhoff, T. (2000b). Generating network-based moving objects. *Proceedings of the International Conference on Scientific and Statistical Databases (SSDBM)*, (pp. 253-255).

Cheng, R., Kalashnikov, D.V., & Prabhakar, S. (2003a). Evaluating probabilistic queries over imprecise data. *Proceedings of ACM SIGMOD International Conference on Management of Data*, (pp. 551-562).

Cheng, R., Prabhakar, S., & Kalashnikov, D.V. (2003b). Querying imprecise data in moving object environments. *Proceedings of the 19th IEEE International Conference on Data Engineering*, (pp. 723-725).

Gaede, V., & Guenther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, *30*(2), 170-231.

Hadjieleftheriou, M., Kollios, G., Tsotras, V.S., & Gunopoulos, D. (2002). Efficient indexing of spatio-temporal objects. *Proceedings of the International Conference on Extending Data Base Technology*, (pp. 251-268).

Kalashnikov, D.V., Prabhakar, S., Aref, W., & Hambrusch, S. (2002). Efficient evaluation of continuous range queries on moving objects. *Proceedings of the International Conference on Databases and Expert System Applications (DEXA)*, (pp. 731-740).

Kollios, G., Gunopolos, D., & Tsotras, V.J. (1999). On indexing mobile objects. *Proceedings of the ACM Symposium on Principles of Database Systems*, (pp. 261-272).

Moreira, J., Ribeiro, C., & Abdessalem, T. (2000). Query operations for moving objects database systems. *Proceedings of the 8th International Symposium on Advances in Geographic Information Systems (ACMGIS-00)*, (pp. 108-114).

Moreira, J., Saglio, J.M., & Ribeiro, C. (1999). Representation and manipulation of moving points: An extended data model for location estimation. *Journal of Cartography and Geographic Information Systems*, *26*(2), 109-123.

Nascimento, M.A., & Silva, J.R.O. (1998). Towards historical r-trees. *Proceedings of the ACM International Symposium on Applied Computing*, (pp. 235-240).

Nascimento, M.A., Silva, J.R.O., & Theodoridis, Y. (1999). Evaluation for access structures for discretely moving points. *International Workshop on Spatio-Temporal Database Management (STDBM'99)*, LNCS 1678.

Pfoser, D., & Jensen, C.S. (1999). Capturing the uncertainty of moving-object representations. *Computer Science, 1651*, 111-132.

Pfoser, D., & Jensen, C.S. (2003). Indexing of network-constrained moving objects. *Proceedings of the International Symposium on Geographic Information Systems*, (pp. 25-32).

Pfoser, D., & Tryfona, N. (2001). Capturing fuzziness and uncertainty of spatiotemporal objects. *Computer Science*, *2151*, 112.

Pfoser, D., Jensen, C.S, & Theodoridis, Y. (2000). Novel approaches in query processing for moving objects. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, (pp. 395-406).

Porkaew, K., Lasaridis, I., & Mehrotta, S. (2001). Querying mobile objects in spatiotemporal databases. *Proceedings of the International Symposium on Spatial and Temporal Databases (SSTD)*, (pp. 59-78).

Rigaux, P., Scholl, M., & Voisard, A. (2001). *Spatial databases.* Morgan Kaufmann.

Saglio, J.M., & Moreira, J. (2001). Oporto: A realistic scenario generator for moving objects. *GeoInformatica*, *5*(1), 71-93.

Saltenis, S., & Jensen, C.S. (2002). Indexing of moving objects for location-based services. *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, (pp. 463-472).

Saltenis, S., Jensen, C.S., Leutenegger, S.T., & Lopez, M.A. (2000). Indexing the positions of continuously moving objects. *Proceedings of the ACM SIGMOD Symposium on the Management of Data*, (pp. 331-342).

Sistla, P., Wolfson, O., Chamberlain, & Dao, S. (1998). Querying the uncertain position of moving objects. *Computer Science, 1399*, 310.

Tao, Y., & Papadias, D. (2001). The MV3R-Tree: A spatial-temporal access method for timestamp and interval queries. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, (pp. 431-440).

Tao, Y., Papadias, D., & Sun, J. (2003). The TPR*-tree: An optimized spatio-temporal access method for predictive queries. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, (pp. 790-801).

Tayeb, J., Ulusoy, O., & Wolfson, O. (1998). A quadtree based dynamic attribute indexing method. *Computer Journal*, 41, 185-200.

Theodoridis, T., Silva, J.R.O., & Nascimento, M.A. (1999a). On the generation of spatiotemporal datasets. *Computer Science, 1651*, 147-164.

Theodoridis, T., Silva, J.R.O., & Nascimento, M.A. (1999b). On the generation of spatiotemporal datasets. *Proceedings of the International Conference on Large Spatial Databases (SSD'99)*, (pp. 147-164).

Theodoridis, Y. (2003). Ten benchmark database queries for location-based services. *The Computer Journal*, *46*(6), 713-725.

Theodoridis, Y., Vazirgiannis, M, & Sellis, T. (1996). Spatio-temporal indexing for large multimedia applications. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, (pp. 441-448).

Trajcevski, G. (2003). Probabilistic range queries in moving objects databases with uncertainty. *Proceedings of the Third ACM International Workshop on Data Engineering for Wireless and Mobile Access*, (pp. 39-45).

Trajcevski, G., Wolfson, O., Zhang, F., & Chamberlain, S. (2002). The geometry of uncertainty in moving objects databases. *Proceedings of the Eighth International Conference on Extending Database Technology*, LNCS, (vol. 2287, pp. 233-250).

Wolfson, O., Jiang, L., Sistla, A.P., Chamberlain, S., Rishe, N., & Deng, M. (1999a). Databases for tracking mobile units in real time. *Computer Science, 1540*, 169-186.

Wolfson, O., Sistla, A.P., Chamberlain, S., & Yesha, Y. (1999b). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, *7*(3), 257-387.

Wolfson, O., Sistla, A.P., Xu, B., Zhou, J., Chamberlain, S., Yesha, Y., & Rishe, N. (1999c). Tracking moving objects using database technology in DOMINO. *Next Generation Information Technologies and Systems*, 112-119.

Section V

Data Mining

Chapter XI

Spatio-Temporal Prediction Using Data Mining Tools

Margaret H. Dunham, Southern Methodist University, Texas, USA

Nathaniel Ayewah, Southern Methodist University, Texas, USA

Zhigang Li, Southern Methodist University, Texas

Kathryn Bean, University of Texas at Dallas, USA, USA

Jie Huang, University of Texas Southwestern Medical Center, USA

Abstract

The spatio-temporal prediction problem requires that one or more future values be predicted for time series input data obtained from sensors at multiple physical locations. Examples of this type of problem include weather prediction, flood prediction, network traffic flow, and so forth. In this chapter we provide an overview of this problem, highlighting the principles and issues that come to play in spatio-temporal prediction problems. We describe some recent work in the area of flood prediction to illustrate the use of sophisticated data mining techniques that have been

examined as possible solutions. We argue the need for further data mining research to attack this difficult problem. This chapter is directed toward professionals and researchers who may wish to engage in spatio-temporal prediction.

Introduction

Forecasting future values for systems that contain both spatial and temporal features (spatio-temporal) is extremely complex. As an example, consider the problem of predicting precipitation at one location. The amount of previous rainfall in areas close to the target certainly affects this forecast. However, there are many other factors (temperature, time of day, wind direction, wind speed, and so forth) that impact the rainfall prediction. The area of spatio-temporal prediction has been the focus of much research in recent years (Deutsch, & Ramos, 1986; Dougherty, Corne, & Openshaw, 1997; Jothityangkoon, Sivapalan, & Viney, 2000; Kelly, Clapp, & Rodriguez, 1998; Pokrajac, & Obradovic, 2001; Roddick, Hornsby, & Spiliopoulou, 2000; Singh, Chaplain, & McLachlan, 1999). Due to the extreme complexity of predicting these future values, common practice is to utilize domain experts with extensive experience in both forecasting and the problem domain itself. For example, for flood prediction, the *National Oceanic and Atmospheric Administration (NOAA)* actually employs specialists whose job is to understand the history and specifics of predicting floods on one river. A different domain expert may be hired for a different river. Due to the extensive use of domain experts, spatio-temporal prediction is extremely expensive, and due to the complexity of the nature of the problems, prediction accuracy is often low.

In this chapter we argue for more data mining research into the development of sophisticated modeling, machine learning and prediction tools that can assist domain experts in solving spatio-temporal prediction problems. The interesting challenge is to see if these models can use all the data available without the need of expert intervention. This could be useful, since experts in the particular domain of interest – flood prediction, networks – often may not be experts in these data mining tools, which are drawn from statistics, genetic modeling, algorithmic heuristics and much more. But creating models general enough to be understood by non-experts has proven to be a difficult balancing act, especially when these complex models use many parameters and when there are many types of data to consider.

In this chapter we first introduce the problem and some previous research and solutions. Finally we summarize and make recommendations for future work.

Problem Definition

Spatio-temporal prediction is widely used in many diverse applications, such as environmental protection, flood control, atmosphere surveillance, waste disposal management, traffic management and so on. The general discrete time spatio-temporal prediction problem can be defined as:

Given a collection of spatial locations where time series data is collected at discrete points in time, the spatio-temporal prediction problem is to predict a future value at one of the locations.

Thus, we can state the spatio-temporal prediction problem as one of predicting one or more future values of time series data. Although we could examine a continuous time spatio-temporal prediction, most work has been done looking at a discrete time system where data is collected at regular intervals. For example, the Minnesota Department of Transportation collects freeway data at monitoring stations in the Twin Cities metropolitan area every 30 seconds. (www.dot.state.mn.us/tmc/trafficinfo/data). Example 1 is based on the Tao Dataset of sea surface temperature readings in the Pacific(http://ingrid.ldgo.columbia.edu/SOURCES/.IGOSS/.TOGA-TAO.cdf/.dataset_documentation.html).

Example 1. There are n temperature sensor stations located at buoys at various sites in the ocean. Each station measures the temperature on the surface of the ocean every day at noon. The spatio-temporal prediction problem here would be to predict the future temperature at noon for one or more of these stations given the historical time series data of previous temperature readings. Notice that in this example, a major problem is predicting the flow of the water.

Analysis of spatio-temporal systems is complex, since it consists of a large amount of irregular outcomes that incorporate space and time factors. The challenges of spatio-temporal prediction problems include:

- **Scalability:** Due to the vast amount of data that exists (infinite), any solutions must be able to scale well.
- **Dynamic Model:** The nature of spatio-temporal prediction problems is that the model used to predict a time series value today may not accurately predict it in the future.Any solution must adapt to the changing environment.
- **Variable Models:** Even though one application domain may be modeled similarly from location to location, the exact model will differ. For example, a Neural Network used to predict rainfall in Dallas will not work in Houston. We do not want to create a modeling technique for each specific location.

Rather, we want to develop generic modeling techniques that can be applied to any location.

- **High Dimensionality:** These spatio-temporal prediction problems contain many different parameters that affect the model.
- **Hidden Relationships:** Not only are there many parameters, we may not even know what all of the parameters are.
- **Unknown Spatial Influence:** Many spatio-temporal prediction problems contain unknown spatial relationships. For example, the exact movement of weather patterns is not known precisely. The further into the future the prediction is being made, the more problematic the spatial influence becomes.

We are not arguing that more sophisticated data mining techniques can solve all these problems, but they can help with some – particularly the dynamic and hidden relationships issues.

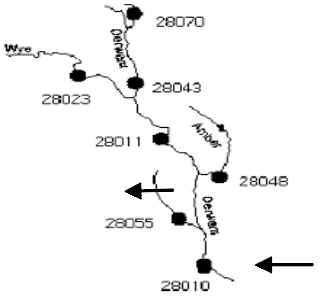
Flood Prediction Example

Flood prediction (forecasting) is an example of a spatio-temporal prediction problem whose solution can be addressed because the problem is automatically simplified due to the nature of the problem itself; that is, predicting a flood (or alternatively, a water level or flow value) at a particular point in a river has a well defined spatial aspect, namely, the flow of the river and the lay of the land. Figure 1 illustrates this aspect**.** This figure shows the Serwent Catchmentas provided by the British National River Flow Archive. The catchment of a river is the geographic area into which any precipitation will go into the river and upstream river water will flow into the river downstream. We don't need to worry about sensor data obtained for spots outside the catchment. In addition, we know the general direction in which water will flow within a catchment.

While flood prediction simplifies the spatial influence issue, it does not eliminate it completely. Looking at Figure 1, we see that sensor readings at location 28043 definitely impact those at 28010, but we do not know what the exact influence is. Certainly it would be safe to assume that the impact is somewhat less than the readings at location 28055. But how much? Another issue here is the temporal lag between the readings. The time lag between the influence of a water level reading at 28043 is probably greater than that at 28055, but the actual values vary.

There are many common spatio-temporal prediction problems similar to flood prediction. Traffic engineers examine the flow of traffic on highways to predict traffic delays and determine where best to spend funds to upgrade roadways.

Figure 1. Derwent catchment of Upper Derwent River



Network traffic engineers similarly examine flow of packets between sites to predict routing delays and prevent network downtime. Similar spatio-temporal prediction problems include electric flow in electric grids and water usage in public water systems.

Other spatio-temporal problems may not exactly fit into the flood prediction paradigm, but may be simplified by adding spatial influence assumptions to help develop solutions to the more general problem. When looking at predicting ocean water temperature, the movement of water may be approximated based on knowledge by experts as to the normal movements of ocean water.

Solving the Spatio-Temporal Prediction Problem

In this section we briefly review some previous research in the area of spatio-temporal prediction. We classify the work into counting models, stochastic models, neural networks and Markov models. This survey is not meant to be exhaustive, but illustrative.

Counting Models

We view counting models to be algebraic formulations used to capture complex issues such as spatial influence, temporal lag, high dimensionality and hidden relationships. This is a common approach with current weather prediction and flood prediction systems.

Environment studies like flood prediction (Burnash, 1996; Comet, 2000), distribution of ice channels and distribution of hydrological parameters in oceans (AARI, 2004) can be approximated, and a prediction can be made by creating a deterministic mathematical counting model. A scientist, based on computation fluid dynamics equations, can design a model that simplifies the relationship among hydrological parameters. This work was demonstrated by Kathryn Bean, one of this chapter's co-authors, for modeling the distribution of ice channels in the Kara Sea while employed at the Artic and Antarctic Research Institute in Russia.

By far, the most common approaches to solving the flood prediction problem are based on the use of counting mathematical models. One model is created for each flood prediction site, and it attempts to capture all of the unique features of the catchment at that site. These include such things as river structure upstream, flow rate and soil absorption. Values collected by sensory input include such things as water level upstream, temperature, humidity, time of day and rainfall at various points in the catchment.

The National Weather Service, part of NOAA, uses an approach based on the *Sacramento Soil Moisture Accounting Model*(NOAA, 2004). This technique predicts water levels by measuring rainfall in the catchment and estimating runoff and soil absorption. To use the model, at least 20 different parameter values have to be estimated by the domain expert (Comet, 2000). Over time, the model is adjusted by modifying these parameter estimates.

A recent survey of flood forecasting has summarized some of the current work in applying counting models to flood forecasting (Julien, Molnar, Johnson, & Combs, 1998).

Stochastic Models

Stochastic models have also been used to address the spatio-temporal prediction problem. While studying how to manage livestock's waste in a watershed, Cressi and Majure (1997) tried to design a spatio-temporal prediction model mainly based upon spatial statistics analysis. The whole area was divided into smaller grid surfaces, and their spatial characteristics were summed. To capture the temporal characteristics, they used a "three-day area of influence." In other words, data collected at an upper stream more than three days ago would not be considered to affect the lower stream's data. This model did generate a good prediction, unfortunately at the price of a "large variation of the predicted values" with a little modification of the input data. Although this overfitting problem in their work was attributed to the low sampling density in space and time, it is rational to suspect that spatial statistics, with the use of a straightforward time-window assumption, is not sophisticated enough for a reliable spatio-temporal prediction.

Deutsch et al. (1986) took a similar approach, but in an opposite direction, to address spatio-temporal-related hydrological research. For temporal correlation they constructed a time series model at each concerned location. For the spatial correlation between each pair of locations, they introduced a very simple neighboring matrix – if two locations were within a pre-defined threshold distance, the corresponding entry in the neighboring matrix would be set as 1; otherwise 0. The role of this matrix was to help merge the spatial correlation and facilitate the computation. However, a simplification for spatial correlation like this may miss other important non-distance factors and make the model incomplete. For instance, the probability of flooding at a certain position along the river does not only depend on how far away it is from the upper stream location where there was heavy rain, but also on other factors, like the saturation of the soil, the forest coverage percentage, the cross-section area, gradient of the river bed, the turning angle and so on. It is even possible that the flooding is involved with some hidden reasons that are not thoroughly understandable or are simply hard to be measured and assessed.

In order to perform prediction for a dynamic spatio-temporal system, a model needs to be constructed to incorporate both time and space variability. *AutoRegressive Integrated Moving Average (ARIMA)* time series modeling is one of the popular modeling techniques taking temporal aspects into account. Space variation is usually added statistically to it to reflect the dependence of the system outcome on relative direction as well as distance between locations (Box, & Jenkins, 1970). ARIMA is a powerful model for both stationary and nonstationary time series. The autoregressive portion represents the deviation of the current value of a stochastic process from its mean at time t with a linear aggregate of p previous values of the process and a random *a* drawing from a fixed distribution, which is assumed to be Normal and having mean zero, so called "white noise." Moving average models express the deviation linearly on q number of previous random *a* drawing (Box et al., 1970). Both models describe the stationary process; nonstationary summation is added to model the difference of the process from stationary (Box et al., 1970, 1994; Wei, 1989).

In the spatio-temporal problem, the space lag also needs to be incorporated into the model. *Space-Time AutoRegressive Integrated Moving Average (STARIMA)* is one popular model of this type. It uses a spatial hierarchical ordering of the neighbors of each site and a sequence of *NXN* weighting matrices for N locations to model the influence that the different locations have on a given site (Pfeifer, & Deutsch, 1980a). STARIMA has been widely used on various spatio-temporal problem domains, such as hydrologic modeling (Deutsch et al., 1986) and crime analysis (Pfeifer et al., 1980b).

Various other statistical approaches have been used to model spatio-temporal data by extending geostatistics or spatial statistics with temporal effects. Waller,

Carlin,, Xia and Gelfand (1997) built hierarchical spatio-temporal mapping of the disease rate on an existing Bayesian spatial model fitting in *Markov Chain Monte Carlo (MCMC)* implementation. Stroud, Muller and Sanso (2001) proposed a Bayesian method for analyzing a dataset that is discrete in time and continuous in space. The model, which was tested with tropical rainfall levels and Atlantic ocean temperatures, was to find a "mean function at each time as a locally weighted mixture of regression surfaces," and then incorporate the temporal variability by allowing the regression coefficients to change through time. Handcock and Wallis (1994) studied northern United States global warming trends using a Bayesian model of space time *Kriging*, which is a geostatistical approach to modeling for predicting unknown values from data observed at known locations while spatial dependence over time may vary.

By using the ARIMA technique, one can predict the current water level (or water flow) value as the sum of the products of the historical values multiplied on some coefficients, plus the products of independent "shock" (random variable with zero means) multiplied on the same coefficients. The hardest part of this modeling technique is finding the size of these two sets and choosing suitable coefficients to adequately predict Mother Nature. It is impossible to consider several types of time series – rainfall, temperature and other variables – simultaneously as input parameters for ARIMA (Wei, 1989). A contiguous time series, free of noise, is the only way to use this technique.

Artificial Neural Networks

Artificial Neural Networks have been used successfully (Dunham, 2003). *Neural Networks (NN)* are made up of several layers, including an input layer and an output layer. As input data is propagated through the layers of the model, it is mapped to an output. A neural network can "learn" to accurately map a given input to a given output by self-adjusting the parameters in its layers. Unlike counting models, neural networks allow forecasters to make predictions without fully understanding the input-output relationship.

A neural network has several advantages over a stochastic model like ARIMA. A neural network, after training, allows forecasters to obtain a prediction easily and quickly. Several types of time series as input parameters can be used simultaneously (Openshaw, Kneale, Corne, & See, 1998). A neural network can give adequate predictions while running on noisy datasets.

A neural network approach has some disadvantages, such as possible training/convergence errors, overfitting, bad interpretability and so on. One of them relates to the uncertainty of creating an optimal neural architecture. How to choose the optimal number of input parameters for ARIMA is well known;

however, based on our research, only empirical methodologies determine the optimal input domain of a neural network model for spatio-temporal prediction.

Neural networks are attractive for the flood prediction problem because they reduce the need for a precise understanding of the complex relationship between the input and the output. A number of researchers have done extensive work to apply neural networks on flood prediction. Openshaw et al. (1998) provide a comprehensive report of an effort to compare the performance of different types of neural networks to conventional and other statistical flood models. They aim to find the relationship between various time-lagged inputs from this site and an output at a later time. Their experiments demonstrate that a global approach using one neural network to model all the input data was not practical because most of the data was concentrated around low river levels. Instead, they classify the input into different groups - called hydrograph types - depending on whether the river levels are rising, falling or flat. With these models, they determine that neural networks perform better than the statistical and conventional models.

Markov Models and Variants

The Markov chain model is widely used to model spatio-temporal systems, such as computer I/O requests (Oly, & Reed, 2002), hydrological observations (Yapo, Sorooshian, & Gupta, 1993), robot behavior (Goldberg, & Mataric, 1999), and so forth. In this section, the Markov chain model and some of its variations are discussed in applications of modeling spatio-temporal systems.

A stochastic process is a *Markov process* when it satisfies the Markov property. Isaacson (1976) states that "a stochastic process $\{X_k\}$, K = 1,2,… with state space S = {1, 2, 3…} is said to satisfy the *Markov property* if for every n and all states $i_1, i_2, \ldots i_n$ it is true that $P[X_n = i_n | X_{n-1} = i_{n-1}, X_{n-2} = i_{n-2}, \ldots, X_1 = i_1] = P[X_n = i_n | X_{n-1} = i_{n-1}]$," where P is the transition probability. Simply speaking, the Markov property declares that the transition probability from current state i_{n-1} to next state i_n depends only on the current state of the process and has nothing to do with the earlier states in the history of the process.

