SIMILARITY QUERIES IN LINEAR CONSTRAINT DATABASES

by

Ying Deng

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Peter Z. Revesz

Lincoln, Nebraska

December, 1999

Similarity Queries in Linear Constraint Databases

Ying Deng, Ph.D.

University of Nebraska, 1999

Advisor: Peter Z. Revesz

The abstraction ...

ACKNOWLEDGMENTS

I owe a special thanks to my advisor, Dr. Peter Revesz, who introduced me to this exciting area of research and gave me invaluable advice on the development of this dissertation, which would not exist without his motivation, guidance, patience.

I would like to thank Dr. Sharad Seth, Dr. Ashok Samal, Dr. Sunil Narumalani, for having made available their time and commitment to serve on my supervisory committee.

I would also like to thank my colleagues with whom the great pleasure of teamwork is discovered.

A big thanks to my friends who helped me and made my living in Lincoln so much fun.

I dedicate this dissertation to my parents for their endless love, always supporting and encouraging me.

Contents

Chapter 1

Introduction

1.1 Motivation

In the past decades, spatial database systems have attracted extensive attention because of the steadily increasing number of applications. The applications are found in architecture, environmental protection, computer aided design(CAD), very large scale integration circuit(VLSI) design, medical imaging and geographical information systems(GIS). Spatial database systems offer spatial data models and query languages. A formally defined semantics in terms of finite representation and efficient techniques to implement operations are critical for a spatial database system.

In *relational databases*, data and relationship between data are represented by tables with well structured data tuples. Program evaluation is bottom-up based on all the instances explicitly stored in the database. Optimization via algebraic transformation, selection propagation, etc. have obtained a great deal of achievements. The techniques of indexing and hashing make secondary memory access efficient and contribute to practical implementations. Some works use the relational data model for limited spatial database applications.[49].

In previously proposed *spatial databases*, the data modeling is based on vector and raster. However, not all problems can be expressed in these two models and further extensions are requested by many applications. With more and more functions being added to commercial systems, more attention is paid to the limitations of the employed models. Efforts to combine these two models do not overcome all the problems. "They combine only the power of the two models, but do not provide what is not included in one or the other"[16]. A spatial object typically consists of two kinds of information: meta-data and spatial features [17]. Meta-data is generally expressible as a scheme of a relational database. Spatial features or visual features derived from image processing or computer vision computations are traditionally based on discrete values of raw data. It has certain drawbacks such as sensitivity to noise, unsuitabilities for modeling moving objects and inefficiency for information retrieval. Extracting higher level information that are more robust and have a more intuitive meaning is expected.

In *constraint databases* a ground fact, or tuple, is a conjunction of constraints over a certain number of variables. Constraints can finitely represent data even in an infinite domain which is necessary for spatial and temporal. With a set of constraints:

the constraint data model is the infinite point set in space which fulfills all the constraints. Spatial data are expressed with simplicity and completeness. Spatial data manipulation involves accessing large volumes of spatial data and performing complex geometry computations, and thus requires the investigation on relationships between constraint programming and database query languages. Researches have shown that it is possible to combine the bottom-up, efficient declarative database programming and efficient constraint solving [34].

The constraint data model provides a finite representation of the unrestricted relational data model. The value of any attribute in the constraint data model is specified implicitly using variables and constraints [48]. In conjunction with the uniformed data representation, constraint tuples, constraint databases do not need to treat spatial objects as special cases and therefore simplify the query optimization problem, and offer a dynamic query evaluation based on the optimizers developed for constraint programming systems.

Constraint databases to model continuous variables and therefore images and spaces (not only discrete points) is novel. Constraint databases build on the current research in the database community, where constraint databases are being increasingly used to solve spatial problems. Constraint databases contribute to the design task for the layout of highways and provides solutions to many spatial design problems, such as the design of distribution networks for cable based communications etc[16].