A Markov process is called *Markov chain* if the state space is countable or finite. A Markov chain model, which we will call *Markov model (MM)* in the rest of this chapter, is constructed with states and transitions that can be visualized as a weighted directgraph with collection of *m* vertices, *S*, and directed edges, *E*:

$$S = \{N_k \mid K = 1,2, \ldots m\}, \text{ and } E = \{ <N_i, N_j> \mid i \in 1, 2, \ldots, m, j \in 1, 2, \ldots, m\}$$

With a vertex and an edge in the graph corresponding to a state and a state transition in MM respectively, the weight on each edge of graph is then the

transition probability $a_{ij} = P(N_j \mid N_i)$ of an MM. If we consider MM as a complete graph, the transition probability distribution can be represented by an m × m matrix, so called *transition matrix*. In real life, there are no systems completely satisfying the Markov property, so this restriction is often loosely defined and assumed.

Once an MM is chosen to be the model for a system, it is constructed by defining the appropriate state representations and transition probabilities. The state representation of an MM is usually chosen by domain experts to well represent the property of the system modeled. It is expected that the number of states in an MM is enough to simulate the different states of the system but not so many that there is no significant difference between MM states. To develop a model to simulate the daily rainfall amount based on historical observation, Haan, Allen and Street (1976) used rainfall amount as state representation and grouped observed rainfall amount into six classes (states), which was found to be a reasonable choice of clustering to model the given data after experimenting on several class boundary settings. A statistical method was adopted by Yapo et al. (1993) to cluster observed streamflows when constructing a flood prediction model, in which the K-mean clustering algorithm was used to "minimize the total sum of the square distances from a streamflow value to cluster center" in order to find the optimal number of intervals and enough streamflow data in each interval. Once the states in the model are decided, the state transition probabilities are usually decided by the ratio of n_{ij}/n_i, where n_{ij} is the number of times that state transits from state i to state j, and n_i is the number of times the system is in i.

The model construction methods introduced above are static, which means all states and transition probabilities are defined in advance based on analyzing the existing data observations of the system. There also are several dynamic model construction algorithms that dynamically build the model, starting from one single state until an optimal model is found to mimic the reality.Cormack and Horspool (1987) introduced a dynamic Markov model for data compression, in which the model was dynamically constructed by using a "cloning algorithm," which expands a single state MM to a complex model by duplicating states according to the number of times the candidate state is visited by different predecessor states. Goldberg et al. (1999) presented the *Augmented Markov model (AMM)*, in which the observation of robot behavior data was read dynamically as input to modify the current model, which initiated as a single state Markov model. New states and transitions were added to the current model in real time when the input symbol had never been seen before in the model state space. Existing states in AMM also needed to be split (duplicated) when necessary to eliminate the non-exist path through that particular state.

We have developed another variation of Markov model, so called *Extensible Markov model (EMM)*, which also uses dynamic model construction algorithms

to build the model (Dunham, 2004). An EMM can be viewed as a time-varying MM where each state in the model represents a cluster of real-world states. At any snapshot, an EMM is an MM. However, over time, the graphical structure of the model (number of states in the graph, number of transitions, labeling of the states and labeling of the transition probabilities) changes. When a real-world state occurs, it is matched to the most similar state in the current model using an algorithm technique, EMMSim, which matches input data at time t + 1 to existing states in the MM at time t. The beauty of EMM is that EMMSim provides a generic method for matching input data with existing EMM states to make decisions of whether a new state needs to be added, and another algorithm, EMMBuild, updates the model structure and transition probability to incorporate the new input information in EMM.

A *Hidden Markov model (HMM)* models a system by decoupling the observed properties or *symbols* from the states that may produce them. There are two stochastic processes that occur in an HMM. One process determines what the next state should be using a transition probability distribution. This is identical to the stochastic process in a Markov model. The second process determines what symbol should be observed when the model is in a particular state. This process is controlled by an *observation probability distribution* (Rabiner, & Juang, 1986). So in addition to the set of states, *S*, and the transition probability distribution, *A*, an HMM needs a set of observable symbols, *V*, and an observation probability distribution, *B*:

$$V = \{ v_i \} \text{ and } B = \{ b_j(k) \mid j \in 1, 2, \ldots, m, k \in 1, 2, \ldots, |V| \}$$

Here $b_j(k)$ is equal to the probability of observing symbol v_k in state N_j.

Many of the same limitations presented for Markov models also apply to HMMs. They assume that the current state depends only on the previous state and the transition probability is independent of time. One additional consideration is that the observation probability distributions in each state are independent of each other.

Two of the fundamental HMM problems are how to *recognize* an observation sequence with a model and how to *uncover* the state sequence that optimally matches a given observation sequence. The first problem can be described as the probability of a specific observation sequence *O* given a HMM *l*. Rabiner et al. (1986) describes a dynamic programming algorithm known as the *forward-backward procedure* for solving this problem. To solve the second problem, we must establish what we mean by "optimal." Optimality is usually taken to mean the state sequence that would maximize the probability of the given observation sequence for the given model. Another dynamic programming approach – the *Viterbi algorithm* – is popular in the literature and uses this notion of optimality when solving this problem.

A third problem, which must be solved in both applications, is how to *train* the HMM, or determine the probability distributions *A* and *B* for the system being modeled. The popular approach to this is to guess the parameters initially and re-estimate them iteratively, improving the model each time, using various EM algorithms. A precursor to this is to determine how many states should exist in the model. Since the states represent some property of the system, this number can generally be determined based on the application.

Recent Data Mining Techniques Used for Flood Prediction

We have examined the use of four data mining techniques in predicting floods (or to be more precise, water level at a target location in a river). They are: HMM, EMM, combined time-series ARIMA model and Neural Networks (STIFF) and a more sophisticated approach to neural networks (NN-ACC).We briefly introduce these techniques and the results found during performance evaluations of them.

Hidden Markov Models

There have been several approaches to using HMMs in flood prediction. One approach uses multiple HMMs, each representing a discrete state of the prediction site. For example, one HMM could represent the occurrence of a flood, and another could represent an average river condition. In each model, the upstream measurements are treated as *observations*, and the time (relative to a starting time) as states. Then, during prediction, experts choose the model that best matches or *recognizes* the given observations using the *forward-backward procedure* (Rabiner et al., 1986). We call this a *Recognition-Based Model*.

The second approach draws an analogy between the components of an HMM – an observable sequence of symbols and a related but hidden state sequence – and the components of the flood prediction problem – an observable sequence of upstream river conditions and a hidden (unknown) sequence of future river conditions at the target site. Given an observation of the present condition of the upstream sites, the flood prediction problem is to *uncover* the best corresponding state sequence that represents the river conditions at the prediction site. Here it helps to think of the states as *related to* rather than *causing* the occurrence of the observation symbols. The accuracy of this model might reveal a few things about the nature of this relationship. We call this a *Viterbi-Based Model*.

Our initial experiments examining these two approaches using the water levels as measurements did not yield very positive results. In particular, the Viterbi-Based Model had many states, including some that were used too infrequently during training. Results improved when we considered the *change* in water level relative to the first measurement in a fixed time window because this reduced the variability of the measurements. But the HMMs still did not perform well compared to some existing NN techniques. Thus, we do not consider the use of HMMs further in this chapter.

Extensible Markov Models

We also examined the use of the *EMM* in flood prediction. In our experiments, the EMMSim algorithm was implemented using four different similarity measures (Dice, Jaccard, Cosine and Overlap). The threshold of similarity measurement determines whether a new node need to be added to the model; that is, if the similarity between an input reading and each of the states in current EMM is below the threshold, a new node is created representing a new state that is significantly different from current existing states. Our model was built and tested for flood prediction using data of river sensor readings (Dunham et al., 2004) obtained from the following Web site: www.ccg.leeds.ac.uk/simon/nndown.html, which provides real information of water levels at a catchment (Ouse Catchment) in the United Kingdom. The accuracy of the EMM prediction, which in this case is the water level at a designated location one hour ahead of time, was measured using *Root Means Square (RMS*) and *Normalized Absolute Ratio Error (NARE)* as described below:

$$\text{RMS} = \sqrt{\frac{\sum_{t=1}^{N}(O(t)-P(t))^2}{N}}$$

$$\text{NARE} = \frac{\sum_{t=1}^{N}|O(t)-P(t)|}{\sum_{t=1}^{N}O(t)}$$

Here O(t) is the observed value and P(t) is the predicted value at time t; N is the total number of test data and t is the time variable.

EMM prediction performance was compared with a Neural Network based prediction system, *River Level Forecasting (RLF)* that is an "Artificial Neural Networks for Flood Forecasting" available on the same Web site (Openshaw et al., 1998). Experiments showed that the number of states in the EMM grows at

a sublinear rate and levels off once the model has learned the current river behavior. If the river behavior changes, the model will begin learning again (Dunham, 2004). When comparing the performance of EMM to RLF on prediction accuracy, Table 1 shows the EMM performance is better.

There are several issues about EMM that deserve further investigation. First, even though the number of states tends to converge when more data are provided, it did not stop growing. To solve this problem, presumably a preset maximum number of states could be used to restrict the growth of number of EMM states, while the learning of links and transition probabilities are still carried on. Second, the water level data that was used to build and test EMM is pretty stable. More modifications of the EMM algorithm might be required when it is applied to data that varies strongly. Possible solutions would be i) choosing more sophisticated similarity algorithms; ii) including other environmental factors beyond water level to better represent the status of the system, iii) making multiple models to match different segments of the varied data, as proposed by Haan in 1976, where different MM models were built for each month. Third, algorithms of states merging and splitting could be included in EMM to closely model the dynamic problem domain.

Integrating Artificial Neural Network and Time Series

Recent work in flood prediction has investigated how combining different techniques could yield better results. Statistical analysis and artificial neural networks both have provided powerful tools to address spatio-temporal prediction problems. However, each comes with its drawbacks, which somehow cast a shadow on their applicability. We introduce our mixed model as follows.

To put it into a simple description, for spatio-temporal prediction, what our model does is build time series models to capture the temporal patterns, then construct an artificial neural network to retrieve the spatial correlation and finally "add" them together via regression. These characteristics led to the model we call *Spatio-temporal Integrated Forecasting Framework (STIFF)*. In this model, we first identify how many nearby locations could exert some influence on the target location. Then we divide the whole prediction area into sub-areas, of which each sub-area contains only one location we are interested in. Though this step looks straightforward – for example, for a flooding monitoring system, each location where one sensor sits can be one individual sub-area – sometimes it becomes more involved and needs other data mining techniques, such as clustering, classification, and so forth.

After the division, each location is examined independently. We build a time series ARIMA model for each location to capture its individual temporal

Table 1. Comparison of water level prediction of EMM and RLF

		NARE	*RMS*	*Number of States*
RLF		$83 * 10^{-3}$	0.422482	
EMM	*Threshold 0.995*	$34 * 10^{-3}$	0.1733741	92
	Threshold 0.997	$26 * 10^{-3}$	0.1310777	127
	Threshold 0.999	$156 * 10^{-3}$	0.0774781	238

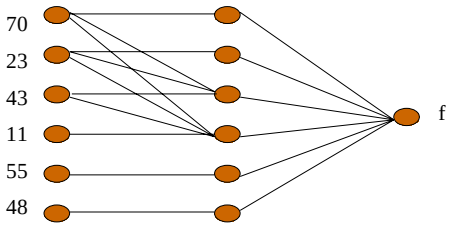
characteristics. With time series models completed, we have the necessary prediction capability at each location. To capture the spatial aspect, we then construct an NN to capture the spatial correlation from the target location's siblings to the target location itself. To make construction simple, normally we just pick up the traditional three-layer (input, hidden, output) structure. However, partial connectivity, instead of full connectivity, between different layers is employed to reflect the spatial structure of the catchment from which NN is constructed. For example, Figure 2 shows the NN structure for the catchment shown in Figure 1. The labels of the input nodes indicate the upstream sensor locations. Each node in the input layer is connected to corresponding nodes in the hidden layer which represent sensor nodes on which it has a direct spatial influence. For example, sensor location 28070 (see Figure 1) has a direct influence on itself as well as 28043 and 28011. The output node represents the prediction, f, at the target location 28010.

To obtain the flood prediction, prediction (using the time series models) or actual readings from the upstream sensors are used as input to the NN model. The actual input value is based on the projected time lag from that upstream sensor to the target location. The final prediction value for the target location is then obtained via statistical regression on both the NN prediction and the time series prediction at the target node. Experiments on data collected from a real catchment have shown that the STIFF model works very well, with higher prediction accuracy and balanced behaviors. With a 30-day look-ahead, STIFF had a NARE value of 0.055 while an NN was 0.167 and time series alone was 0.15. For detailed information, please refer to Li, & Dunham, (2002, 2003a; Li, Dunham, & Xiao, 2003b).

Improved Neural Network Approach

We have examined an NN approach, the *Neural Network with Auto- and Cross-Correlation Model (NN-ACC)*, which overcomes some of the problems

Figure 2. STIFF ANN for catchment in Figure 1



associated with traditional NN modeling in this domain by using cross-correlation/autocorrelation techniques to determine NN input. A cross-correlation function between the upstream stations' water level readings and the corresponding data from the target station, and an auto-correlation function between historical readings and current values at the target, are calculated. In the present approach, the cross-correlation function between the upstream stations' water level readings and the corresponding data from the target station are calculated.In addition to this, an auto-correlation function between historical readings and current value at the target are considered. Based upon a chosen threshold, the precise input values for the NN are determined. The NN-ACC algorithm contains the following steps:

1. Preprocess Historical Data
2. Determine the Optimal Input Domain based upon the Auto/Cross-Correlation Technique
3. Create Optimal Neural Network structure
4. Determine "Good" Initial Values (Weights and Biases) for Neural Network
5. Use Training (Levenberg-Marquardt Algorithm) and Validation Method

A crucial part of step 3 is to determine the number of input nodes and their representations for the feedforward NN with two hidden layers. This information is defined empirically in many publications. In Openshaw et al. (1998), the lag for an upstream station is determined based on speed of water. However, this value could change dramatically during the flood event.

Our research introduces a more deterministic approach to establishing lag. The lag is defined by calculating correlation functions between historical readings of this upstream station and historical readings from the target station. Based upon

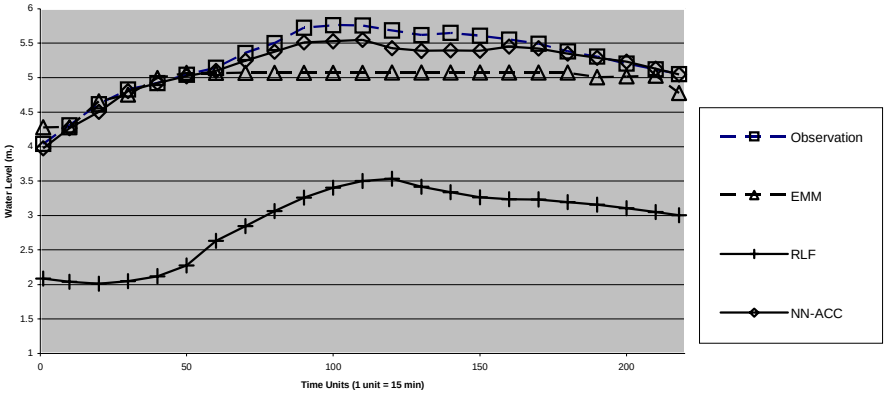
the arguments of this function, for which cross-correlation is greater than or equal to some threshold value, components of a lag window are chosen. A series of experiments for three-hour-ahead prediction were performed to determine the optimal threshold for the best network architecture. As the result, the threshold equal to 0.980 is chosen, corresponding to the five input nodes (|P| = 5). This value consists of four historical water level readings taken from the target station, and one Boolean value of the day/night at time prediction. The data from upstream stations do not have significant influence on the prediction of the neural network value. Cross-correlation function for any upstream station has a very sharp decline around the maximum value, which is significantly less than the corresponding parameter for auto-correlation function. The number of nodes in hidden layers |T| was defined using the methodology described in Baum (1989). If the desired accuracy level is 90%, the number of weights should be 10 times less than the size of input vector set {P1,…,Pn}. This rule is mentioned in various publications as "the rule of thumb" (Bishop, 1994). According to this methodology, an optimal number of hidden nodes is equal to ten (|T| = 10). A series of experiments, performed upon the neural network with five input nodes and two hidden layers for three-hours-ahead prediction, gave approximately the same value of |T|. The output layer has one node (|Z| = 1): the value of the predicted water level. The performance of the NN model was tested by predicting the sensor reading of a water level value at a downstream location of the Ouse Catchment from January 26, 1995 to February 2,1995 (USGS, 2004). To evaluate NN-ACC against other RLF and EMM flood prediction techniques, the one-hour forecast was performed for previously used neural network architecture (5-10-10-1, where |P| = 5, |T| = 10, |Z| = 1, and |P|-|T|-|T|-|Z|). The result of the comparison is shown in Table 2 and Figure 3.

NN-ACC gives very good prediction and outperforms the EMM and RLF models. NN-ACC captures an optimal number of input/hidden nodes and, because of these characteristics, NN-ACC outperforms EMM and RLF frameworks. The EMM model fails to predict the increasing observation values around time unit 100. The RLF NN model yields performance significantly worse than either EMM or NN-ACC.

Conclusions

The spatio-temporal prediction problem is extremely difficult. Conventional solutions using counting models are too labor intensive. By simplifying some of the issues, such as spatial influence, specialized subproblems may be examined. We have evaluated four data mining techniques to attack the flood prediction

Figure 3. Comparison of NN-ACC, EMM and RLF Models, one-hour prediction



problem. Using two new error measurement metrics, we have shown that HMMs do not appear to be better than NNs. However, more sophisticated data mining modeling techniques (EMM, STIFF, NN-ACC) can yield better results than previously proposed methods. While the flood prediction problem is a subproblem of the more general spatio-temporal prediction problem, there are many real-world applications of this type. Although we cannot generalize our results to any type of spatio-temporal prediction problem, we feel that they are promising enough that more research is warranted.

Currently in the real world, it appears that techniques to address spatio-temporal prediction seem to center around the more simplistic, more understood counting techniques. There does not appear to be any acceptance in the real world to more sophisticated, less understood data mining technqiues. However, we feel that in the future this really should change. Due to the very nature of the problem, and its applicability to many real-world applications, future study to examine better data mining solutions is needed. The potential benefits are quite high.

Table 2. Comparison NN-ACC with EMM and RLF flood prediction models

Model	NARE	RMS
EMM	0.065	0.413
RLF	0.447	2.374
NN-ACC	0.0239	0.145

We propose that future research is needed in the following areas:

- Creation of more sophisticated data mining techniques to model the complex spatio-temporal problems, or at least subproblems thereof.
- Evaluation of other data mining forecasting techniques to the spatio-temporal problem.
- Evaluation of combining data mining tools to attack the spatio-temporal prediction problem.

References

Arctic and Antarctic Research Institute (2004). Russian Federal Service for Hydrometeorology and Environmental Monitoring. Retrieved 2004 from *www.aari.nw.ru*

Baum, E. (1989). What size net gives valid generalization? *Neural Computation, 1*, 151-160.

Bishop, C. (1994). *Neural networks for pattern recognition.* Oxford: Oxford University Press.

Box, E.P.G., & Jenkins, M.G. (1970). *Time series analysis: Forecasting and control*. San Francisco: Holden-Day.

Box, G., Jenkins, G., & Reinsel, G. (1994). *Time series analysis*. Englewood Cliffs, NJ: Prentice Hall.

Burnash, R., & Ferral, L. (1996, July), *Conceptualization of the Sacramento Soil Moisture Accounting Model.* NWSRFS Users Manual, Part II.3, National Weather Service, NOAA, DOC, Silver Spring, MD.

Comet, *National Weather Service River Forecast System Sacramento Soil Moisture Accounting Model*, Cooperative Program for Operational Meteorology, Education and Training, Hydromet 00-2, March 7, 2000, Riverside Technology Incorporated, *http://www.comet.ucar.edu/class/hydromet/10_Feb23_2000/docs/ri/sac-sma.ppt*

Cormack, G.V., & Horspool, R.N.S. (1987). Data compression using dynamic Markov modeling. *The Computer Journal, 30*(6), 541-549.

Cressi, N., & Majure, J.J. (1997). Spatio-temporal statistical modeling of livestock waste in streams. *Journal of Agricultural, Biological and Environmental Statistics, 2*(1), 24-47.

Deutsch, S.J., & Ramos, J.A. (1986). Space-time modeling of vector hydrologic sequences. *Water Resource Bulletin (22), 967-980.*

Dougherty, L., Corne, S., & Openshaw, S. (1997). Some initial experiments with

neural network models of flood forecasting on the river Ouse. *Proceedings of the Second Annual Conference of GeoComputation*, (97) (pp. 59-67). Incorporating the Spatial Information Research Centre's 9th Annual Colloquium, University of Otago: New Zealand.

Dunham, M. (2003). *Data mining: Introductory and advanced topics*. NJ: Prentice Hall.

Dunham, M.H., Huang, J., & Meng, Y. (2004). Extensible Markov model. *Proceedings of the IEEE International Conference on Data Mining*, November.

Goldberg, D., & Mataric, J.M. (1999). Augmented Markov models. *USC Institute for Robotics and Intelligent Systems Technical Report*, IRIS-99-367.

Haan, C.T., Allen, D.M., & Street, J.O. (1976). A Markov chain model of daily rainfall, *Water Resource Research*, *12*(3), 443-449.

Handcock, M.S., & Wallis, J.R. (1994). An approach to statistical spatial-temporal modeling of meteorological fields (with discussion). *Journal of the American Statistical Association, 89*, 368-390.

Isaacson, L.D., & Madsen, W.R. (1976). *Markov chains theory and applications*. New York: John Wiley & Sons.

Jothityangkoon, C., Sivapalan, M., & Viney, N.R. (2000). Tests of a space-time model of daily rainfall in Southwestern Australia based on nonhomogeneous random cascades. *Water Resource Research, 36*(1), 267-284.

Julien, P.Y., Molnar, D.M., Johnson, B.E., & Combs, P.G. (1998). Flood forecasting reaches new potential. Retrieved from the American Geophysical Union, *www.agu.org/eos_elec*

Kelly, R.P., Clapp, R.B.J., & Rodriguez, M. (1998). Spatiotemporal autoregressive models of neighborhood effects. *Journal of Real Estate Economics*, *17*(1), 15-33.

Li, Z., Dunham, M.H. & Xiao, Y. (2002,). STIFF: A forecasting framework for spatio-temporal data. *Proceedings of the KDMCD*, PAKDD'02. Springer Verlag. *Lecture Notes in Artificial Intelligence Volume 2797*.

Li, Z., Dunham, M.H., & Xiao, Y. (2003b). STIFF: A forecasting framework for spatio-temporal data. In O.R. Zaïane, S.J. Simoff, & C. Djeraba (Eds.), *Mining multimedia and complex data*. Springer Verlag. *Lecture Notes in Artificial Intelligence Volume 2797*, pp. 183-198.

Li, Z., Liu, L., & Dunham, M.H. (2003a). Considering correlation between variables to improve spatiotemporal forecasting. *Proceedings of the PAKDD'03*.

NOAA (2004). Sacramento soil moisture accounting model. Retrieved from *www.nwstc.noaa.gov/HYDRO/RFS/RFS503b.html*

Oly, J., & Reed, A.J. (2002). Markov model prediction of I/O requests for scientific applications.*Proceedings of the ICS 2002,* (pp. 147-155).

Openshaw, S., Kneale, P., Corne, S., & See, L. (1998). *Artificial neural networks*, MAFF Project OCS967P, Final Report.

Pfeifer, E.P., & Deutsch, S. (1980a). Identification and interpretation of first order space time ARMA models. *Technometrics, 22*(3), 397-408.

Pfeifer, E.P., & Deutsch, S. (1980b). A three-stage iterative procedure for space-time modeling. *Technometrics, 22*(3), 397-408.

Pokrajac, D., & Obradovic, Z. (2001). Improved spatial-temporal forecasting through modeling of spatial residuals in recent history. *Proceedings of the First SIAM International Conference on Data Mining (SDM'01).*

Rabiner, L.R., & Juang, B.H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine, 3*(1), 4-16.

Roddick, J.F., Hornsby, K., & Spiliopoulou, M. (2000). An updated bibliography of temporal, spatial, and spatio-temporal data mining research. *Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining* (pp. 47-164). London: Springer-Verglag.

Singh, G.D., Chaplain, M.A.J., & McLachlan, J. (1999). *On growth and form: Spatio-temporal pattern formation in biology.* NJ: John Siley & Sons.

Stroud, J., Muller, P., & Sanso, B. (2001). Dynamic models for spatio-temporal data. *Journal of the Royal Statistical Society Series B, 63,* 673-689.

United States Geological Society. (2004). *Steam gaging and flood forecasting*. Retrieved August 2004 from *http://water.usgs.gov/widFS_209-95/mason-weiger.html*

Waller, L.A., Carlin, P.B., Xia, H., & Gelfand, E.A. (1997). Hierarchical spatio-temporal mapping of disease rates. *Journal of the American Statistical Association, 92,* 607-617.

Wei W. (1990). *Time series analysis: Univariate and multivariate methods.* Addison-Wesley.

Yapo, P., Sorooshian, S., & Gupta, V. (1993). A Markov chain flow model for flood forecasting. *Water Resource Research, 29*(7), 2427-2436.

Endnotes

[1] This material is based upon work supported by the National Science Foundation under Grant No. IIS-9820841.

Chapter XII

Mining in Spatio-Temporal Databases

Junmei Wang, National University of Singapore, Singapore

Wynne Hsu, National University of Singapore, Singapore

Mong Li Lee, National University of Singapore, Singapore

Abstract

Recent interest in spatio-temporal applications has been fueled by the need to discover and predict complex patterns that occur when we observe the behavior of objects in the three-dimensional space of time and spatial coordinates. Although the complex and intrinsic relationships among the spatio-temporal data limit the usefulness of conventional data mining techniques to discover the patterns in the spatio-temporal databases, they also lead to opportunities for mining new classes of patterns in spatio-temporal databases. This chapter provides a survey of the work done for mining patterns in spatial databases and temporal databases, and the preliminary work for mining patterns in spatio-temporal databases. We highlight the unique challenges of mining interesting patterns in spatio-temporal databases. We also describe two special types of spatio-temporal patterns: location-sensitive sequence patterns and geographical features for location-based service patterns.

Introduction

The globalization of many industries and advances in wireless communication and global positioning systems have led to the development of spatio-temporal applications, such as applications dealing with moving objects, involving objects located in the space (for example, land parcels, whose characteristics may change in time) and dealing with objects which integrate the above two behaviors (for example, pollution phenomenon in the environment system). A spatio-temporal database embodies spatial, temporal and spatio-temporal concepts, and captures simultaneously the spatial and temporal aspects of data. The complex interactions among the objects are captured in the form of the past, present and future states in the modeled environment. With the widespread use of spatio-temporal databases, there is the increasing need for the discovery of interesting and previously unknown, but potentially useful, patterns in spatio-temporal databases.

Currently, spatial data mining and temporal data mining form two separate streams of research. Efforts are either focused on discovering space-sensitive patterns in the form of *spatial patterns* (Chawla, Shekhar, &Wu, 2000; Ester, Frommelt, Kriegel, & Sander, 2000; Han, Koperski, & Stefanovic, 1997; Shekhar & Huang, 2001; Wang, Yang, & Muntz, 1999); or time varying patterns in the form of *sequence patterns* (Agrawal & Srikant, 1996; Pei, Han, & Asl, 2001; Zaki, 1998; Yang, Wang, Yu, & Han, 2002; Mannila, Toivonen, & Verkamo, 1995; Garofalakis, Rastogi, & Shim, 1999). However, spatio-temporal databases contain the complex relationships that cannot be discovered by simply considering the temporal information or spatial information independently.

For example, it is not sufficient to know that "*sales are typically up in the months of February and December*," or that "*sales are high in Asia and North America*." Instead, it is more important to understand the trends in the context of spatial locations, such as "*sales in the Asia region peak in the month of February while sales in the North America continent peak in the month of December*." Further, the pattern "*motorists go to gas stations upon exit from a freeway with 80% likelihood*" can only be discovered when the mining algorithm takes into account the multi-states (that is, the past, present and future states) of the objects.

Data mining in spatio-temporal databases must consider the multi-states of the spatio-temporal data. It must integrate the spatial information and temporal information together to find meaningful spatio-temporal patterns. The knowledge of these spatio-temporal patterns allows one to develop more localized or customized business analysis and strategies, and have potential benefits for many applications, such as the Geographic Information System (GIS), environ-

ment information system, traffic supervision system, mobile phone companies, and so forth.

This chapter is organized as follows: We first review the work for mining in spatial databases and temporal databases and the initial work for mining in spatio-temporal databases. Then, we describe a framework for mining spatio-temporal databases, and illustrate the various types of interesting spatio-temporal patterns that can be discovered. We describe two special types of spatio-temporal patterns in detail; namely, location-sensitive time-sequence patterns, and the geographical features for location-based service patterns. Finally, we conclude with possible research directions for mining interesting patterns in spatio-temporal databases.

Related Work

Mining in temporal databases and in spatial databases presently constitute two separate streams of research. Research effort has been focused on discovering patterns either from spatial or temporal databases. This section first reviews the work done on spatial data mining and temporal data mining before describing early attempts at spatio-temporal data mining.

Mining in Spatial Mining Databases

Spatial data mining is the process of discovering interesting and useful patterns from large spatial databases. Mining patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data, due to the complexity of spatial data.

Spatial data exhibits a unique property in that "*everything is related to everything else, but nearby things are more related than distant things*," which is also known as Tobler's first law of geography (Tobler, 1979). Hence, the goal of spatial data mining is to discover relationships between spatial data and non-spatial data by using spatial proximity relationships, such as topological relationships like *intersects*, *overlap*, *disjoint*; spatial orientation, such as *left_of*, *east_of*; and distance information, such as *close_to*, *far_away*.

Spatial data mining covers a wide spectrum of paradigms for knowledge discovery. The patterns discovered from spatial data includes characteristic and discriminant rules, spatial association rules, extraction and description of prominent structures or clusters, and so forth. (Han & Kamber, 2001). We will briefly review the techniques to extract these patterns from the spatial databases.

Mining of Characteristic and Discriminant Patterns

A *spatial characteristic rule* is a general description of a set of spatial related data. For example, the description of general weather patterns in geographic regions is a spatial characteristic rule. A *spatial discriminant rule* is the general description of contrasting or discriminating features of a class of spatial related data from other classes. For example, the comparison of the weather patterns in two geographic regions is a spatial discriminant rule.

Lu, Han and Ooi (1993) first developed a generalization-based method to discover the characteristic and discriminant rules from the spatial data. The method extracts the general knowledge in two different ways: *non-spatial data dominated generalization* and *spatial data dominant generalization*.

The non-spatial data dominated generalization algorithm creates maps consisting of regions that share the same high-level non-spatial descriptions. It realizes this by merging the neighboring areas with the same generalized non-spatial attributes. In contrast, a spatial data dominant generalization algorithm focuses first on the spatial data. It partitions the regions and merges them based on the hierarchy of spatial data attributes. In the end, it creates maps consisting of areas that share the same spatial descriptions.

Although the generalization-based approach could find some interesting patterns from the spatial databases, the discovery process depends very much on the availability of the hierarchies of the data. Further, the quality and the interestingness of the discovered patterns are also influenced greatly by the fineness and appropriateness of the given hierarchies of data.

Mining of Spatial Association Patterns

The spatial characteristic and discriminant rules describe the spatial and nonspatial relationships at a general concept level, where spatial objects are expressed as merged spatial regions or clustered spatial points. However, they do not reflect the relationships of spatial/spatial data or spatial/nonspatial data, while a spatial association rule does.

A *spatial association rule* describes the implication of one set of features by another set of features in spatial databases. A rule such as "*large towns in British Columbia are close to the sea*" is a spatial association rule. A spatial association rule has the form $X \rightarrow Y$, where X and Y are sets of predicates and some of which are spatial predicates. Commonly used spatial predicates include topological relationships (for example, *intersects*, *overlap)*; spatial orientation (*left_of*, *west_of*); and distance information (*close_to*, *far_away)*.

Koperski and Han (1995) first extended the concept of the association rule to the spatial databases based on the similar philosophy for mining association rules in transaction databases (Agrawal & Srikant, 1994). The following spatial association rule finds the association relationships between schools and parks that are 80% of schools are close to parks: *is_a*(*x*, school) →*close_to*(*x*, park) (80%).

One of the drawbacks of the spatial association rule algorithm is that it depends on the concept of explicit transactions in the databases. However, due to the continuity of the underlying space, this may not be possible or appropriate in some spatial databases. For example, if spatial association rules discovery is confined to some reference feature, for example, city, then transactions can be defined around the instances of this reference feature. However, for those cases where no reference feature is specified, the generalization of this concept will be non-trivial. Moreover, many duplicate counts of association rules may result if we define transactions around locations of instances of all features. For these reasons, Shekhar and Huang (2001) propose a new framework to find co-location patterns by using user-specified neighborhoods to specify groups of items, instead of transaction.

The co-location pattern discovery process finds the subsets of spatial features that frequently locate together. The co-location patterns represent relationships among events happening in different and possibly nearby locations. For example, the pattern "*smoke aerosols alter the likelihood of rainfall in a nearby Region*" is a co-location pattern. A co-location pattern has the form: $f_1 \rightarrow f_2$ (*cp*%), where f_1 and f_2 are spatial features, *cp*% is the conditional probability indicating that at least *cp*% objects having the spatial feature of f_1 are the neighbors to some objects having the spatial feature of f_2.

Other Techniques to Discover Patterns in Spatial Patterns

In addition, various systems and methods have been developed to allow users to conveniently extract patterns in spatial databases. The GeoMiner system developed by Han et al. (1997) incorporates various existing techniques to mine interesting spatial patterns, with provision for the addition of newly developed methods at a later stage. Following that, Ester et al. (1998, 2000) developed a set of database primitives for mining in spatial databases. They use graphs to model the implicit neighborhoods' relationships and add appropriate operations to manipulate these graphs.

Wang et al. (1999) introduced an active spatial data mining approach to support user-defined triggers on dynamically evolving spatial data. They employ a hierarchical structure and decompose the user-defined trigger into a set of sub-

triggers associated with cells in the hierarchy. Updates are suspended in the hierarchy until their cumulative effect causes the trigger to fire.

Data Mining in Temporal Databases

Temporal database mining considers two types of temporal data. They are *time-series data* and *sequence data*. A time-series data is a sequence of real numbers that vary with time; for example, stock prices, exchange rates, biomedical measurements data, and so forth. A sequence data is a sequence of ordered events, with or without concrete notions of time; for example, Web page traversal sequences.

Temporal data mining aims to discover and infer relationships of contextual and temporal proximity among the data. Patterns that can be discovered from temporal data include trends, temporal association patterns and sequential patterns. This section examines techniques for mining sequence patterns and temporal association patterns in time-related data.

Mining of Sequence Patterns

Sequence data is an important type of temporal data. It is a list of *transactions*, where each transaction is a set of literals, called *items*. A transaction time is associated with each transaction. Discovering sequence patterns means finding correlations among the events in sequence data. A *sequence pattern*, or *sequence* for brevity, is an ordered list of elements, and an *element* is a set of items appearing together in a transaction. Elements need not be adjacent in time, but their ordering in a sequence may not violate the time ordering of the supporting transactions. An example of such a pattern is that a *customer typically rents "Star Wars," then "Empire Strikes Back" and then "Return of the Jedi."*

Note that the problem of discovering sequence patterns is different than the problem of association rule discovery. Association rules discover the intra-transaction relationships between various items, while the sequence patterns discover the inter-transaction relationships between data sequences.

Many approaches have been proposed for finding frequent sequential patterns. Agrawal and Srikant (1996) first introduced a breadth-first disk-based algorithm. Subsequently, Han et al. (2000), Pei et al. (2001) and Zaki (1998) investigated depth-first projection-based methods to mine sequence patterns. The depth-first approaches generally perform better than the breadth-first approaches if the data resides in memory. Recently, Yang et al. (2002) presented an algorithm to mine frequent patterns in the presence of noise. Methods to mine sequences by

incorporating constraints to reduce the search space have also been developed (Garofalakis et al., 1999; Mannila et al., 1995; Pei, Han, & Wang, 2002).

Mining of Temporal Association Patterns

Another class of patterns in the temporal databases is the temporal association rules. The discovery of temporal association rules analyzes the data in finer time granularity. It may reveal that the association rule exists only in certain time intervals, and does not occur in the remaining time intervals. For example, the pattern *"coffee and doughnuts have a strong tendency to occur together with high support during the time interval 7AM - 9AM"* is a temporal association rule.

Many approaches have been proposed for mining the temporal association patterns, such as *cyclic association rules* (Özden, Ramaswamy, & Silberschatz, 1998), *periodic association rules* (Han, Dong, & Yin, 1999) and *calendric association rules* (Ramaswamy, Mahahan, & Siberschatz, 1998). Chen and Petrounias (2000) present a framework for mining temporal association patterns. They define a temporal association rule as a pair <*AR*, *TF*>, where *AR* is an implication of the association rule and *TF* is a temporal feature that *AR* possesses. Temporal features can be represented by a valid period, periodicity or a specific calendar. Depending on the interpretation of the temporal feature *TF*, a temporal association rule <*AR*, *TF*> can be expressed in various forms, such as a *cyclic* association rule, an *interval* association rule, a *periodic* association rule or a *calendric* association rule.

Preliminary Work on Data Mining in Spatio-Temporal Databases

Next, we examine the early attempts at pattern extraction in spatio-temporal databases. A system called CONQUEST (Stolorz, Nakamura, Muntz, & Mechoso, 1995; Muntz, Shek, & Mechoso, 1995) was first developed to allow some means of accessing and interpreting spatio-temporal data. It provides an environment that enables geophysical scientists to formulate queries on spatio-temporal patterns on massive data, such as cyclones, hurricanes and fronts.

Following that, researchers (Kumar et al., 2001; Tan, Steinbach, & Kumar, 2001; Steinbach et al., 2001; Steinbach, Tan, Kumar, Klooster, & Potter, 2002) attempted to mine interesting spatio-temporal patterns in earth science data. They applied existing data mining techniques to find clusters, predictive models and trends, and stated that existing data mining algorithms cannot discover all the interesting patterns in spatio-temporal data (Tan et al., 2001).

Abraham and Roddick (1997) introduced the concept of spatio-temporal meta-rules to describe the changes in rule sets obtained from consecutive spatial snapshots. For example, a spatial association rule $A \rightarrow B$ has the confidence and support (c_1%, s_1%) at time t_1, and (c_1%, s_2%) at time t_2. Hence, the meta-rule can be expressed as *"the rule A'!B that describes the data at t_1 changes support between t_1 and t_2."*

Tsoukatos and Gunopulos (2001) presented an algorithm to discover frequent sequences in a depth-first manner over all the locations in spatio-temporal databases. This is essentially a sequence mining algorithm whereby each location is treated as a transaction. The algorithm is able to find the common temporal relationships of events in some locations, but not the relationships of events among these locations.

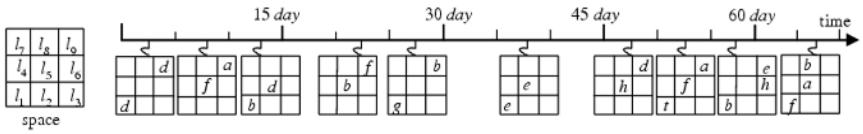
Moving object databases have also received considerable attention recently. Vlachos, Kollios and Gunopulos (2002) proposed a method for finding similar object trajectories in two- or three-dimensional space by formalizing non-metric similarity functions. Zhang, Gunopulos, Tsotras and Seeger (2003) examined the problem of computing temporal and spatio-temporal aggregations over data streams. The aggregates are maintained using multiple levels of temporal granularity. Specialized indexing schemes are also designed for maintaining the aggregates. Tao, Kollios, Considine, Li and Papadias (2004) introduced a method for solving the *distinct counting problem*. This problem arises when an object remains in the query region for several time stamps, which will result in the object being counted multiple times. They propose to solve this problem by integrating spatio-temporal indexes with sketches. They also use the same method to mine the spatio-temporal association rules.

Patterns in Spatio-Temporal Databases

A spatio-temporal database embodies spatial, temporal and spatio-temporal concepts, and captures simultaneously the spatial and temporal aspects of data. The availability of both temporal information and spatial information introduces opportunities and challenges of mining for spatio-temporal patterns that complement existing time-varying patterns or space-sensitive patterns. This section first gives some definitions, followed by a discussion on the various types of interesting patterns that can be extracted in a spatio-temporal database.

Suppose the space S could be divided into a set of locations $L = \{l_1, l_2, ..., l_q\}$. Each location has a unique identifier. Let $F = \{f_1, f_2, ..., f_u\}$ be a set of u *spatial features*, such as drought, rain. Let R be a neighbour relation over the locations in L.

Figure 1. Framework of spatio-temporal databases



A *location-based event* or *loc-event* is a set of spatial features at location l_k occurring at time *t*, denoted as $e = (f_h(l_k), t)$ or simply $e = f_h(l_k)$ when the sequential context is clear.

Let $E = \{e_1, e_2, ..., e_m\}$ be a set of loc-event instances that occur at the same time *t*, where e_i $(1 \leq i \leq m)$ is in the form of $f_i(l)$. For convenience, we write *E* as $<e_1, e_2,..., e_m>$.

Let *W* be the sliding window width, which indicates the temporal proximity of the loc-events in the time dimension. A *sequence* is a list of *E* in the same sliding window sorted by time, which has the same format as in the temporal database (Agrawal & Srikant, 1996).

We observe four types of patterns in the spatio-temporal database shown in Figure 1:

1. **Co-location patterns.** Co-location patterns in a spatio-temporal database aim to determine the set of events that frequently occur together; that is, the locations of these events satisfy the neighbour relation *R*. For example, when *W* = 15 *days*, $<b(l_1), f(l_5), a(l_9), d(l_9)>$ is a co-location pattern.
2. **Global sequence patterns.** A global sequence, $s = \{(s_1 \rightarrow ... \rightarrow s_t)::(X)\}$, where s_i $(1 \leq i \leq t)$ is a set of spatial features (that is, $s_i = \{f_1,..., f_p\}$) and *X* is a set of locations and $X \subseteq L$ is a frequently occurring pattern if there are at least τ different locations containing *s*. For example, $\{(d \rightarrow b \rightarrow e)::\{l_1,l_5,l_9\}\}$ $(\tau = 2)$ is a frequent global sequence pattern.
3. **Local sequence patterns.** A local sequence is a sequence $s = \{(s_1 \rightarrow ... \rightarrow s_t)::(l_x)\}$, where s_i $(1 \leq i \leq t)$ is a set of spatial features (that is, $s_i = \{f_1, ..., f_p\}$) and $l_x \in L$. Given a sliding window *W*, we say that *s* is frequent if there are at least τ different windows at location l_x containing *s*.
4. **Location-sensitive sequence patterns.** A location-sensitive sequence pattern is a list of events sorted by time, denoted as $a = (E_1 \rightarrow ... \rightarrow E_t)$, where E_i $(1 \leq i \leq t)$ is an eventset with the format $<e_1, e_2, ..., e_p>$ and e_k $(1 \leq k \leq p)$ is a loc-event. Given a sliding window *W*, *a* is said to be frequent if there are at least *sup* different windows containing *a*. For example, the pattern $d(l_9) \rightarrow <a(l_9), f(l_5)> \rightarrow b(l_1)$ is a location-sensitive sequence pattern.

Existing association rule or sequence mining techniques can be customized to find co-location patterns and local and global sequence patterns. However, for location-sensitive sequence patterns, the efficiency of existing methods is limited by the large search space and large number of patterns. Hence, more efficient algorithms are needed for mining location-sensitive sequence patterns.