We often hear people say that "daughters look like mothers" or "Nebraska Capital

Building is unique among other buildings". At first glance, similarity judgments like the above are deceptively simple because humans seem to be so good at it. Most three years olds may identify all the pictures of dogs in children's picture books, and in everyday life, people frequently use similarities to judge and describe things. However, experts in AI still can not write a program that would accomplish the picture identifications [14].

For spatial databases, the concept of a 'reasonable query' depends on the spatial properties that are used in the application. Similarity query, in its simplest form, can be stated as, retrieval or select all objects that are similar to the template object. There are a large number of applications associated with similarity query. Large collections of pictures or images, called image databases, already exist or are being created. With a large number of faces in a database, to find a similar face with good precision is a well-known application. Solutions to other problems, such as fingerprint identification used by police department and detectives, house blueprint selection encountered by real estate agents, are also typical applications. Image retrieval methods are divided into text-based and content-based image retrieval, which have been broadly noticed and studied by experts in fields of database management, computer vision, image interpretation, information retrieval, geographic information modeling and so on.

Conventional text-based image query methods depend on keywords or descriptive text associated with the images. In text-based image retrieval, the visual properties can not be accessed directly but only restricted by the particular vocabulary used to describe them. Especially, queries for shape similar to a given shape are not feasible.

Content-based Image Retrieval(CBIR) system helps users retrieval relevant images based on their contents. Previous database researchers advocate attribute based representation of image using conventional database systems. Whereas image interpretation researchers prefer to feature extraction and object recognition to overcome limitations of the former approach, but they suffer expensive computation, difficult modeling and tends to be domain specific. Current research in the purpose of combining the two approaches differ in terms of image features extracted, level of abstraction, and the degree of domain independence [23].

No matter which approach is used, queries facilitating CBIR involve features like color, texture, volume, motion, shape and other objective attributes. IBM's QBIC (Query By Image Content) system [14] uses image and video content as the basis of retrieval. Query by color, texture and shape are provided to users through a graphical user interface. Technology from this system has moved to commercial products and found use in a wide range of applications.

Exploring similarity-based retrieval is one of the main tasks for CBIR systems. There are diverse applications in which similarity query is an important activity: weather forecasting, architecture and engineering design, art gallery and museum management, picture archiving and so on. Data models with express power, query languages with comprehensive and efficient query processing, are at the heart of a similarity query technique. Further research and development work are scientifically deep-seated and commercially important.

Database systems allow user update information stored in it. Data insertion, deletion and other changes should be accomplished without user's knowing exactly how. The database system determines which tuples to add or eliminate to satisfy user's requirement and meanwhile keep the all data consistent. In relational database, query processing and information change are over constants, i.e., tuples contain only real instances. Integrity constraints can be easily maintained.

With constraint databases, information is represented as constraints. Users are not expected to know how constraints are changed but what are the change principle and results. A logical theory which can be used as the principle of database consistence is very important to guide the development. Although it is difficult to say what are good change operators, the principle of minimal change is agree upon by most people. From a model-theoretic point of view, Katsuno and Mendelzon [35] analyze the semantics of revising knowledge base represented by sets of propositional sentences. in terms of minimal change with respect to an ordering among interpretations, they give a characterization of revision schemes that satisfy Gardenfors's [1] rationality postulations. Revesz [51] proposes arbitration as the third kind of change operation beyond update and revision. He shows that arbitration operators can also be characterized as accomplishing a minimal change. The principle of minimal change states that the result of adding the new information to a database should be the set of models of the new information that are closest to some possible models in the current database. Hence the database change problem is largely reduced to find good measure for distance between all possible models. This principle can be applied to constraint database and some early work refers to [52]. It is interesting to see that how the change operators can be defined in terms of a similarity measure, and what are the characters the operators perform.

1.2 Related work

Deploying a similarity query technique involves three primary issues:

1. Data Representation. How can an object be represented in terms of its visual or spatial properties? How can the representation be extracted automatically or semi-automatically when a similarity query is act on to the database?