Mining Location-Sensitive Sequence Patterns

One of the challenges in discovering location-sensitive sequence patterns is the large search space. For example, 100 spatial features (f_1, ..., f_{100}) and 10 locations (l_1, ..., l_{10}) will yield 1000 different loc-items ($f_j|l_k$) ($1 \le j \le 100, 1 \le k \le 10$). These 1000 loc-items will potentially generate up to

$$\binom{1000}{3} = 166{,}167{,}000$$

location-sensitive sequences of length 3. Many interesting patterns are often buried in the sea of frequent patterns. In practice, it is impractical to generate the entire set of frequent spatio-temporal patterns when there are very long patterns that present in the data. Hence, some smaller alternatives of the set of interesting patterns that still contain enough information; for example, the set of frequent closed patterns, the set of maximal frequent patterns, become increasingly attractive and important. Let us consider the problem of mining the set of maximal frequent location-sensitive sequence patterns.

A location-sensitive sequence pattern with k location-based events ($k = \sum_j e_j$) is called a *k-LocSeq*. For example, $a(l_1) \rightarrow <b(l_2), a(l_3)>$ is a 3-*LocSeq*. A pattern $P = (E_{P1} \rightarrow E_{P2} \rightarrow ... \rightarrow E_{Pt})$ is a *sub-LocSeq* of another pattern $Q = (E_{Q1} \rightarrow E_{Q2} \rightarrow ... \rightarrow E_{Qs})$, denoted as $P \subseteq Q$, if there exist integers $1 \le i_1 < i_2 < ... < i_t \le s$ such that $E_{Pj} \subseteq E_{Qij}$ for all E_{Pj}. If P is a sub-LocSeq of Q, Q is also called the *super-LocSeq* of P. A location-sensitive sequence pattern is *maximal* if it is not a sub-LocSeq of any pattern.

Given a database D of location sequences, neighbour relation R, minimum support *minsup* and sliding window W, the problem of mining location-sensitive sequence patterns in spatio-temporal databases is equivalent to finding the set of all maximal frequent location-sensitive time sequence patterns.

Methods to mine frequent sequence patterns are not suitable for mining the maximal frequent sequence patterns. The Apriori-based sequence mining algorithm (Agrawal & Srikant, 1996) is not efficient for mining maximal frequent

patterns for two reasons. First, it generates and counts *all* the 2^k subsets of *each* *k*-sequence, and hence, it does not scale for long patterns. Second, it uses the breadth-first approach which finds all the frequent k-sequences before considering (*k+1*) sequences. This approach limits the effectiveness of look-aheads since useful longer frequent patterns have not yet been discovered.

Tsoukatos and Gunopulos (2001) and Zaki (1998) utilized a depth-first approach to improve the performance in discovering maximal frequent patterns. While this approach can find the maximal frequent patterns and prune away the infrequent sequences, they require the exhaustive enumeration of *all* the possible k-sequences to find the frequent (*k*+1) sequences. When the patterns are long, the effectiveness of these approaches becomes limited.

On further investigation, we observe that a sequence pattern of length 2 specifies a temporal relationship between frequent items that must be maintained in the higher-order sequence patterns. These temporal relationships can be utilized to drastically reduce the number of candidates generated. Hence, we design a new candidate generation method that uses the depth-first approach, and develop an efficient algorithm, called FlowMiner, for mining maximal frequent location-sensitive sequence patterns (Wang, Hsu, Lee, & Wang, 2004).

The FlowMiner algorithm first scans the database to retrieve all the neighborhoods *NL* according to some neighbour relation *R*, such as topological relations, metric relations or direction relations. We are interested in the locations where instances of spatial features occur. After obtaining all the neighborhoods *NL*, the location-time sequence patterns for each neighborhood $L \in NL$ can be discovered in three main steps: First, we determine all the frequent 1-*LocSeqs*. These frequent 1-*LocSeqs* are sorted in decreasing order of their frequencies. Next, we form the candidate 2-*LocSeqs* by joining two frequent 1-*LocSeqs*. Finally, we utilize the temporal constraints specified by the 2-*LocSeq* to generate the subsequent *k*-*LocSeqs* ($k > 2$). This is possible because each 2-*LocSeq* specifies a temporal relationship that must be maintained by the higher-order *LocSeq*s.

Use $set(d(l_9) + a(l_9))$ to indicate the set of frequent 2-*LocSeq*s generated by the loc-events $d(l_9)$ and $a(l_9)$. The original $set(d(l_9) + a(l_9))$ consists of three 2-*LocSeq*s $\{a(l_9) \rightarrow d(l_9), \langle a(l_9), d(l_9)\rangle, d(l_9) \rightarrow a(l_9)\}$, indicating that the loc-event $a(l_9)$ may occur *before*, *at the same time* or *after* the loc-event $d(l_9)$ respectively. Suppose we have three sets of frequent 2-*LocSeq*s:

$$set(d(l_9) + a(l_9)) = \{d(l_9) \rightarrow (l_9)\}$$
$$set(d(l_9) + b(l_1)) = \{d(l_9) \rightarrow (l_1)\}$$
$$set(d(l_9) + f(l_5)) = \{d(l_9) \rightarrow (l_5)\}$$

If we extend a 3-*LocSeq* $<a(l_9), f(l_5)>$'‡$b(l_1)$ to a 4-*LocSeq* using the loc-event $d(l_9)$, then an enumeration-based candidate generation method will generate the following five 4-*LocSeq*s:

$d(l_9)\rightarrow<a(l_9), f(l_5)>\rightarrow b(l_1)$
$<a(l_9), d(l_9), f(l_5)>\rightarrow b(l_1)$
$<a(l_9), f(l_5)>\rightarrow d(l_9)\rightarrow b(l_1)$
$<a(l_9), f(l_5)>\rightarrow< b(l_1), d(l_9)>$
$<a(l_9), f(l_5)>\rightarrow b(l_1)\rightarrow d(l_9)$

Note that we only consider it as one position when a loc-event e_k is inserted into an eventset E_i since they describe the same fact. For example, suppose the event is $d(l_9)$ and the eventset is $<a(l_9), f(l_5)>$. There are three different combinations by inserting $d(l_9)$ in $<a(l_9), f(l_5)>$; that is, $< d(l_9), a(l_9), f(l_5)>$, $<a(l_9), d(l_9), f(l_5)>$ and $<a(l_9), f(l_5), d(l_9)>$. All these combinations have the same meaning in that events $d(l_9)$, $a(l_9)$ and $f(l_5)$ occur at the same time. Here, we assume the events in an eventset are sorted alphabetically.

With this, it is clear that $d(l_9)$ can only be inserted into $<a(l_9), f(l_5)>\rightarrow b(l_1)$ at the position before the eventset $<a(l_9), f(l_5)>$. Otherwise, we will violate the temporal relationships defined by the set of 2-*LocSeq*s above as shown in Figure 2. Thus, FlowMiner will only generate a single candidate 4-*LocSeq*, that is, $d(l_9)\rightarrow<a(l_9), f(l_5)>\rightarrow b(l_1)$.

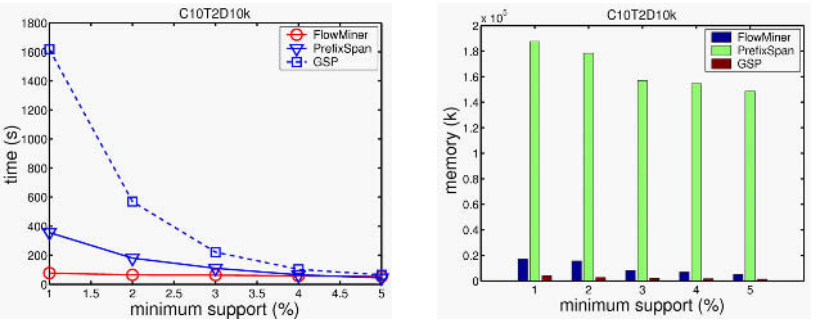
The above example demonstrates that the temporal relationships specified by all the 2-*LocSeq*s facilitate the elimination of a large number of infrequent candidates from being generated. This leads to a highly efficient candidate generation algorithm. Details of the algorithm to discover the location-sensitive sequence patterns can be found in Wang et al. (2004).

We carried out experiments to compare FlowMiner with GSP (Agrawal & Srikant, 1996), and PrefixSpan (Pei, Han, & Asl, 2001). Figure 3 shows the results for the synthetic dataset C10T2D10k. Figure 3(a) gives the runtime for

Figure 2. Temporal relationships support of candidates

Candidates	$set(d(l_9) + a(l_9)) = \{d(l_9) \rightarrow a(l_9)\}$	$set(d(l_9) + b(l_1)) = \{d(l_9) \rightarrow b(l_1)\}$	$set(d(l_9) + f(l_5)) = \{d(l_9) \rightarrow f(l_5)\}$
$d(l_9)\rightarrow<a(l_9), f(l_5)>\rightarrow b(l_1)$	√	√	√
$<a(l_9), d(l_9), f(l_5)>\rightarrow b(l_1)$	X	X	√
$<a(l_9), f(l_5)>\rightarrow d(l_9)\rightarrow b(l_1)$	X	X	√
$<a(l_9), f(l_5)>\rightarrow< b(l_1), d(l_9)>$	X	X	X
$<a(l_9), f(l_5)>\rightarrow b(l_1)\rightarrow d(l_9)$	X	X	X

Figure 3. Response time vs. maximum frequent patterns

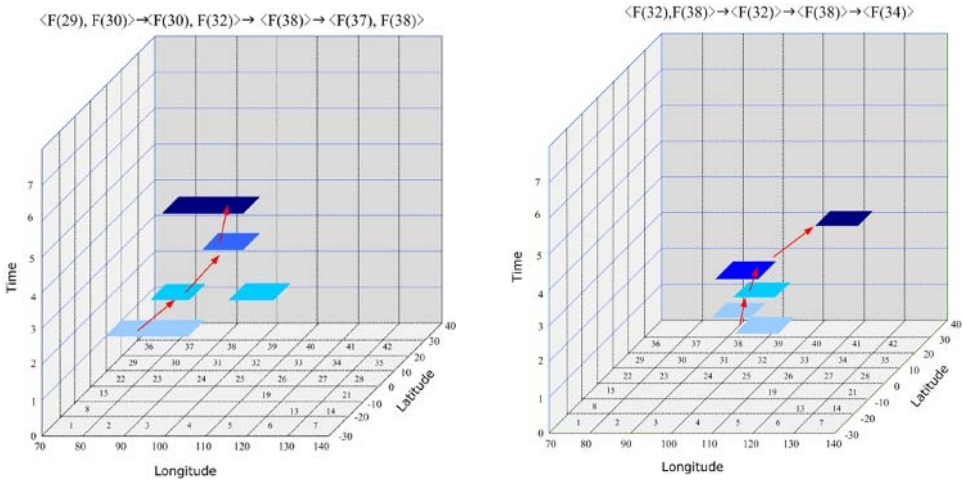


(a) (b)

varying minimum support, and Figure 3(b) records the amount of memory used by the three algorithms. We observe that FlowMiner outperforms GSP. PrefixSpan outperforms FlowMiner when minimum support is large, but when minimum support is low, FlowMiner outperforms PrefixSpan. The amount of memory used by both FlowMiner and GSP are much smaller than PrefixSpan. The results confirm that FlowMiner is more scalable as compared to PrefixSpan.

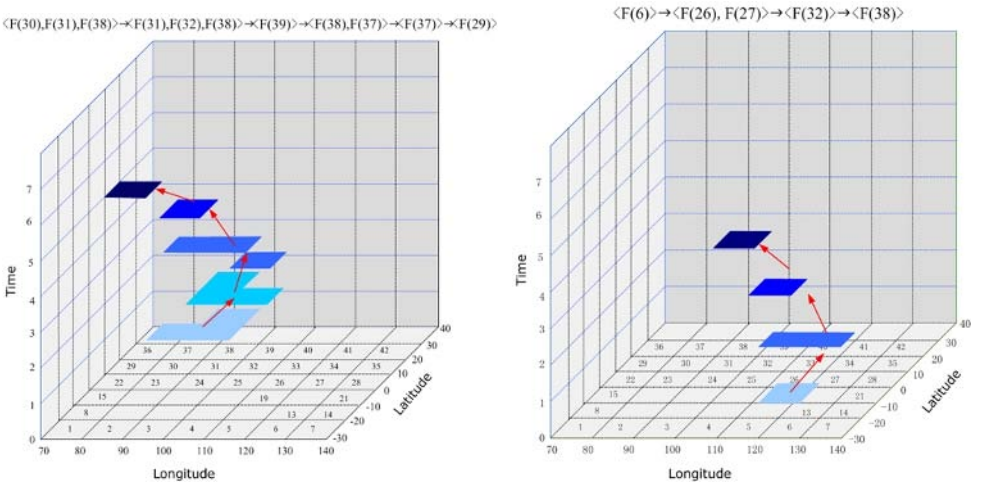
We also obtained two years of forest fire satellite images from a remote imaging centre. These images contain 2,495,097 forest fire occurrences. We divide the region into grids such that each region is 10 degrees in the longitudinal direction and 10 degrees in the latitudinal direction. This translates to 49 regions in our

Figure 4. Fire location-sensitive sequence patterns in forest fire dataset



(a) First Trend: From West to East in March and April

Figure 4. Fire location-sensitive sequence patterns in forest fire dataset (continued)



(b) Second Trend: From South to Northwest in April and May

database. Figure 4 shows a sample of the location-sensitive sequence patterns that we have discovered in this dataset. The location-sensitive pattern is shown at the top of the figure and the corresponding event fire in the patterns is indicated as F. The temporal relationship is reflected in the change of the color tone from light to dark while the spatial relationship is reflected in the color region. The fire spots indicate two distinct spread patterns. The first trend is from West to East (Figure 4(a)). This occurs mainly in March and early April. The second trend is from South to Northwest (Figure 4(b)), which happens in April and May. The fire spread patterns have been confirmed with the weather maps.

Patterns Involving Geographical Features

A special class of spatio-temporal databases is the moving object databases. Shekhar and Huang (2001) and Morimoto (2001) devised methods to find service requests frequently issued near each other in spatial databases. While such techniques are efficient in identifying the frequently co-located patterns, the patterns found may be too general for useful analysis.

Wang et al. (2004) observed that service requests are typically prompted by the surrounding geographical objects. For instance, when nearing the exit of a freeway, a driver would usually ask for information on the nearby restaurants and gas stations. If we are able to identify a set of requests for services that are always issued together, when certain geographical features present, then the service provider or area developer can take the opportunity to promote a set of services that are frequently associated with the same geographical features. Furthermore, suppose the group of subscribers issuing these co-located service requests shares some common characteristics. We may be able to discover useful information regarding the habitual patterns of this group of subscribers.

Consider the following frequently co-located service requests of SARS1 patients issued a few days prior to them being diagnosed as having contracted the disease SARS. These service requests do not reveal any hint as to the likely source of contamination of these SARS patients:

{*Restaurant, Entertainment Place, Taxi, Hotel*}; {*ATM, Taxi, Eating Places*}; {*Taxi, Hair Salon*}

However, if we incorporate the geographical features of these co-located service requests, then we have the following patterns:

{{*Airport*}::{*Taxi, Hotel*}}
{{*Supermarket*}::{*ATM, Taxi, Restaurant*}}
{{*Supermarket*}::{*Taxi, Hair Salon*}}

which indicate that both the airport and the supermarket have a high concentration of SARS patients. As a result, a more-detailed screening programme can be designed for these areas.

Geographical-Based NRS

We refer to the sets of service requests that are frequently located near each other in the spatial distance as *Neighbouring service Request Sets* (NRS). Finding the geographical features of NRSs involves mining patterns across two types of databases; that is, moving object databases and spatial databases. Moving object databases capture both the current and historical locations of moving objects. *Geographical features* are the objects in the physical world, such as roads, buildings, bridges, and so forth. In the spatial databases,

geographical features are indicated by a polygon or a minimum bounding rectangle (MBR).

Note that it is important to determine the conditions under which we consider two service requests as being "close" and when the service requests are "close" to geographical features. This is achieved through the use of two distance thresholds, denoted as D_n and D_g, respectively. Two service requests, r_1 and r_2, are considered "close" if the Euclidean distance between them satisfies $dist(r_1, r_2) \leq D_n$. Similarly, a geographical feature g is said to be interesting with respect to a neighbouring service request set S if it is always close to all the points in S, that is, $dist(S, g) \leq D_g$.

A geographical feature g is *frequent* if the number of instances exceeds the minimum support. We call the frequent geographical features of a *k-nrs* as the geographical features of *k-nrs*, denoted as $\{\{g_1, g_2, ..., g_n\}::\{r_1, r_2, ..., r_k\}, f_g\}$, where $\{r_1, r_2, ..., r_k\}$ is a *k-nrs*, and g_j $(1 \leq j \leq n)$ are the frequent geographical features of this *k-nrs*, and f_g is the minimum support.

Mining Geographical Features of NRS

In this section, we describe the process of discovering the geographical features of the frequent NRS from the two databases. This involves two steps: (a) Finding the frequent NRS from the moving object database; (b) Linking the geographical features in the spatial database to the frequent NRS.

Finding Frequent NRS

In order to find all the frequent NRS, we generate the candidate NRS and introduce an iterative method to count the support of the candidate NRS.

Candidate Generation: We observe that NRS satisfies the Apriori property; that is, *any subpattern of a frequent pattern is also frequent*. Hence, we use the Apriori approach (Agrawal & Srikant, 1994) to generate the candidate NRS.

The candidate generation process proceeds level-wise. At level k, the *k-nrs* $(k \leq 2)$ is generated by joining two $(k\text{-}1)$-*nrs*, $T1 = (S.r_1, \ldots, S.r_{k-2}, S.r_{k-1})$ and $T2 = (S.r_1, \ldots, S.r_{k-2}, S.r_k)$. Once the candidates are generated, we need to count the valid instances by performing pairwise distance comparisons of the k requests to ensure that the instances satisfy the distance threshold D_n. However, this is not feasible due to the exponential complexity of this algorithm. To overcome this, an iterative method called FindNRSSupport is introduced.

The basic idea behind FindNRSSupport is to make use of the centroid information to prune away the non-promising instances before performing the expensive pairwise distance computations.

The algorithm counts the valid *k-nrs* as follows: It first finds the closest pairs $<T1_j.o_j, T2_i.r_k>$ satisfying the distance threshold D_n, where $T1_j.o_j$ is the centroid of the instance $T1_j$. Then, it checks whether the pairwise distance among the k points in *k-nrs* satisfies the distance threshold.

Linking the Geographical Features in the Spatial Database to the Frequent NRS

To find the interesting geographical features of S, we first compute the centroid for each instance s_j of S. Next, we retrieve all the geographical features in the spatial database that lie within the user specified distance of D_g from the centroid. This is achieved by constructing a query window whose center is the centroid and whose radius is equal to D_g. Finally, we update the frequencies of the retrieved geographical features of S. Note that a geographical feature is only counted once for S in its one instance search, no matter how many instances of this geographical feature are close to the S instance.

Figure 5. Algorithm aprioriGSS

Algorithm AprioriGSS

Input: *DB*: *Moving object DB, Spatial DB*; *Dist-threshold*: D_n, D_g; *Minimum support*: f_n, f_g
Output: Geographical features of all *k-nrs*
1. F_1 = {frequent 1-*nrs* with geographical features}
2. **for** (k = 2; $F_{k-1} \neq \Phi$; k ++) **do**
3. C_k = CandidateGen(F_{k-1})
4. **for** all candidates $c \in C_k$ **do**
5. $c.sup$ = FindNRSSupport (c, D_n)
6. $C_k = \{c \in C_k \mid c.sup \geq f_n\}$
7. **for** all candidates $c \in C_k$ **do**
8. G = FindGeoFeature(c, D_g)
9. **for** all $g \in c.geofeature$ and $g \in$ G **do**
10. $g.sup$ + +
11. $F_k = \{c \in C_k \mid c.geofeature \neq \Phi$ and $\forall g \in c.geofeature, g.sup \geq f_g\}$
12. Answer = $\cup F_k$

Algorithm

Based on the above discussion, we designed an algorithm, called AprioriGSS. Figure 5 gives the details of the AprioriGSS algorithm. Lines 3 to 6 find the frequent *k-nrs*. Once the frequent *k-nrs* are found, Lines 7-10 search the spatial database to determine the geographical features of these *k-nrs*. The *k-nrs* that do not have any interesting geographical features will be pruned away even though they are frequent. Only the *k-nrs* with interesting geographical features are used to generate the candidates of (*k* + 1)-*nrs* (Line 11). This pruning strategy allows us to reduce the number of candidates generated.

Comparative Study

Finally, we show the usefulness of geographical-feature based NRSs as compared to co-located service requests. We generate service requests according to some common real-life habits.

We used 100 service request types, each of which occurs with a probability ranging from 0% to 70%. A correlation coefficient indicates the percentage of service requests of a given type that are close to a geographical feature. This coefficient *E* varies from 0% to 100% and it is treated as a parameter decided by the user. For each service request type, *E*% of its requests is generated close to a geographical feature based on the Gaussian distribution. The locations of

Table 1. Co-located service requests vs. geographical-based NRS

Co-Located Service Requests	Support Count
{ATM, Pharmacy, Entertainment Place}	2761
{Restaurant, Gas Station, Direction Guide}	1734
{Taxi, ATM, Shopping Mall, Hairdressing Salon}	1845
{Client's Office, Direction Guide, Restaurant}	3421
{ATM, Taxi, Restaurant, Client's Office}	2142
{Restaurant, Entertainment Place, Taxi, Hotel}	1421

(a)

Geographical-Based NRS	Support Count
{{Clinic}::{ATM, Pharmacy}}	1323
{{Shopping Complex, Hotel}::{Restaurant, Entertainment Place}}	1123
{{Freeway Exit}::{Gas Station, Direction Guide}}	2312
{{Airport}::{Taxi, Hotel}}	889

(b)

these requests are varied according to a deviation parameter $V = 50$ from the centroid of the geographical feature. The remaining 1-E% requests are generated using the uniform distribution. A total of 100,000 requests are generated.

We utilize the Singapore map as our spatial database, which contains 50 types of geographical features. Table 1 shows some of the interesting patterns we found and their support counts. When we show both sets of patterns to some decision makers, they unanimously prefer the geographical-based NRS patterns.

Conclusion and Future Research Direction

Mining in spatio-temporal databases is still at its infancy. While some may feel that spatio-temporal databases are nothing more than another type of high-dimensional database, we have seen the patterns that can be discovered in spatio-temporal databases go beyond just regarding the objects as points in a high-dimensional space. It is important to have a clear understanding of the relationships among events, time and space. Fundamental issues such as representation schemes, data cleaning, discovery algorithms and interestingness issues need to be examined in new lights.

Numerous research opportunities exist in the mining of spatio-temporal databases. Traditional data mining techniques do not deal adequately with problems of high dimensionality, uncertainty and spatial or temporal correlations. Future research directions in spatio-temporal data mining include:

- Incorporate spatio-temporal autocorrelation into standard data mining techniques like regression, classification, clustering and association rules.
- Create new similarity models and indexing techniques for higher-dimensional trajectory data; and efficient algorithms that do the trend/sub-trajectory matching queries.
- Create new measures of quality of rules generated, since traditional measures of support may not be meaningful here.

References

Abraham, T., & Riddick, J. (1997). Discovering meta-rules in mining temporal and spatiotemporal data. *Proceedings of the Eighth International Data-*

base Workshop, Data Mining, Data Warehousing and Client/Server Databases (IDW'97), (pp. 30-41).

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94),* (pp. 487-499).

Agrawal, R., & Srikant, R. (1996). Mining sequential patterns: Generalizations and performance improvements. *Proceedings of the Fifth International Conference on Extending Database Technology,* (pp. 3-17).

Chawla, S., Shekhar, S., & Wu, W. (2000). Predicting locations using map similarity (plums): A Framework for spatial data mining. *Proceedings of the International Workshop on Multimedia Data Mining, MDM/KDD 2000,* (pp. 14-24).

Chen, X., & Petrounias, I. (2000). Discovering temporal association rules: Algorithms, language, and system. *Proceedings of the 16th International Conference on Data Engineering,* (p. 306).

Ester, M., Frommelt, A., Kriegel, H.P., & Sander, J. (1998). Algorithms for characterization and trend detection in spatial databases. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining,* (pp. 44-50).

Ester, M., Frommelt, A., Kriegel, H. P., & Sander, J. (2000). Spatial data mining: Database primitives, algorithms, and efficient DBMS support. *Data Mining and Knowledge Discovery,* 193-216.

Garofalakis, M., Rastogi, R., & Shim, K. (1999). Spirit: Sequential pattern mining with regular expression constraints. *Proceedings of the 25th International Conference on Very Large Data Bases,* (pp. 223-234).

Han, J., & Kamber, M. (2000). Data Mining: Concepts and Techniques. *The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor.* San Francisco, CA: Morgan Kaufmann Publishers.

Han, J., Dong, G., & Yin, Y., (1999). Mining segment-wise periodic patterns in time-related databases. *Proceedings of the 15th International Conference on Data Engineering,* (pp. 106-115).

Han, J., Koperski, K., & Stefanovic, N. (1997). Geominer: A system prototype for spatial data mining. *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data,* (pp. 553-556).

Han, J., Pei, J., Asl, B.M., Chen, Q., Dayal, U., & Hsu, M. (2000). Freespan: Frequent pattern projected sequential pattern mining. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* (pp. 355-259).

Koperski, K., & Han, J. (1995). Discovery of spatial association rules in geographic information databases. *Proceedings of the Fourth Interna-*

tional Symposium on Large Spatial Databases (SSD), (pp. 47-66).

Kuman, V., Steinbach, M., Tan, P.-N., Klooster, S., Potter, C, & Torregrosa, A. (2001). Mining scientific data: Discovery of patterns in the global climate system. *Proceedings of the Joint Statistical Meetings.*

Lu, W., Han, J., & Ooi, B.C. (1993). Discovery of general knowledge in large spatial databases. *Proceedings of the Far East Workshop on Geographic Information Systems,* (pp. 275-289).

Mannila, H., Toivonen, H., & Verkamo, A.I. (1995). Discovering frequent episodes in sequences. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95),* (pp. 210-215).

Morimoto, Y. (2001). Mining frequent neighboring class sets in spatial databases. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* (pp. 353-358).

Muntz, E.M.R., Shek, E.C., & Mechoso, C.R. (1995). Exploratory data mining and analysis using conquest. *IEEE Pacific Conference on Communications, Computers, Visualization and Signal Processing,* (pp. 218-286).

Ozden, B., Ramaswamy, S., & Silberschatz, A. (1998). Cyclic association rules. *Proceedings of the 14th International Conference on Data Engineering,* (pp. 412-421).

Pei, J., Han, J., & Asl, B.M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings of the 17th International Conference on Data Engineering,* (pp. 215-224).

Pei, J., Han, J., & Wang, W. (2002). Mining sequential patterns with constraints in large databases. *Proceedings of the 2002 ACM International Conference on Information and Knowledge Management,* (pp. 18-25).

Ramaswamy, S., Mahahan, S., & Siberschatz, A. (1998). On the discovery of interesting patterns in association rules. *Proceedings of the 24th International Conference on Very Large Databases,* (pp. 368-379).

Shekhar, S., & Huang, Y. (2001). Discovery of spatial co-location patterns. *Proceedings of the Seventh International Symposium on Advances in Spatial and Temporal Databases,* (pp. 236-256).

Steinbach, M., Tan, P.N., Kumar, V., Klooster, S., & Potter, C. (2002). Data mining for the discovery of ocean climate indices. *Proceedings of the Fifth Workshop on Scientific Data Mining, (SDM 2002),* (pp. 7-16).

Steinbach, M., Tan, P.N., Kumar, V., Klooster, S., Potter, C., & Torregrosa, A. (2001). Clustering earth science data: Goals, issues and results. *KDD 2001 Workshop on Mining Scientific Dataset.*

Stolorz, P., Nakamura, H., Muntz, E.M.R.R., & Mechoso, C. (1995). Fast spatio-temporal data mining of large geophysical datasets. *Proceedings of the First International Conference on Knowledge Discovery and Data*

Mining (KDD-95), (pp. 300-305).

Tan, P.N., Steinbach, M., & Kumar, V. (2001). Finding spatio-temporal patterns in earth science data. *KDD 2001 Workshop on Temporal Data Mining.*

Tao, Y., Kollios, G., Considine, J., Li, F., & Papadias, D. (2004). Spatio-temporal aggregation using sketches. *Proceedings of the 20th IEEE Interntational Conference on Data Engineering,* 214-226.

Tobler, W. (1979). Cellular geography, philosophy in geography. *Gale and Olsson,* Eds. Dordrecht, Reidel. 379-386.

Tsoukatos, I., & Gunopulos, D. (2001). Efficient mining of spatiotemporal patterns. *Proceedings of the Seventh Interntational Symposium on Advances in Spatial and Temporal Databases,* (pp. 425-443).

Vlachos, M., Kollios, G., & Gunopulos, D. (2002). Discovering similar multidimensional trajectories. *Proceedings of the 18th IEEE International Conference on Data Engineering,* (pp. 673-684).

Wang, J., Hsu, W., & Lee, M. (2004). Discovering geographical features for location-based services. *Proceedings of the Ninth International Conference on Database Systems for Advanced Applications, (DASFAA),* (pp. 244-254).

Wang, J., Hsu, W., Lee, M.L., & Wang, J. (2004). FlowMiner: Finding Flow Patterns in Spatio-Temporal Databases, *The 16th IEEE International Conference on Tools with Artificial Intelligence.*

Wang, W., Yang, J., & Muntz, R. (1999). Sting+: An approach to active spatial data mining. *Proceedings of the 15th International Conference on Data Engineering,* (pp. 116-125).

Yang, J., Wang, W., Yu, P.S., & Han, J. (2002). Mining long sequential patterns in a noisy environment. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data,* (pp. 406-417).

Zaki, M. (1998). Efficient enumeration of frequent sequences. *Proceedings of the Seventh International Conference on Information and Knowledge Management,* (pp. 68-75).

Zhang, D., Gunopulos, D., Tsotras, V.J., & Seeger, B. (2003). Temporal and spatio-temporal aggregations over data streams using multiple time granularities. *Information Systems, 28,* 61-84.

Endnotes

[1] Severe Acute Respiratory Syndrome

Chapter XIII

Similarity Learning in GIS:

An Overview of Definitions, Prerequisites and Challenges

Giorgos Mountrakis, University of Maine, USA

Peggy Agouris, University of Maine, USA

Anthony Stefanidis, University of Maine, USA

Abstract

In this chapter we review similarity learning in spatial databases. Traditional exact-match queries do not conform to the exploratory nature of GIS datasets. Non-adaptable query methods fail to capture the highly diverse needs, expertise and understanding of users querying for spatial datasets. Similarity-learning algorithms provide support for user preference and should therefore be a vital part in the communication process of geospatial information. More specifically, we address machine learning as applied in the optimization of query similarity. We review appropriate definitions of

similarity and we position similarity learning within data mining and machine learning tasks. Furthermore, we outline prerequisites for similarity learning techniques based on the unique characteristics of the GIS domain. A description of specific methodologies follows based on the highly diverse attributes of GIS datasets (for example, text, images, video), and application examples are presented. We summarize previously set requirements and present future trends expected to emerge in the coming years.

Introduction

Databases containing geographic information have been through substantial changes in the past decade. Improvements in acquisition methods coupled with increased information dissemination through the Internet have altered substantially their characteristics. Nowadays geospatial databases are increasingly incorporating temporal information in order to enable novel analysis capabilities through spatiotemporal modeling. They may also incorporate diverse information types like video, text and sound, in addition to the traditional raster, vector and thematic types. All these changes are expanding the applicability of geographic information systems (GIS), while at the same time increasing the difficulty in modeling and managing their content.

Problem Description

The increase in geospatial information availability has been matched by an expansion of the relevant user community, and an increase in the complexity of demands by such users. Thus, even though there exist increasing volumes of geospatial information, there are very good chances that a GIS query may not have an exact match in a corresponding database. To address this problem, *similarity* algorithms have been developed. The goal of a similarity algorithm is to compare each query to the available information in the database and produce a metric (or ranking). This metric expresses how close (appropriate) each of the available answers is to the query. For example, when a user requests an aerial photograph from 1956, the similarity algorithm will compare the request (Time = 1956) to the temporal footprint of the aerial photographs available in a database, and will rank these photographs based on their similarity to the user request. A popular example of such a similarity algorithm is nearest neighbor: The photograph that is closest (temporally) to 1956 would be the best choice.

Traditional similarity algorithms have been widely accepted because their simplicity allows fast answers to user queries, accompanied by preference

indices describing how close an answer is to the query. In constrained environments where the usage of information does not vary significantly, similarity algorithms perform well. But what happens when there is a highly variable "expectation" of results? Geographic databases are a representative example of this diversity. For example, two users might perform the same query, seeking a satellite image with 1m ground pixel size (resolution = 1m). However, their preferences may vary significantly due to different usage/restrictions. For example, user A may accept a black-and-white aerial image of the same area, with ground resolution of 0.1m as 100% similar to his/her request, while user B may object to such an alternative because it does not offer the radiometric properties (that is, multispectral representation) of the satellite image.

Similarity Learning

To address issues like the above, a *similarity learning* algorithm is deployed. Similarity cannot be assessed based on a fixed set of universal metrics, but rather should make use of adjustable profiles. This way, results can be customized to take into account particular user needs and/or application requirements. The learning process in a similarity learning algorithm involves the identification of such profiles based on a training set provided by the user.

In the context of this chapter, *similarity learning* is defined as the task of capturing user/application similarity preference and incorporating this information in the query process to support the retrieval of geospatial information. Similarity learning has been an active research field for more than two decades. As computational power increases every year, more sophisticated algorithms are developed. In the field of GIS, one would expect that the recent information proliferation and the diversification of the user community would be accompanied by more demanding, customized queries that only a learning algorithm could address efficiently. But in reality, similarity learning, although popular in other domains, remains at an early stage in GIS applications. There are a few attempts to address the issue in the literature, but these seem isolated and concentrate on specific aspects of the problem (for example, addressing only spatial similarity and ignoring other attributes). On the other hand, there is an extensive collection of learning techniques in other domains that do not consider the unique characteristics of spatial databases. Transition of these techniques to GIS collections is feasible but leaves room for improvement. Furthermore, the move towards multimedia geospatial information (including, for example, sound) is extending the applicability of geospatial data, but at the same time is increasing the complexity of user preferences. This mandates advanced similarity-learning algorithms that can seamlessly merge this multi-type information.

Outline and Scope of the Chapter

In this chapter we provide an overview of similarity learning in GIS and spatial databases. More specifically, we begin with appropriate definitions of similarity as supplied by psychologists and applied in databases. We proceed to position similarity learning within the broader framework of knowledge discovery and data mining, with emphasis on issues related to machine learning. In the rest of the chapter we focus specifically on GIS. We outline prerequisites for similarity learning techniques based on special characteristics of the GIS domain, and provide a categorization of similarity attribute types, leading to a description of specific methodologies based on the highly diverse potential attributes/tasks. We conclude with a discussion on the applicability of current approaches, summarize previously set requirements and present future trends.

We should note that the goal of this chapter is to provide a general framework for the variety of similarity-learning tasks as they surface from the special characteristics of and the emerging trends in the GIS domain. Considering the current early stage of research in this area, the chapter serves as a survey that identifies unique requirements and challenges in addition to discussing explicit solutions.

Similarity Learning: Multi-Disciplinary Viewpoints

Before we proceed to examine similarity learning within GIS databases, it is appropriate to position this task within the broader research web. In this section we provide a brief overview, essential for a GIS researcher addressing similarity-learning issues though not necessarily an expert in related fields.

Our overview begins with definitions and realizations from psychology, to provide a more theoretical foundation for this practical task. We continue with a short description of similarity learning within data mining, and finally, we present similarity learning in the context of machine learning. A more specific analysis of the latter is presented in later sections, after the special characteristics of the GIS domain are established. These three fields (psychology, data mining, machine learning) reveal a multifaceted, challenging identity for similarity.

Similarity in the Context of Psychology

The intuitive path towards understanding similarity and the mechanisms that affect it begins in the field of psychology. Human ability to assess similarity is an important aspect of human perception, and cognition and plays a central role in theories of human knowledge representation, behavior and problem solving (Holt, 2000). One of the most influential works in psychology with respect to similarity comes from Tversky (1977). He describes similarity as "an organizing principle by which individuals classify objects, form concepts and make generalizations." In an earlier work, Quine (1969) proposed that our sense of similarity, our grouping of kinds, is both innate and accumulative; innate in terms of having the concept of similarity embedded in our inborn senses but also accumulative because people learn through maturation and development.

In another interesting work Popper (1972) stated that similarity between two things is always relative to a certain respect in which they are compared, a certain perspective or interest. In addition, Chi, Feltovich and Glaser (1981) showed that the basis for similarity changes with expertise. In their experiments they noticed that novice users tend to classify on the basis of superficial or surface features, while experts investigated deeper, underlying principles. Furthermore, Smith and Heise (1992) argued that the role of perceptual similarity in conceptual development has been substantially underestimated because of the tendency to view perceptual similarity as fixed. For in-depth analysis from the psychology perspective, the reader is referred to Medin, Goldstone and Gentner (1993).

Considering the particularities of spatial databases, we can recognize that current typical query processes partially support the above findings. For example, traditional exact-match queries do not conform to the exploratory nature of GIS datasets, while other non-adaptable query methods fail to capture the highly diverse needs, expertise and understanding of users querying geospatial datasets. Introducing similarity-learning algorithms in the communication process of geospatial information will enhance user capabilities. In the next two sections of our general literature review, we examine methodologies from the more practical point of view.

Similarity Learning in Data Mining

The similarity learning task can utilize methods from the wide range of data mining tasks. *Classification* is an example; it refers to learning a function that maps a data item into one of several classes (Hand, 1981). Similarity learning involves the classification of an input into one of several classes, based on its

similarity to these predefined classes. Several methodologies can be borrowed from classification, when the output of the similarity learning algorithm is discrete (that is, categorical). If the output is continuous, then *regression* techniques are more appropriate. These techniques are used to map a data item to a real-valued prediction value by learning a function that does this mapping. An important trend in recent years is the incorporation of temporal information in GIS. *Time series analysis* examines the value of an attribute as it varies over time; therefore, useful techniques can be borrowed.

The above tasks explicitly help in a similarity-learning process. In addition to these, there are others that can optimize the learning process without affecting the similarity learning per se. *Association rules* is one such example, where relationships are uncovered among data. Such analysis can help, for example, to learn dependencies between successive similarity queries; in other words, project future queries. *Clustering* is usually accomplished by determining the similarity among the data based on predefined attributes. It can be used as a pre-processing step. *Summarization* involves methods for finding a compact description for a subset of data. These techniques are often applied to interactive exploratory data analysis and automated report generation and can be integrated with the input/output of a similarity-learning algorithm, but not the learning process itself. Here we should mention that borderlines along these tasks are not crisp, since one task might borrow techniques developed for another; nonetheless, each of these tasks has its distinct methodologies (Dunham, 2002).

Similarity Learning in Machine Learning

Machine learning (ML) has proven to be a fruitful area of research, spawning a number of different problems as well as algorithms to their solutions. These algorithms vary in their goals, training datasets, learning strategies and representation of data. Applications of machine learning algorithms have found fertile ground in the database community for data mining purposes. Similarity learning is one of many applications of ML in databases. The current evolution of similarity-learning algorithms is the result of years of influence from different disciplines that ML encompasses, such as statistics, databases and artificial intelligence. Consequently, a major trend in the database community is to combine results from these seemingly different disciplines into one unifying algorithmic approach. Therefore, a multi-disciplinary approach is an inherited requirement for similarity learning tasks.

If we relate the problem of learning from data to the general notion of inference in classical philosophy, we can identify two main phases (Kantardzic, 2002):

- **Induction:** Learn or estimate unknown dependencies in the system from a given training set.

- **Deduction:** Use these dependencies to predict outputs for future input values in the system.

Induction can be seen as the progress from particular cases (training data) to a general mapping or model. On the other hand, deduction starts with a general model and, using given input values, it progresses to particular cases of output data. Clearly, similarity learning is an inductive task, since the model is not known in advance; it is identified through the training process.

There are two types of inductive learning methods, *supervised* and *unsupervised*. *Supervised learning* is used to estimate an unknown dependency from known input-output samples. A supervised approach learns by example. A training input should be provided together with some correct answers (outputs). The term "supervised" is used to emphasize that the output values are known; in essence, provided by a teacher. Under the unsupervised learning category we observe only the features and have no measurements of the outcome (that is, no output values are provided during the learning process). *Unsupervised learning* does not require a teacher; the learner forms and evaluates the model alone. The goal is to describe how the input data are organized (Hastie, Tibshirani, & Friedman, 2001).

Since most similarity-learning algorithms learn from example, they can be categorized as a supervised inductive task. The user is required to provide a similarity evaluation to a presented example, acting as a teacher for the algorithm. Popular ML methods that could be used include neural networks, decision trees, instance-learners, genetic algorithms and others.

Specific Prerequisites for Learning in Spatial Databases

In the previous section we positioned similarity learning with respect to data mining and ML. In this section we focus on similarity learning within GIS. We introduce a variety of desired properties that a learning algorithm should incorporate to address similarity within a geospatial environment as a result of the unique characteristics of the domain. These characteristics range from difficulty in formalizing the geographic domain and granularity issues to high data volume and diverse data types leading to conceptual dimensionality grouping and high dependencies among dimensions (attributes). An in-depth discussion can be found in Mountrakis, Agouris and Stefanidis (2004).

Beyond the traditional ML goals, such as scalability (algorithms should scale up to perform well with massive real-world datasets) and robustness (ability to

perform well consistently), GIS-specific demands require the following prerequisites:

i) **Complex design and integration.** The diverse nature of geographic data types imposes several challenges. For example, widely different techniques should be used for similarity learning applied to text, images and numeric database tuples. In other words, specific tools should be incorporated to address particular data types within this diverse group (specific tools for specific jobs). Integration of similarity results from all data types into one final product using a seamlessly unified approach for tools that might be extensively diverse is an additional concern. These tools might come from different disciplines, from statistics to artificial intelligence. Integration techniques should also handle potential *dependencies*, making this task even harder.

   Another issue that relates to non-linear systems is that as complexity increases it becomes more difficult to control system behavior. Ideally, a learning system should be complex enough to model the underlying problem but simple enough to train and analyze. Especially in geospatial/multimedia datasets, where the learning task is characterized by such diverse and complex properties, this so-called *transparency* (ability to assess the contribution of each processing element) of the system is essential. System transparency allows user interpretation and fine-tuning of each processing element behavior. Also, updating of the algorithm to new behavior could potentially require less effort. If the design is successful, even non-expert users can interfere during training, a desirable characteristic in every learning process.

ii) **Training set constraints.** A learning algorithm is as good as the given training set. In the case of similarity learning, the training set is provided by the user; therefore, user expertise plays an important role in the process. Novice users might have a vague idea of what they are looking for, while expert users are more demanding, thus more specific to the sample they would provide. This can affect the learning algorithm in several ways.

   First, the *type* of training set itself may vary. It can be continuous (for example, a percentage of similarity) or discrete (“similar,” “very similar,” etc). It can include a detailed relationship between different samples (for example, how much more similar is one from the other) or remain at a ranking level (sample 1 is more similar than sample 2). These training choices often dictate or eliminate the possible learning techniques used.

   Another important limitation is the available number of training samples; in other words, the *size* of the training dataset. It is somewhat cumbersome to rely on users to provide countless training samples before they start using the system. That is an inherent problem associated with the complexity of

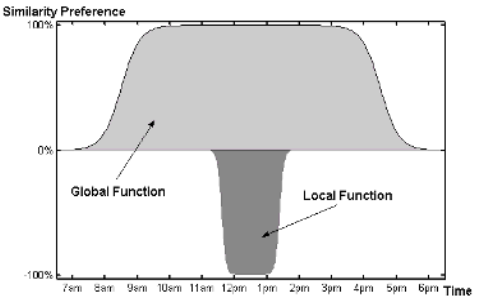
the domain and the dimensionality of the input learning space. Potential solutions include a multi-level training process where the complexity of the learning algorithm is adjusted based on the complexity of the problem (Mountrakis & Agouris, 2003), and incorporating domain knowledge in order to bound the hypothesis space. In the first case, during the training process the user is initially asked to supply a limited number of training sets. The learning algorithm should then investigate the underlying complexity and if necessary refine the training process through an intelligent relevance feedback technique where only "gray" areas are further processed. The incorporation of domain knowledge is discussed below.

iii) **Incorporation of prior domain knowledge.** One of the key elements to the successful development of a learning algorithm is the addition of explicit knowledge as expressed through rules, functions and design choices. In our case, this would translate into knowledge on the nature of similarity preference the user might communicate. This is not directly associated with ML, but can affect it in a great respect if ignored. Relevant studies are performed in the psychology discipline and cognitive scientists. Their findings are not always easily translatable; therefore, they cannot be directly incorporated into learning algorithms. Nonetheless, some fundamental rules of similarity behavior can be extracted and supported. For example, Shepard (1987) showed that similarity exponentially decreases proportionally to the distance of the candidate from the target value. Yet a large portion of similarity-learning techniques assumes a linear decrease of interest within each attribute and then attempts to recover from that using complex non-linear systems to aggregate similarity from each dimension. An example of incorporation of Shepard's findings can be found in Mountrakis and Agouris (2003).

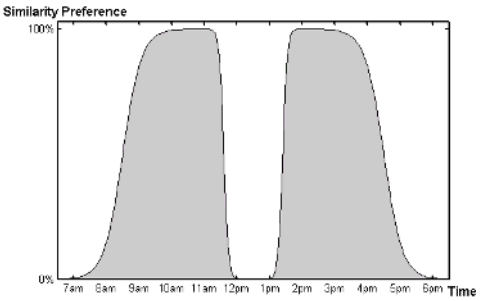
Geospatial information is the subject of numerous diverse applications. Accordingly, one could recognize multiple user profiles in terms of preferences. For example, a user queries for an aerial photograph taken in summer of 2000. In response to this request, the database could have two aerial photographs, one from winter 1999 and another from summer 1997. For the majority of users the 1999 photograph would have a better ranking than the 1997. But for some users (for example, biologists studying deciduous trees and therefore not interested in winter pictures), the 1997 photograph might be more appropriate. Identifying user groups that share common profiles and assigning each subsequent user to the correct profile group is a major challenge when designing learning algorithms. This profile grouping can be a result of a large number of extracted profiles and/or domain expert input for expected similarity behavior.

Similarity behavior types should also not be ignored by the training process. We can identify two types of similarity preference rules based on the

Figure 1. Temporal preference example



a) Global and local similarity functions



b) Their aggregation forming the final preference

applicability range, *global* and *local* ones. Global rules have influence on a large range of the input space of the algorithm, while local rules have more restricted range of application but usually great effect to justify their existence. Combination of both types may often be necessary to describe complex user preference.

For example, a user might request imagery to evaluate the parking lot congestion of a university campus. Naturally the user would expect an image during working hours (8am-5pm). This temporal similarity preference is shown with light gray in Figure 1a (upper part). However, in addition to this broad preference, another highly localized preference pattern also exists, whereby lunch break hours (12pm-1pm) should be excluded, as they do not reflect a typical situation. This negative localized preference is indicated by the dark function (Figure 1a, lower part). The overall complex preference is expressed by the aggregation of global and local functions (Figure 1b).

iv) **Uncertainty support.** Uncertainty is an issue that the GIS community has focused on in recent years. As data collection techniques become more comprehensive, uncertainty measures are included with the observations. We can recognize two interesting uncertainty-related issues for similarity learning: uncertainty in training samples and uncertainty of the database content.

- **Uncertainty in training samples.** User preferences expressed during a training sample selection are not deterministic values, and therefore introduce some uncertainty in subsequent similarity assessment. For example, a user may state during training that a 1999 map

is *80% or "very similar"* to a 1994 requested map, while at the same time provide a confidence value for the answer (for example, *90% certain or "almost certain" of this fact).* Subsequent similarity-assessment algorithms have to make use of this information by trying to fit more accurately samples with higher confidence than others.

- **Uncertainty in database content.** In this case, uncertainty is associated with the actual content of the database. Learning algorithms would have to learn how uncertainty affects users preference. For example, users might want imagery from 1999, but they also want the accompanied date to be accurate enough (for example, temporal uncertainty of a day) so that they can get additional data from other sources and reference them accurately in time with the requested imagery. To accommodate this, learning for this specific example should take place in two different dimensions, the *temporal* dimension and the *temporal uncertainty* dimension. Therefore, uncertainty information associated with the observed values would act as additional input in the learning process; in other words, as another training sample that the algorithm should model.

Similarity Learning in GIS

In this section we provide an overview of learning techniques applied to the highly diverse data types of a geographic database. Each object stored in the database is described by a set of attributes. The attribute *class* is a template definition of the variables for a particular kind of object. Thus, an attribute *instance* is a specific value of a class; it contains real values consistent with the class definition. Similarity assessment can be performed in these two levels, the *class* and the *instance* level. Attribute class-level similarity assessment provides an evaluation of the degree to which two different classes resemble each other semantically, and is described below. Attribute instance-similarity assessment aims at the evaluation of the degree to which two different values of the same attribute are similar, and is examined in detail in following sections. In these sections, and in order to assist our analysis of instance-based similarity learning, we follow a classification of data types which is by no means exhaustive or unique; it should rather be viewed as a basic organization of the wide variety of database content from the similarity learning point of view. A detailed discussion comes afterwards based on the data type organization. We examine each group of data types, what distinguishes them from others, and the applicability of corresponding similarity-learning techniques.

Semantic Similarity

This similarity assessment addresses the semantic comparison of a database object class to the corresponding class of the query object. The query formulation may be using terms/descriptors that differ from the ones in the database ontology. This is a challenge that has been acknowledged as the identification and resolution of differences in the definition of geographical concepts (Kavouras & Kokla, 2002). For example, if a user queries on "Image Scale" and the database has an attribute class described as "Ground Pixel Size," the question to be addressed is whether these two terms are similar, and if so, by how much. A learning algorithm could make appropriate customization for users. The issue we examine is often encountered in geospatial applications, due to the aforementioned trend to integrate heterogeneous repositories of information, with different ontological schemata used to describe their content. Within the context of this chapter, an ontology is defined as a type of knowledge base describing concepts through definitions that express the semantics of a domain. Its purpose is to reflect the relevance of data in a query process by providing a declarative description of semantic information independent of data representation (Goñi, Mena, & Illarramendi, 1998). Some representative metrics used for this semantic-similarity assessment are synonyms, common features and semantic relations of entity classes. For a recent application of ontologies in the spatial domain, along with a more detailed view on ontologies, the reader is referred to Rodriguez and Egenhofer (2003), Fonseca, Egenhofer, Agouris and Camara (2002), and for a discussion on spatiotemporal issues, to Grenon and Smith (2004).

Semantic similarity is not constrained to a direct attribute-to-attribute comparison at the class level. As mentioned before, attributes can be grouped together to form conceptual entities. For example, a Satellite Image entity in a spatial database might represent a number of attributes, such as scale, spatial coverage and time. The same can be claimed for another entity, Aerial Photograph. Even though with the proliferation of novel sensors the boundaries between these two concepts become fuzzy, they still have certain differentiating properties. The challenge from a similarity learning point of view is to understand the user perception of similarity between the concepts/entities of Aerial Photographs and Satellite Images. By using this information, advanced integration of multi-source retrieval can be achieved (if desired).

Within the GIS domain, semantic similarity has been a fairly active area for the past few years. Geographic semantic learning, though, has not caught up yet, mostly due to lack of generally accepted semantic metrics. Even so, we should mention a couple of initial steps, where semantic similarity is user-specific. Malerba, Esposito, Lanza and Lisi (2001) present an algorithm for learning concepts and dependencies among them in topographic maps. Also, Rodriguez

and Egenhofer (2004) have proposed a semantic matching algorithm (without the learning part), where the matching process supports contextual considerations; in other words, user-dependent relative importance in distinguishing semantic matching features.

Data/Attribute Type Organization

Geospatial datasets have evolved significantly through the years. Initially the spatial component was added to traditional databases, leading to spatial databases. More recently, advances in sensors and computing capabilities have enabled the collection and storage of multitemporal instances of spatial objects leading to spatiotemporal databases. Furthermore, geospatial datasets are beginning to be enhanced with multimedia capabilities by adding, for example, sound and annotations to generate prototype multimedia GIS. This evolution of information content in GIS databases is shown schematically in Figure 2.

As a result of this evolution of information content, GIS databases currently include a variety of data types, ranging from the traditional thematic, vector and raster data to dynamic (for example, temporal sequences, video) and multimedia data types. Different data types render themselves suitable to different similarity learning algorithms, so the following classification is important:

- *Thematic* information tends to be alphanumeric, and is not geometric or positional.
- *Vector* and *raster* data are traditionally associated with geospatial databases.
- By considering multitemporal information, we introduce *dynamic* datasets in our GIS as ones that capture temporally evolving GIS information.
- Finally, information such as (often unstructured) text annotation or sound is adding a *multimedia* dimension to traditional GIS databases. It should be noted that even though video datasets are often considered multimedia, under our classification scheme they are treated as dynamic datasets to differentiate between spatiotemporal and multimedia databases.

Figure 2. Evolution of information content in GIS databases

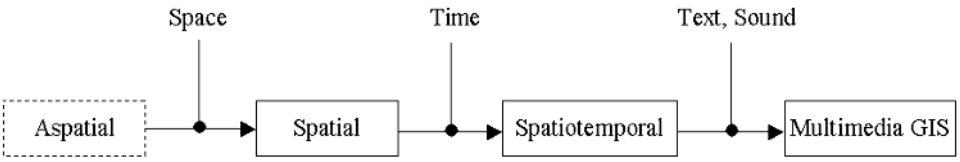


Table 1. State of the art in data type modeling for GIS applications

↓Database, Data Type →	*Thematic*	*Vector*	*Raster*	*Dynamic*	*Multimedia*
Spatial	■	■	■		
Spatiotemporal	■	■	■	▣	
Multimedia GIS	■	■	■	▣	□

(■ = advanced modeling solutions, ▣ = early formalization stage, □ = rudimentary solutions)

Table 1 offers a visualization of the above presented classification scheme of GIS databases and corresponding data types. Squares' filling signifies modeling progression in these data types. Dynamic data have been added the past decade, and there are some works addressing several spatiotemporal issues, but there is still a long way to go. Multimedia data have been included recently, so most research problems are still open. The dotted line between vector and raster type suggests that often database content supports one of the two types, but not both simultaneously.

Thematic Type

Thematic type attributes are not restricted to GIS datasets. Due to their extended applicability in the spatial domain, though, they have been the focus of research for geographers and cartographers for many decades now. Many taxonomies have been introduced based on different measurement scales. Such examples include counts versus measurements, qualitative versus quantitative, and metrical versus categorical measurements (Hand, Mannila, & Smyth, 2001). For our categorization, we make use of the four scales of measurement as introduced by Stevens (1946); namely nominal, ordinal, interval and ratio measurement types.

- **Nominal.** Data that do not have a natural ordering fall in this category. They can be numbers or text and they are used as labels or names. For example, the owner names of land parcels.
- **Ordinal.** These attribute types are ordered but do not express information about the differences between the ordered values. An example would be the values of "black," "gray," and "white" for color.
- **Interval.** An interval attribute has numerical distances between any two levels of the scale. They do not have a measurement origin, though. A typical example would be a temperature reading (Fahrenheit, Centigrade, and so forth).

- **Ratios.** When the attribute values have an origin in addition to being ordinal, then they belong to the ratio type. An example would be the percentage of free parking spots.

From the similarity perspective, *interval* and *ratio* scales of data require similar learning techniques. Their only difference relies on having an origin of measurement or not. This can be rectified through appropriate normalization, a common preprocessing step for data preparation for machine learning algorithms. It is important to mention that even though the type of data in this category is common with other non-geographic databases, similarity preference might not be. For example, the "Ground Pixel Size" thematic attribute can demonstrate diverse preference based on application usage. Photogrammetrists will probably be less flexible than oceanographers for larger pixel size, just as non-profit organizations will be more stringent towards smaller pixel size due to acquisition costs.

Ordinal data do have some relative order, but because this distance between ordered values is not quantifiable, regression techniques (for example, neural networks) are not easy to apply. Other methods, such as decision trees, might be more appropriate. As for the last scale category, the *nominal*, it is a textual matching process. Learning involves identification of possible relations between nominal values (for example, synonyms, same root). A thesaurus is often used, and learned domain knowledge is incorporated as these relations. Many methods used in this category overlap similarity learning especially in textual databases that are examined in more detail later.

Vector Type

Vector models are an essential part of most GIS databases. Spatial objects are represented as points, lines, polygons or hyper-polygons. We distinguish two categories of spatial vector matching; the single object similarity, where individual objects are compared, and the scene similarity, which is comprised of similarity assessment between collections of objects.

Single Object Similarity

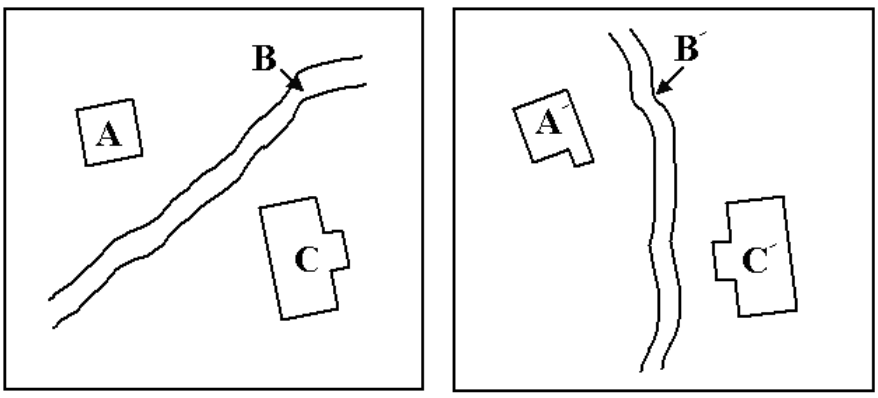
There are two major issues involved in object matching from the learning perspective: *geometric transformations* and *shape changes*. The first type of similarity learning involves identification of user preference towards geometric transformations. By geometric transformations we mean translation, rotation,

scaling and others. We assume that the query object can match a database object with application of one or more geometric transformations. Users are presented with such (query, database) pairs of spatial objects. The goal of a learning algorithm is to identify how each of these transformations affects similarity results. For example, when users query for a football stadium, they might be more tolerant toward rotation and translation but have a more strict preference towards scale. Note that the geometric transformations should be adjusted to the vector representation (for example, rotation and scaling do not apply to points, more than one scale for 2D polygons, and so forth).

In the second case, similarity learning attempts to capture user tolerance to *shape changes* that are not described by any known geometric models. In other words, this category contains algorithms for shape matching. There is substantial work in the literature addressing this issue. For a recent review of methods, refer to Veltkamp (2001). A simple example of a vector query is shown in Figure 3. The goal for a single object matching is to find a one-to-one correspondence similarity metric; in other words, A→A', B→B' and C→C'.

In general, we distinguish the following vector similarity approaches: ones that use the vector spatial representation as is, and ones that calculate shape statistics (descriptors), such as area, circularity, eccentricity, compactness, major axis orientation, Euler number, concavity tree and shape numbers (Ballard & Brown, 1982; Prokop & Reeves, 1992). In a learning process, relative weights and dependencies of these descriptors are captured. There are also approaches that transform the problem in a different similarity space (for example, using Fourier transformation).

Figure 3. Spatial vector query example



a. Vector query *b. Candidate vector response*

Scene Similarity

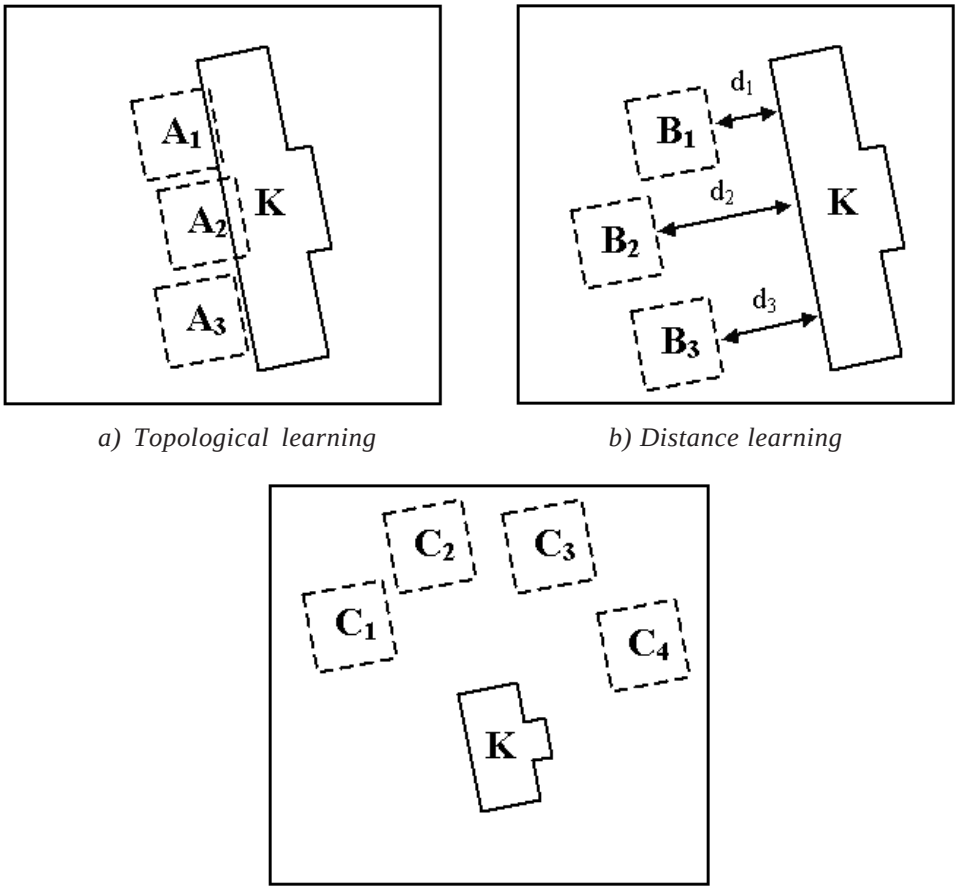
In some spatial queries, the user might request a set of objects that are spatially related to each other. Similarity in this case involves not only a one-to-one match between each object of the query and the database, but also the presence of some spatial neighborhood relations within each set of objects in the query and the database result. Thus, the single object similarity calculation described earlier as A→A', B→B' and C→C' (Figure 3) is substituted by a scene similarity of (A, B, C)→(A', B', C'). The examined spatial relations are classified into three basic types: topological, distance and direction, which can be combined logically to form more complex neighboring relations (Ester, Kriegel, & Sander, 2001).

i) **Topological.** Topological relations depend on the boundaries, interiors and complements of the two objects whose spatial relation is under examination. These relations are *disjoint*, *meet*, *overlap*, *equal*, *cover*, *covered by*, *contain* and *inside of* and were first formally introduced by Egenhofer (1991). From the similarity learning point of view, the goal is to model users' preference towards these relations. Even though these relations are binary, sometimes users demonstrate a certain degree of tolerance, depending on the application. For example, the *overlap* relation might require a minimum area percentage of the two objects to overlap, otherwise be characterized as *disjoint*; the *cover* relation a percentage of the boundaries to be common, otherwise it is considered a *containment* relation; the *inside of* relation could be satisfied if a common area threshold is met, and so forth. We should note that since the learning algorithm would model users' perceptions of these relations, users' definitions might contradict the strictly mathematical ones. An example of learning the *meet* relation is shown in Figure 4a. The user is presented three pairs (KA_1, KA_2, KA_3) and is requested to evaluate whether the meet relation holds true. It would be also worthy to investigate whether some users might be interested in a degree of membership to a relation or if it indeed is a binary one.

ii) **Distance.** Another popular spatial neighborhood relation is based on the distance between two objects. Depending on the definition of the distance metric, many distances are used; for example, the closest or the largest distance, the distance between object centers of gravity and others. Different users might have different preferences as objects get closer or further apart in a spatial scene (Figure 4b). This would be the subject of learning for similarity calculation.

iii) **Direction.** Directional relationships are frequently used in everyday lives. Because of their widespread use they are often incorporated as additional spatial relations. Some examples of such relations include *left of*, *above of*, *north of*, and so forth. It was soon realized that directional relationships are

Figure 4. Spatial scene relation learning



a) Topological learning

b) Distance learning

c) Directional learning

fuzzy concepts, since they depend heavily on human interpretation (Peuquet & Zhan, 1987; Takahashi, Shima, & Kishino, 1992). A learning approach would be able to facilitate this subjective nature of the problem. An example of learning the *north of* relation based on sample pairs (KC_1, KC_2, KC_3, KC_4) is shown in Figure 4c.

A learned characteristic could detect dependence between spatial relations. For example, the area of acceptance for a given direction increases with distance (Peuquet & Zhan, 1987). Scale and orientation, independent direction and distance similarity metrics integrated with standard topological relations can be

found in Stefanidis, Agouris, Georgiadis, Bertolotto and Carswell (2002). These metrics can be easily adjusted for learning.

Raster Type

In the last two decades, GIS systems have expanded from the traditional vector format to a raster type of information. Typical raster-type examples include imagery and digital elevation models. Advances in sensors have brought a variety of imagery directly available to GIS databases, imagery that can range from a close-range picture taken with an amateur digital camera to aerial photographs and satellite imagery.

Similarity search in image databases is a very active field of research with a considerable amount of published works (see Peng, Bhanu, & Qing, 1999). For our overview we present the general approaches implemented, and focus on typical GIS imagery examples. Image search can be performed either using data description or the data content, leading to description-based and content-based retrieval systems, respectively (Han & Kamber, 2001). Ideally, a combination of these two approaches should be incorporated in an image similarity algorithm.

i) **Description-based retrieval.** These systems perform object retrieval based on image descriptors, such as keywords, captions, size, time of creation and others. This information is also known as metadata description. Similarity learning within image metadata follows the same rules and methods of the thematic-type data, introduced earlier.

ii) **Content-based retrieval.** In this category the actual content of an image is analyzed in order to have a similarity result. Unfortunately, there are many distortions in image data to make this a trivial task; translations, rotations, non-linear distortions, scale variability, illumination changes and occlusion problems can significantly affect the image content. Distortion-invariant methods are currently known only in constrained visual environments (Hand et al., 2001).

The similarity match can take place using a feature representation of the image (*feature matching*) or the original image (*template matching*). *Feature matching* facilitates better indexing and storing of images, but transforms human perception of similarity into a predefined set of features that do not always completely capture the complexity of human intuition. Examples of features used include color statistics (for example, mean, variance, centroid and histograms), texture measures (coarseness/scale, contrast), shape descriptors (area, circularity, eccentricity) and wavelets (captures shape, texture and location in a single unified framework).

From the above features, statistics like color and texture ("low-level" properties) are computationally inexpensive and are often sufficient for general-purpose databases. GIS imagery, though, represents striking similarity in terms of general low-level properties (Agouris & Stefanidis, 1998). Therefore, shape features and also scene configuration metrics should be employed to achieve high classification accuracy. A learning process could further fine-tune results to user preference by extrapolation of important features and identification of dependencies among them. A few notable works are that of Cromp and Cambell (1993) and Ma and Manjunath (1996) for defining some characteristics for exploration of remotely sensed images and for using wavelets to learn similarity in aerial photographs, respectively.

The *template matching* category of content-based retrieval accesses the images directly, and not some reduced representation of them. Template-matching methods include correlation, least squares matching and neural network pattern classification (Brown, 1992). Learning in this case is comprised of relative weights of each pixel's contribution based on its own and neighboring gray values. For example, a user might be presented with a range of airplane images and in return show a degree of similarity to what the user is looking for. If a neural network is used to model this similarity, a multi-dimensional vector is created equal to the amount of pixels in the image. An attempt to extract hidden features takes place, features that represent how users evaluate similarity; in other words, what potential shape statistics and/or pixel spatial relations might contribute to the similarity assessment. Alternatively to direct raster learning, object extraction algorithms can be used so vector-type learning techniques can then be applied (see the section on vector types).

Dynamic Data Types

With the addition of temporal information to GIS, new query capabilities have risen. Objects do not show a static behavior, but are observed over a time period, creating a dynamic representation. We categorize this information as dynamic data type. It is in essence a temporally enhanced version of the previous data types; namely thematic, vector and raster. But the addition of time increases complexity significantly. One of the most important effects is the validity of information (Tao, Mamoulis, & Papadias, 2003). An object or a user preference might hold true for one temporal instance but be drastically different for another. This temporal dependency of information mandates that time is not treated like another attribute, but rather as a complex one.

If we examine dynamic types from the similarity learning perspective, we can identify two major categories; the *aspatial* (that is, thematic) *dynamic* information and the *spatial dynamic* one (that is, spatiotemporal). Within the context of

aspatial, temporally evolving attributes are examined. Space is used as a reference but not as part of the evolved attributes. For example, a query might be: "Find a school in this area that has the following temporal pattern in their student enrollment." This is a subject that has attracted a lot of attention due to its applications (for example, the stock market). Usually it is found under the term of time series analysis (Agrawal, Faloutsos, & Swami, 1993; Faloutsos, Ranganathan, & Manolopoulos, 1994). A popular similarity measure uses the longest common subsequence as a distance metric (Agrawal, Lin, Sawhney, & Shim, 1995). Rafiei and Mendelzon (1997) improve this technique by allowing transformations, including shifting, scaling and moving average, on the time series before similarity queries. Since methods from this field of research can be directly implemented for GIS data, we will not analyze them further.

Spatial dynamic information is one of the distinct characteristics of modern geographic databases. A variety of technological advances in acquisition systems has significantly increased the spatiotemporal (ST) information availability (for example, pocket-size GPS sensors and low-cost video cameras). Similarity assessment in ST datasets is the issue we examine here. In earlier sections we analyzed some similarity challenges in vector and raster data. The addition of time permits more complex queries, spatiotemporal pattern-matching ones.

Following the same organization of previous sections, we recognize two ST matching processes, single object and scene comparisons. We can also distinguish three major types of ST evolution: boundary changes, movement or a combination of both. These three types require observations from at least two different times, so a comparison can be performed (leading to ST change). There is also another type of ST queries, looking for a single boundary representation at a specific temporal instance where no implicit comparison exists. Here we omit the last kind, since it is indirectly addressed through ST pattern matching.

Spatiotemporal Single Object Similarity

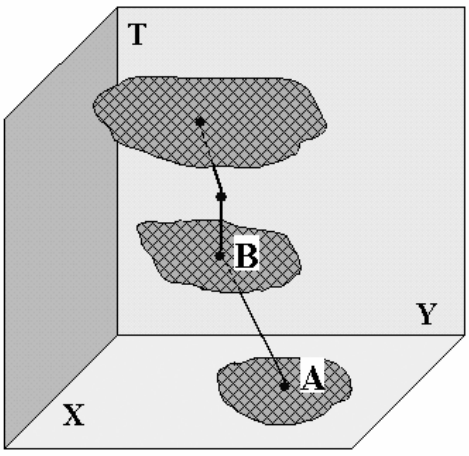
In this category we first examine similarity learning in boundary changes (deformation), in location changes (movement) and in a combination of both.

Boundary deformation corresponds to object shape variations through time. For example, a user might be looking for satellite imagery capturing an ocean oil spill. Assuming that the user knows the spill's starting time the user can approximate future states. Also, variable oil concentration might mandate different spectral bands at different temporal instances. A learning algorithm would be able to encapsulate this boundary preference and provide the user with the appropriate feedback; for example, by selecting among different satellite images or even aerial photographs.

Another type of ST pattern matching involves that of *moving objects*. A moving object is reflected by variations of the location of its center over time. The goal in this case is to compare object movement to a requested pattern. A fairly large amount of publications exist (for example, Vlachos, Gunopulos, & Kollios, 2002; Cotelo Lema, Forlizzi, Güting, Nardelli, & Schneider, 2003) providing different similarity measures. Learning user preference, though, in such pattern queries has not been addressed yet, to our knowledge. By doing so we would be able to adjust dependencies that go beyond distance metrics, like preferences that hold true for specific ST values. User preference may depend on the actual ST values requested and not solely on the distance metric between query and candidate result. For example, two candidate results might have the same ST "variation" to two queries, but because one query covers a specific ST area user preference might be different (for example, a moving car downtown during rush hour and a moving truck in the forest).

In more complicated scenarios, requested ST patterns might exhibit both boundary deformation and movement (Stefanidis, Eickhorst, Agouris, & Partsinevelos, 2003). The issue is more complex than a simple combination of the two individual approaches. For example, a query might request an ST pattern match of Figure 5. In one case, the users' goal is to find other days with similar pollution behavior from an industrial plan at point A. Thus, users are much more flexible as the ST pattern evolves into the future, but not at its start, as it defines the source of pollution (point A in Figure 5). Users may also want to enforce specific starting times for the pattern and/or be more accommodating to how fast the boundary changes. In another case, though, the same pattern may be describing the path of a pack of wolves. In that sense, point B may have a higher

Figure 5. Spatiotemporal pattern



weight in the similarity matching process, as it may represent a meeting point with another herd by a river. So users will be more (spatio-temporally) flexible on how the wolf pack behaved before or after the meeting than the ST footprint around point B. Of interest may be the continuous group behavior of this pack, examining, for example, whether they still form a compact unit or they scatter around, whether they move at a constant pace, and so forth. Such complex user preferences will be part of the learning challenges during the training process.

Spatiotemporal Scene Similarity

In this type of similarity learning, there is a pattern-matching process involving more than a single object. ST relations between objects can be identified as an extension of the spatial ones with the addition of the temporal dimension. These neighboring relations were defined earlier in this chapter as topological, distance and direction, but they have to be extended (and learned) in the spatiotemporal domain.

It is interesting to see how time affects these relations. Among others, Pfoser, Jensen and Theodoridis (2000) presented methods to answer topological and navigational queries in a database that stores trajectories of moving objects. The realization that some initial research exists is evident, but the lack of learning capabilities is a significant constraint. A successful GIS query process should be able to support different user preferences in ST scene queries, and not have a fixed-metric approach where all users are considered equal. We see this specific task as one of the most demanding and emerging ones from the variety of tasks mentioned in this chapter, due to its uniqueness to GIS databases and its difficulty to be resolved.

Multimedia Type

Multimedia information is often composed of images, video, text and audio sources. For our analysis, we distinguished images as a raster type of information and video as a dynamic type. The inherent spatial reference supported in images and videos mandates a more detailed examination, especially for GIS databases. In this section, we focus briefly on the remaining two multimedia types; namely, text and audio. For example, an audio query might be: "Find an audio that sounds like this, coming from area A and recorded in time T from provider P."

Even though audio and especially textual similarity have attracted significant attention, the incorporation of such types to geographic databases is still an open issue. Consequently, learning techniques have yet to be developed to operate within GISs. Possible dependencies between textual and audio similarity results

with other GIS attributes are an important task in order to tie these non-trivial data types to a comprehensive GIS similarity-learning algorithm. We see this task as a significant future trend, as GIS usage expands beyond the traditional spatial databases to complex multimedia ones.

Conclusion and Future Trends

The objective of this chapter was to provide an overview of similarity learning within GIS applications. It is evident that the highly diverse expectations of query results in GIS environments mandates a sophisticated adaptable approach to user preference. In the previous sections we investigated the application of similarity learning on geospatial datasets as a means for query return improvement. We provided similarity definitions and multi-disciplinary overlapping techniques to the learning process. We highlighted the special characteristics of GIS queries/objects, and discussed their influence on similarity learning. Characteristics such as diverse data types, dimensionality dependency and granularity issues, among others, impose several challenges that a successful similarity-learning algorithm should address. Along these lines, we presented some prerequisites like uncertainty support and knowledge incorporation.

We then focused on the diverse data types of modern GIS datasets. We presented complex similarity issues related to each type and showed potential goals for learning within each one. The realization that these tasks are far from trivial is evident. Another important challenge in a similarity-learning environment is the integration of highly diverse similarity types/algorithms into one unified technique. Issues of dependency between dimensions add to the already high complexity. A framework should be established for the integration of these multi-type matching results (for example, image-based, database-based, document-based) to provide answers to user requests.

In the coming years we expect a wide range of machine learning applications to rise in the GIS field. Similarity learning is one task that cannot be overlooked. As datasets and user numbers continue to grow rapidly, the need for query "personalization" will grow as well. The new era of "intelligent" geographic systems should be able to accommodate users with different expertise and backgrounds. The identification that temporal and spatial data are indeed special and need to be explicitly accommodated also is important. The goal could not be achieved with a single-disciplinary approach, but rather, it requires integration and cooperation of many fields of research. This is the main reason for not having the issue addressed already, even though the need for solutions is apparent.

Acknowledgments

This work was supported by the National Science Foundation through grant ITR-0121269, and the National Imagery and Mapping Agency through NMA 401-02-1-2008.

References

Agouris, P., & Stefanidis, A. (1998). Intelligent image retrieval from large databases using shape and topology. *Proceedings of the IEEE International Conference on Image Processing (ICIP) '98*, (vol. 2, pp. II779-II783).

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficiency similarity search in sequence databases. In D.B. Lomet (Ed.), *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO'93),* (pp. 69-84).

Agrawal, R., Lin, K.-I., Sawhney, H.S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. *Proceedings of the 21st International Conference on Very Large Databases*, (pp. 490-501).

Ballard, D.H., & Brown, C.M. (1982). *Computer vision.* NJ: Prentice Hall.

Brown, L.G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, *24*, 325-376.

Chi, M.T.H., Feltovich, P.J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, *5*, 121-152.

Cotelo Lema, J.A., Forlizzi, L., Güting, R.H., Nardelli, E., & Schneider, M. (2003). Algorithms for moving objects databases. *The Computer Journal*, *46*(6), 680-712.

Cromp, R.F., & Cambell, W.J. (1993). Data mining of multidimensional remotely sensed images. *Proceedings of the Second International Conference on Information and Knowledge Management,* (pp. 471-480).

Dunham, M.H. (2002). *Data mining: Introductory and advanced topics.* NJ: Prentice Hall.

Egenhofer, M. (1991). Reasoning about binary topological relations. *Second Symposium on Large Spatial Databases*. O. Gunther & H.-J. Schek

(Eds.), *Lecture notes in Computer Science*, (vol. 525, pp. 143-160). Springer Verlag.

Ester, M., Kriegel, H.-P., & Sander, J. (2001). Algorithms and applications for spatial data mining. In H.J. Miller & J. Han (Eds.), *Geographic data mining and knowledge discovery*. London: Taylor and Francis.

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (pp. 419-429).

Fonseca, F., Egenhofer, M., Agouris, P., & Camara, G. (2002). Using ontologies for integrated geographic information systems. *Transactions in GIS, 6*(3), 231-257.

Goñi, A., Mena, E., & Illarramendi, A. (1998). Querying heterogeneous and distributed data repositories using ontologies. *Information Modelling and Knowledge Bases IX*, 19-34.

Grenon, P., & Smith, B. (2004). SNAP and SPAN: Towards dynamic spatial ontology. *Spatial Cognition and Computation, 4*(1), 69-103.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Cambridge, MA: MIT Press.

Hand, D.J. (1981). *Discrimination and classification*. New York: Wiley.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction.* New York: Springer.

Holt, A. (2000). Understanding environmental and geographical complexities through similarity matching. *Complexity International, 7*, 1-16.

Kantardzic, M. (2002). *Data mining: Concepts, models, methods, and algorithms.* New York: Wiley-IEEE Press.

Kavouras, M., & Kokla, M. (2002). A method for the formalization and integration of geographical categorizations. *International Journal of Geographical Information Science, 16*(5), 439-453.

Ma, W.Y., & Manjunath, B.S. (1996). Texture features and learning similarity. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 425-430).

Malerba, D., Esposito, F., Lanza, A., & Lisi, F.A. (2001). Machine learning for information extraction from topographic maps. In H.J. Miller & J. Han (Eds.), *Geographic data mining and knowledge discovery*. London: Taylor and Francis.

Medin, D.L., Goldstone, R.L., & Gentner, D. (1993). Respects for similarity. *Psychological Review, 100*(2), 254-278.

Mountrakis, G., & Agouris, P. (2003). Learning similarity with fuzzy functions of adaptable complexity. *Proceedings of the 8th International Symposium on Spatial and Temporal Databases,* Lecture Notes in Computer Science, 2750, 412-429. Spinger Verlag.

Mountrakis, G., Agouris, P., & Stefanidis, S. (2004). *The unique characteristics of GIS databases.* Technical Report. Retrieved February 2003 from *www.aboutgis.com/publications/t1.pdf*

Peng, J., Bhanu, B., & Qing, S. (1999). Probabilistic feature relevance learning for content-based image retrieval. *Computer Vision and Image Understanding, 75*(1-2), 150-164.

Peuquet, D., & Zhan, C.-X. (1987). An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition, 20*(1), 65-74.

Pfoser, D., Jensen, C., & Theodoridis, Y. (2000). Novel approaches in query processing for moving objects data. *Proceedings of the 27 Conference on Very Large Databases,* (pp. 395-406).

Popper, K.R. (1972). *The logic of scientific discovery.* London: Hutchinson.

Prokop, R.J., & Reeves, A.P. (1992). A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphics models and image processing, 54*(5), 438-460.

Quine, W.V. (1969). *Ontological relativity and other essays.* New York: Columbia University Press.

Rafiei, D., & Mendelzon, A. (1997). Similarity-based queries for time series data. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (pp. 13-25).

Rodríguez, A., & Egenhofer, M. (2003). Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering, 15*(2), 442-456.

Rodríguez, A., & Egenhofer, M. (2004). Comparing geospatial entity classes: An asymmetric and context-dependent similarity measure. *International Journal of Geographical Information Science,* 17 (in press).

Shepard, R.N. (1987). Toward a universal law of generalization for physical science. *Science, 237,* 1317-1323.

Smith, L.B., & D. Heise (1992). Perceptual similarity and conceptual structure. In B. Burns (Ed.), *Percepts, concepts and categories.* Amsterdam: Elsevier.

Stefanidis, A., Agouris, P., Georgiadis, Ch., Bertolotto, M., & Carswell, J. (2002). Scale and orientation-invariant scene similarity metrics for image queries. *International Journal of Geographical Information Science*, *16*(8), 749-772.

Stefanidis, A., Eickhorst, K., Agouris, P., & Partsinevelos, P. (2003). Modeling and comparing change using spatiotemporal helixes. In E. Hoel & P. Rigaux, (Eds.), *ACM-GIS'03 Conference* (pp. 86-93). New Orleans: ACM Press.

Stevens, S.S. (1946). On the theory of scales of measurement. *Science*, *103*(2684), 677-680.

Takahashi, T., Shima, N., & Kishino, F. (1992). An image retrieval method using inquiries on spatial relationships. *Journal of Information Processing*, *15*(3), 441-449.

Tao, Y., Mamoulis, N., & Papadias, D. (2003). Validity information retrieval for spatio-temporal queries: Theoretical performance bounds. *Proceedings of the 8th International Symposium on Spatial and Temporal Databases*, (pp. 159-178).

Tversky, A. (1977). Features of similarity. *Psychological Review*, *84*(4), 327-352.

Veltkamp, R.C. (2001). Shape matching: Similarity measures and algorithms. *Proceedings of the International Conference on Shape Modeling and Applications,* (pp. 188-197).

Vlachos, M., Gunopulos, D., & Kollios, G. (2002). Robust similarity measures for mobile object trajectories. *Proceedings of the Workshop on Database and Expert Systems Applications,* (pp. 721-728).

About the Authors

Yannis Manolopoulos received his DiplEng in electrical engineering and a PhD in computer engineering from Aristotle University of Thessaloniki (1981 and 1986, respectively). He has been with the Department of Computer Science of the University of Toronto, the Department of Computer Science of the University of Maryland at College Park, and the Department of Electrical and Computer Engineering at Aristotle University. Currently he is a professor with the Department of Informatics, Aristotle University. He has published more than 150 papers in refereed scientific journals and conference proceedings. He is co-author of a book on *Advanced Database Indexing* and *Advanced Signature Indexing for Multimedia and Web Applications* by Kluwer. He served/serves as PC co-chair of the Eighth National Computer Conference (2001), the Sixth ADBIS Conference (2002), the Fifth WDAS Workshop (2003), the Eighth SSTD Symposium (2003), the First Balkan Conference in Informatics (2003) and the Sixteenth SSDBM Conference (2004). Also, currently, he is vice-chairman of the Greek Computer Society and chair of the Greek SIGKDD Section. His research interests include spatiotemporal databases, databases for the Web, data mining, data/file structures and algorithms, and performance evaluation of secondary and tertiary storage systems.

Apostolos N. Papadopoulos received his DiplEng in computer engineering and informatics from the University of Patras (1994) and a PhD in informatics from Aristotle University of Thessaloniki (2000). He has published several research papers in journals and proceedings of international conferences. From

March to August 1998, he was a visiting researcher at INRIA research center in Paris to perform research in benchmarking issues for spatial databases. Currently, he is a lecturer with the Department of Informatics, Aristotle University of Thessaloniki. He is a member of the Greek Computer Society and the Technical Chamber of Greece. His research interests include spatial and spatiotemporal databases, data stream processing, parallel and distributed databases, data mining and physical database design.

Michael Vassilakopoulos received a DiplEng in computer engineering and informatics from the University of Patras (1990) and a PhD in computer engineering from the Aristotle University of Thessaloniki (1995). He has been with the Department of Applied Informatics at the University of Macedonia and the Department of Informatics at Aristotle University of Thessaloniki. Currently, he is an associate professor with the Department of Informatics at the Technological Educational Institute of Thessaloniki. His research interests include access methods, spatiotemporal databases, WWW, multimedia and GIS.

* * *

Talel Abdessalem received an MSc from the University de Paris II (1992) and a PhD from the University of Paris IX - Dauphine (1997). During his PhD, he was involved in the research activity of the database group of the Computer Science and Decision-Aid Systems Laboratory (LAMSADE). In 1998, he joined the Computer Science and Networks Department of Télécom Paris (Ecole Nationale Supérieure des Télécommunications) as an associate professor. His teaching and research activities mainly concern database systems, XML technologies and GIS.

Peggy Agouris is an associate professor with the Department of Spatial Information Science & Engineering, University of Maine (USA). She holds a DiplEng from the National Technical University of Athens, and an MSc and PhD from Ohio State University. Her broad areas of expertise are digital image processing and analysis, geospatial information management and GIS. The total budget of her externally funded research activities at the University of Maine exceeds $5 million. Dr. Agouris has co-edited three books and published more than 50 articles in journals, books and refereed conference proceedings. She has served as chair or program committee member for various international conferences and workshops, and as an editorial board member and reviewer for scientific journals and national and international research foundations.

Nathaniel Ayewah is pursuing an MSc in computer science at Southern Methodist University, Texas (USA). His research interests include data mining and artificial intelligence. He has also conducted research in distributed knowledge capture, culminating in a publication at the K-CAP conference. He is currently working on a project to use Wavelet analysis to optimize noise reduction technologies in hearing aids.

Kathryn Bean graduated from St. Petersburg Technical University (1987) with an MSc in engineering in physics. After receiving her diploma, she worked in various research companies in the St. Petersburg area. Her research interests concentrated in the areas of computation methods in fluid dynamics and theory of probability. Upon her arrival in the United States, Kathryn worked for various information technology companies. In spring 2003, she received her second MSc, this time in computer science, from Southern Methodist University. With the assistance of Professor Margaret Dunham, Bean performed the data mining research that later, in part, became her contribution to this chapter. Since fall 2003, Bean has been working toward a PhD in computer science at the University of Texas at Dallas (USA), where she is concentrating her research in the fields of bio-informatics and data mining under the guidance of professor Sergey Bereg. Bean expects to complete her studies by spring 2005.

Karla A.V. Borges received her BSc in civil engineering from PUC-MG (1982) and her MSc in informatics and public administration from the João Pinheiro Foundation (1997). She is the leader of geographic data modeling efforts for the GIS project of Belo Horizonte, Brazil, and former head of the Urban Management Applications Department at Prodabel (Belo Horizonte's municipal information technology company). She also served as a program committee co-chair for the Fourth Brazilian Symposium on GeoInformatics (GeoInfo 2002). Currently, she is a PhD student with the Computer Science Department of the Universidade Federal de Minas Gerais, Brazil. Her main interests are geographic databases, geographic data modeling, urban GIS and ontologies.

Nieves R. Brisaboa received her first degree in computer science and a PhD from the University of A Coruña, Spain (1998 and 2003, respectively). She is founder and director of the Databases Laboratory at the University of A Coruña, from which she supervised the research in databases undertaken at this university. As director of the laboratory, she was the main researcher of 12 projects founded by the public administrations of Galicia and Spain. In addition, she was the director of various development projects founded by public companies (local administration of A Coruña province and the Royal Academy

of Galician Language). She serves as a reviewer for international journals and conferences, and supervised six PhD theses. Currently, she is an associate professor with the Computer Science Department, University of A Coruña. Her research interests include digital libraries, text retrieval, compressed text retrieval, deductive databases and spatial databases.

Joyce C.P. Carvalho received her BSc in computer science and MSc in computer science from the Universidade Federal de Minas Gerais, Brazil (1990 and 1994, respectively). Currently, she is a PhD student with the Computer Science Department of the Universidade Federal de Minas Gerais and a member of the database research group there. Her main interests include database modeling, semi-structured data, data integration, information retrieval and Web data management.

Antonio Corral received his BSc from the University of Granada (Spain) (1993), and his PhD in computer science (European Doctorate) from the University of Almeria (Spain) (2002). Since July 2003, he has been an assistant professor at the Department of Languages and Computation, University of Almeria. He has received several research scholarships in Spain (FIAPA and University of Almeria) and Greece (CHOROCHRONOS), and has published in several journal and conference proceedings. His main research interests include query processing in spatial databases and GIS.

Clodoveu A. Davis Jr. received a BSc in civil engineering from Universidade Federal de Minas Gerais (1985). He also has a MSc and PhD in computer science from the Universidade Federal de Minas Gerais, Brazil (1992 and 2000, respectively). He led the team at Prodabel that conducted the implementation of GIS technology in the city of Belo Horizonte, Brazil, and coordinated several geographic application development efforts. He participates as an associate researcher at the database group of the Universidade Federal de Minas Gerais. He also served as a program committee co-chair for the Fourth Brazilian Symposium on GeoInformatics (GeoInfo 2002). Currently, he is a professor at the Informatics Institute of the Catholic University of Minas Gerais, a researcher at Prodabel and editor of *Informatica Publica*, a Brazilian journal on information technology for the public sector. His main interests are urban GIS, geographic databases, map generalization and multiple representations in GIS.

Margaret (Maggie) H. Dunham (formerly Eich) received a BA and MSc in mathematics from Miami University, and a PhD in computer science from Southern Methodist University, Texas, USA (1970, 1972 and 1984, respec-

tively). Since August 1984, she has been first an assistant professor, associate professor and now professor in the Department of Computer Science and Engineering at Southern Methodist University, Dallas. In addition to her academic experience, Dunham has nine years of industry experience. Her current research interests are in mobile computing and data mining. She served as editor of the *ACM SIGMOD Record* (1986-1988). She has served on the program and organizing committees for several ACM and IEEE conferences, and also served as guest editor for a special section of *IEEE Transactions on Knowledge and Data Engineering* devoted to main memory databases as well as a special issue of the *ACM SIGMOD Record* devoted to mobile computing in databases. She was general chair of the ACM SIGMOD conference held in Dallas in May 2000. She is currently an associate editor for *IEEE Transactions on Knowledge Engineering* and is author of a recently published book, *Data Mining Introductory and Advanced Topics*. She has published more than 70 technical papers in various database areas. Dunham lives in Dallas with her husband Jim, daughters Stephanie and Kristina, and cat Missy.

Jost Enderle received an MSc in computer science from the University of Ulm (1999). He is currently working at RWTH Aachen University (Germany) as a research and teaching assistant in the group of Dr. Thomas Seidl. Enderle's research interests include the managing of complex objects in object-relational databases, especially the indexing and join processing of interval data. As a member of the German Standards Committee for Data Management and Data Exchange, he contributes to extensions of forthcoming SQL database language standards.

Dimitrios Gunopulos completed his undergraduate studies at the University of Patras in 1990, and graduated with MA and PhD degrees from Princeton University (1992 and 1995, respectively). He is an associate professor in the Department of Computer Science and Engineering, University of California - Riverside (USA). His research interests are in the areas of data mining, databases, Web mining and algorithms. In 2000, he received the US National Science Foundation CAREER Award. He has held positions at the IBM Almaden Research Center (1996-1998) and at the Max-Planck-Institut for Informatics (1995-1996). His research is supported by NSF, DoD and ATT.

Marios Hadjieleftheriou received a DiplEng in electrical and computer engineering from the National Technical University of Athens (1998). He is a graduate student with the Computer Science Department, University of California - Riverside (USA). Research interests include temporal, spatio-temporal and multi-dimensional indexing.

Wynne Hsu received her BSc in computer science at the National University of Singapore and her MSc and PhD in electrical engineering from Purdue University (1989 and 1994, respectively). She is an associate professor at the Department of Computer Science, School of Computing, National University of Singapore. Her research interests include knowledge discovery in databases with emphasis on data mining algorithms in relational databases, XML databases, image databases and spatio-temporal databases. She is a member of the ACM society.

Jie Huang received an MSc in computer science from Southern Methodist University (2002), and an MD and MSc in neurological science from China Medical University (1994 and 1997, respectively). While pursuing an MSc at SMU, Jie received the Outstanding Graduate Student award. Her research interest was studying novel Markov Model approaches to perform event prediction and forecasting. Particularly, Jie initiated the Extensible Markov Model research project, which is proposed to be applied in bioinformatic research, such as protein structure prediction, and so forth. Jie now is a PhD research fellow at University of Texas (USA), Southwestern Medical Center at Dallas, where she is studying biological signaling pathways using both molecular biology and bioinformatic techniques. Jie has published eight papers in medical and biological research fields since 1997.

Geneviève Jomier is professor and head of the research group in databases and software engineering at Lamsade Laboratory of Paris Dauphine University. She has also been a scientific advisor at INRIA, an assistant professor at Paris-Sud Orsay and Nancy Universities and a scientific researcher at CNRS. Her research domains have been modeling and performance evaluation of computer systems, distributed R-DBMS, OO-DBMS (O_2), versions in DBMS, software engineering, GIS, image, time series and spatio-temporal databases, evolution, and so forth. Her results include more than 80 articles in scientific journals and conferences, prototypes, academic and industrial projects. She has been involved in numerous national and international research collaborations.

Eamonn Keogh is an assistant professor of computer science at the University of California - Riverside, USA. His research interests are in data mining, machine learning and information retrieval. Several of his papers have won best paper awards, including papers at SIGKDD and SIGMOD. Keogh is the recipient of a five-year NSF Career Award for "Efficient Discovery of Previously Unknown Patterns and Relationships in Massive Time Series Databases."

Hans-Peter Kriegel received his MSc and PhD (1973 and 1976, respectively) from the University of Karlsruhe. He is a full professor for database systems and the chairman of the Department for Computer Science, University of Munich, Germany. His research interests are in spatial and multimedia database systems, particularly in query processing, performance issues, similarity search, high-dimensional indexing and parallel systems. Data exploration using visualization led him to the area of knowledge discovery and data mining. Kriegel has been chairman and program committee member in many international database conferences. He has published more than 200 refereed conference and journal papers. In 1997 he received the internationally prestigious "SIGMOD Best Paper Award" for the publication and prototype implementation "Fast Parallel Similarity Search in Multimedia Databases," together with four members of his research team.

Alberto H.F. Laender received a BSc in electrical engineering and an MSc in computer science from the Universidade Federal de Minas Gerais, Brazil (1974 and 1979, respectively), and a PhD in computing from the University of East Anglia (1984). He joined the Computer Science Department of the Universidade Federal de Minas Gerais in 1975, where he is currently a full professor and the head of the database research group. In 1997, he was a visiting scientist at the Hewlett-Packard Palo Alto Research Laboratories. He has served as a program committee member for several national and international conferences on databases and Web-related topics. He also served as a program committee chair (or co-chair) for the 19th International Conference on Conceptual Modeling held in Salt Lake City, Utah, in October 2000; the Ninth International Symposium on String Processing and Information Retrieval held in Lisbon, Portugal, in September 2002; the 18th Brazilian Symposium on Databases held in Manaus Brazil, in October 2003; and the Fifth ACM International Workshop on Web Information and Data Management held in New Orleans, Lousiana, in November 2003. He is a founding member of the Brazilian Computer Society and an editorial board member of the *Journal of the Brazilian Computer Society* and the *Information Systems Review*. His research interests include conceptual database modeling, database design methods, database user interfaces, semi-structured data, Web data management and digital libraries.

Mong Li Lee received her PhD, MSc and BSc (Hons 1) degrees in computer science from the National University of Singapore (1999, 1992 and 1989, respectively). She currently is an assistant professor in the School of Computing at the National University of Singapore. Her research interests include data cleaning, data integration of heterogeneous and semistructured data, and performance database issues in dynamic environments.

Zhigang Li is a PhD student in Southern Methodist University (USA) under the direction of Dr. Margaret H. Dunham. Li's research interests include data mining, Web mining, forecasting and database. In the past years he has published several papers in such conferences as KDMCD, PAKDD, WAIM and ICDM. He plans to finish his study in 2005.

Nikos A. Lorentzos received a first degree in mathematics from Athens University (1975), an MSc in computer science from Queens College, CUNY (1981) and a PhD in computer sciencefrom Birkbeck College, London University (1988). He has been involved successfully in a number of projects funded by the European Union. He was the technical manager of ORES, an ESPRIT III project aiming at the development of tools for the management of temporal data. Work completed under his supervision includes the specification, design and implementation of a temporal relational algebra, SQL extension and temporal DBMS. He serves as a reviewer for journals and conferences and acts on the program committees of conferences. He has participated in research projects, has acted as evaluator of programs submitted for funding to the European Union and as PhD examiner at various European universities. He is a co-author of the book *Temporal Data and the Relational Model* (with C. Date and H. Darwen). Today he is an associate professor at the Agricultural University of Athens, Informatics Laboratory. His research interests include data modeling, temporal and spatial databases, GIS and image processing.

Nikos Mamoulis received a DiplEng in computer engineering and informatics from the University of Patras (1995) and a PhD in computer science from the Hong Kong University of Science and Technology (2000). Since September 2001, he has been an assistant professor at the Department of Computer Science, University of Hong Kong. In the past, he has worked as a research and development engineer at the Computer Technology Institute, Patras, Greece, and as a post-doctoral researcher at the Centrum voor Wiskunde en Informatica (CWI), The Netherlands. His research interests include spatial, spatio-temporal, multimedia, object-oriented and semi-structured databases, constraint satisfaction problems.

Maude Manouvrier received a PhD in computer science from the University of Paris-Dauphine (2000). She is currently working as an assistant professor at Paris IX Dauphine University and is a member of the Databases and Software Engineering Group of the LAMSADE Lab. Since 1998, she has been working in an international CNRS-CONICIT cooperation with the Central University of Venezuela. During her doctoral studies, her research dealt with the management of similar, large, loosely or highly structured objects in databases. Currently her

research interests are focused on indexing mechanisms for querying the evolution of data (spatio-temporal data, images, time series, and so forth).

Claudia B. Medeiros received a PhD in computer science from the University of Waterloo (1985), an MSc in informatics from PUC-Rio and a degree in electrical engineering from the same university. She is a full professor of computer science at the Universidade Estadual de Campinas (UNICAMP), Brazil. She is the head of the database research group at this university, and her projects center on design and development of scientific databases applications, with emphasis on geographic data. She is an author or co-author of approximately 50 papers on databases and software engineering methodologies and has been (co)PI on more than 30 research and development projects — some of which involved partners in Germany, France, Argentina, Chile and the USA. Presently she is chair of the Latin America Liaison Committee for ACM SIGMOD, and president of the Brazilian Computer Society. She is a member of the editorial board of the *VLDB Journal* and Kluwer's *GeoInformatica*.

Eduardo Mena received a BSc in computer science from the University of the Basque Country and a PhD in computer science from the University of Zaragoza, Spain. He is an assistant professor in the Department of Computer Science and System Engineering at the University of Zaragoza, Spain, and is a member of the Interoperable Databases research group (University of the Basque Country and University of Zaragoza). His research interest areas include interoperable and heterogeneous information systems, semantic Web and mobile computing. His main contribution is the OBSERVER project. For a year he was a visiting researcher in the Large Scale Distributed Information Systems Laboratory at the University of Georgia. He has published several publications in international journals, conferences and workshops, including a book about ontology-based query processing. He has also served as a referee of international journals and as a program committee member of international conferences and workshops.

Giorgos Mountrakis is a PhD candidate in the Department of Spatial Information Science & Engineering and a research assistant for the National Center for Geographic Information and Analysis at the University of Maine, USA. He holds a DiplEng from the National Technical University of Athens and an MSc from the University of Maine. In the past he has worked on spatiotemporal GIS modeling and image processing applications. His current focus includes applications of machine learning and artificial intelligence in the GIS field. Mountrakis has published numerous papers in journals and refereed conferences. For more

information, please visit *www.aboutgis.com.*

José Moreira received a PhD in Computer Science from the Ecole Nationale Supérieure de Télécommunications de Paris and Faculdade de Engenharia da Universidade do Porto, his MSc from Université Pierre et Marie Curie - Paris VI and his undergraduate degree from Universidade Portucalense. He is assistant professor of software engineering at Universidade de Aveiro, Portugal. His background includes data structures, databases and network management. His research interests cover spatial and spatio-temporal database systems. The main topics are on moving objects representation in database systems and benchmarking.

Cédric du Mouza received an MSc in computer science from the University Pierre et Marie Curie - Paris VI (2000) and an MSc in advanced computer science from the University of Manchester (1999). He also holds an engineering diploma from the Institut d'Informatique d'Entreprise (IIE). He is a PhD student in the database group of the Conservatoire National des Arts et Métiers (CNAM) in Paris, supervised by P. Rigaux. His work focuses on the representation and querying of moving objects.

Dimitris Papadias is an associate professor for the Computer Science Department, Hong Kong University of Science and Technology. Before joining HKUST in 1997, he worked and studied at the German National Research Center for Information Technology (GMD), the National Center for Geographic Information and Analysis (NCGIA, Maine), the University of California at San Diego, the Technical University of Vienna, the National Technical University of Athens, Queen's University (Canada), and University of Patras (Greece). He has published extensively and been involved in the program committees of all major database conferences, including SIGMOD, VLDB and ICDE.

Martin Pfeifle received a degree in electrical engineering from the University of Cooperative Education in Stuttgart (1991) and a diploma degree in computer science from the Fernuniversität Hagen (2001). He currently works at the University of Munich (Germany) as a research and teaching assistant in the group of Dr. Hans-Peter Kriegel. Pfeifle's research interests include database support for virtual engineering, with a strong emphasis on spatial index structures and similarity search in spatial databases. Furthermore, he is interested in the area of knowledge discovery in databases, especially in density-based clustering.

Marco Pötke received a diploma degree in computer science (1998) and a PhD in database systems from the University of Munich (2001). He is currently working as a managing consultant and PLM specialist for AG&M, Germany, a member of the Capgemini Group. His major research interests are in the area of spatial query processing and knowledge discovery on spatial objects.

Katerina Raptopoulou is a PhD candidate in informatics at Aristotle University. She received a BSc in informatics (Honors) (1999) and an MSc in medical informatics (2002), both from the Aristotle University of Thessaloniki, Greece. Her research interests are spatiotemporal databases and, more particularly, the efficient indexing and querying of the moving objects' trajectories.

Philippe Rigaux holds an MSc in mathematics and computer science from the University of Paris VII. From 1986 to 1992, he was a software engineer in private industry. From 1992 to 1995, he was a research assistant in the database group of the Conservatoire National des Arts et Métiers (CNAM) in Paris, where he obtained his PhD (1995). He is currently an assistant professor in Laboratoire de Recherche en Informatique (LRI) database group. He has conducted research in the area of database management, strongly oriented toward spatial applications. His research interests include the design of query languages and query evaluation techniques. He has been involved in the program committee of various conferences in the field, such as SSD, ACM-GIS and SSDBM. He is co-writer of the book *Spatial Databases.*

Marta Rukoz received a PhD in computer science at Pierre et Marie Curie University (1989). From 1990 to 1999, she was coordinator of the Center of Parallel and Distributed Computing of the University Central of Venezuel. She currently is working at the University Central of Venezuela as a professor and head of the Postgraduate Studies in Computer Science. She has been working as an invited professor at the University of Central Florida (USA), the Technological Institute of Villetaneuse (France), and Pierre et Marie Curie and Paris-Dauphine Universities (France). Her work includes more than 15 papers in international journals and more than 20 participations at international conferences. Her investigation areas are parallel and distributed algorithms, distributed databases, image processing and image representation and retrieval.

Thomas Seidl received an MSc from the Technical University of Munich (1992) and a PhD from the University of Munich (1997). He is a full professor for computer science (data management and data exploration) at RWTH Aachen University, Germany. His research interests are data management and data

mining for complex data in large databases. For data mining, the focus is on effective and efficient similarity search, and applications include spatial and multimedia databases for medical images, molecular biology and mechanical engineering. Concerning data management, his interest is on relational database systems as a "virtual storage machine" for indexing structures in order to support efficient query processing for complex objects.

Altigran S. da Silva received a BSc in data processing from the Federal University of Amazonas - Manaus (1990), where he has been a lecturer since 1991. He received his MSc and PhD in computer science from the Federal University of Minas Gerais (1995 and 2002, respectively). Currently, he is an associate professor at the Computer Science Department, Universidade Federal do Amazonas (Brazil), and participates as an associate researcher with the database group of the Universidade Federal de Minas Gerais. He has been working on a number of research projects funded by Brazilian national agencies, including the National Research Council (CNPq), and has served as external reviewer and program committee member for conferences on databases and Web technology worldwide. In 2003, he was the organizing committee chair for the 18th Brazilian Symposium on Databases/17th Brazilian Symposium on Software Engineering. His main research interests include extraction and management of semi-structured data, Web information retrieval and database modeling and design. He is a member of the Brazilian Computer Society and ACM.

Anthony Stefanidis is assistant professor with the Department of Spatial Information Science & Engineering and the National Center for Geographic Information and Analysis at the University of Maine (USA). He holds a DiplEng from the National Technical University of Athens and MSc and PhD degrees from Ohio State University. His expertise is in the analysis of digital imagery and video for spatiotemporal applications and virtual modeling. Stefanidis has co-edited two books and authored numerous refereed papers on these topics. His research activities in the US are funded by the National Science Foundation, the National Geospatial Intelligence Agency, and other federal agencies and companies dealing with geospatial information.

Yannis Theodoridis received his DiplEng in 1990 and PhD in 1996 in electrical and computer engineering, both from the National Technical University of Athens. Since January 2002, he has been assistant professor at the department of Informatics, University of Piraeus, Greece, and also has a joint research position at the Computer Technology Institute (CTI). His research interests include spatial and spatiotemporal databases, location-based data management,

data mining and geographical information systems. He has co-authored one book published by Kluwer and more than 30 articles in scientific journals such as *Algorithmica*, *ACM Multimedia*, *IEEE Transactions in Knowledge and Data Engineering*, and in conferences such as ACM SIGMOD, PODS, ICDE and VLDB. His work has more than 250 citations in scientific journals and conference proceedings. He has served in the program committee for several conferences, including SIGMOD, ICDE and SSTD, and as general chair for the 8th International Symposium on Spatial and Temporal Databases (SSTD'03). He is member of ACM and IEEE.

Jose R. Rios Viqueira received his first degree in computer science from the University of A Coruña (1998) and a PhD in predoctoral scholarship from the University of A Coruña, Spain (2003). He was an active researcher in the CHOROCHRONOS TMR project that were funded by the European Union, where he undertook research in the area of spatial, temporal and spatio-temporal databases. He has been involved successfully in a number of research and development projects funded by the public administration of Galicia, Spain, the Spanish Government and the European Union. He acts as reviewer in journals and conferences. Today he is assistant professor with the department of Electronics and Computer Science, University of Santiago de Compostela. He also is a member of the Databases Laboratory of the Department of Computer Science, University of A Coruña and founding member and president of the board of directors of Enxenio S.L. His research interests include spatial and spatio-temporal data management.

Michail Vlachos holds a PhD from the University of California, Riverside. He received his BSc in informatics with highest honors from Aristotle University, and he was a recipient of the Fulbright Foundation scholarship for graduate studies. His research interests expand on the areas of data mining, databases, time-series, clustering and classification of multimedia data. Currently, he is a research staff member at IBM T.J. Watson Research Center, USA.

Junmei Wang received her BEng and MEng in electrical engineering from Xi'an Jiaotong University (1998 and 2000, respectively). She currently is a graduate student at the Department of Computer Science, School of Computing, National University of Singapore. Her research interests focus on knowledge discovery in spatio-temporal databases.

Ouri Wolfson received his BA in mathematics and his PhD in computer science from Courant Institute of Mathematical Sciences, New York University. His

main research interests are in database systems, distributed systems, transaction processing and mobile/pervasive computing. He currently is a professor in the department of Computer Science at the University of Illinois at Chicago, where he directs the Databases and Mobile Computing Laboratory and the newly established Mobile Information Systems Research Center. He served as a consultant to Argonne National Laboratory, the US Army Research Laboratories and the Center of Excellence in Space Data and Information Sciences at NASA. He also is the founder of Mobitrac, a high-tech startup company specializing in infrastructure software for location-based services and products. Before joining the University of Illinois he was on the computer science faculty at the Technion and Columbia University, and he has been a member of technical staff at Bell Laboratories.

Index

J

K

L

M

N

O

P

Q

R

S

T

U

V