2. Similarity Measure. Similarity measure is to assess the deviation from equivalence. Given a representation scheme, how should any two objects or relations be compared or matched? What measures should be employed to determine the visual or spatial similarity(or dissimilarity) of them?

3. Query Method. How should the tuples in the database be organized to enable efficient search for relations that are similar to a given query template? What indexing mechanism should be employed?

Recent progresses are impressive and existing similarity query techniques resolve these issues in various ways.

There are two common approaches. In a model-driven system, each picture which is called as a model, is used as a test target, for which the input model is compared individually against each shape in the database, or against a number of features chosen to identify shapes to find a match. Model-driven techniques were thought to be not well-suited for spatial information retrieval because of their linear time complexity with the number of models.

In recent years, the data-driven approach has emerged. In a data-driven technique, an index structure organizes the known picture or model collection. Given an unknown model, the index is searched to find matching models.

A few techniques have been proposed for spatial search or similar shape retrieval.

Jagadish's technique [26] uses a rectangular cover for featuring a shape. Each shape in their database consists of an (ordered) set of rectangles. The ideas of sequential description of an object's feature is applied to rectangular covers. The shape is represented by means of the relative positions and sizes of these rectangles. Their similarity measurement is based on area difference which counts the unoverlapped area when one shape is placed "on top of" the other. They also proposed a spatial indexing based on shape similarity rather than spatial location, as other standard spatial indexing methods do. Partially occluded objects can't be dealt with under their method. In addition, some shapes may have more than one sequential descriptions which leads to ambiguity into the similarity and there are no good solutions for this potential problem. Either the size of the database has to be multiplied to keep all the sequences, or special efforts at query time are required. The first option is not feasible because it requires too much space, while the other is slow.

The visual shape of an object is very difficult to encode using traditional relational databases. Information loss is an unavoidable problem. Mehrotra and Gary's approach [41] computes the boundary segments of a shape. An ordered sequence of boundary points represents the shape's feature. A pair of points are arbitrarily chosen to form a "basis vector", which is used to normalize the coordinate system and encode a set of features. Information loss is inevitable here because the normalization process hides distortions of a picture via regular transformations of scale, rotation and translation. Both rigid and articulated shapes are within the handling ability of their technique. An articulated shape is represented by a collection of its rigid components which is identified with the user's involvement and assistance.

They define the similarity between two shapes as the Euclidean distance between the normalized feature vectors. The index design and search is based on the ordered set of points of a boundary. Similar shapes are found along the tree-structured index search until the leaf which contains a information on location and shape feature is reached and the shape similarity constraint is satisfied.

Since the basis vector could be any boundary segment and it plays a critical role in the retrieval process, users have to be very careful with the selection of query features, which change with the user's point of view. The query is not robust. Such abstract-level problems as noise sensitivity or distortion effects of sharp convex angles or larger line segments should not be explicit and presented to the user.

Boundaries of spatial objects are usually represented by rectangular bounding boxes. Jagadish [25] proposes a technique with polyhedra to reduce the redundant boundary area for an object and therefore to overcome some problems introduced by the rectangular boundaries. With polyhedra some improvement is obtained on more precisely representing boundaries, but this method requires more complex computations. The P-tree index structure associated with their technique requires extra bounds that have to be stored and checked at every access.

Their techniques can not ideally deal with spatial design problems. Let's look at an example. Consider the query that asks which states Highway I-84 passes through from Potland to Ogden. With the techniques employing rectangular boundaries, 7 states (Washington, Idaho, Montana, Oregon, California, Nevada and Utah) are retrieved for this query. Actually the highway goes through only 3 of these states. If polyhedral boundary is used for the highway, some but not all irrelevant states are excluded.

Using the constraint data model [34], the highway is represented as a collection of line segments and the states are represented as sets of polygons. The query for the intersection of the highway and the states can be easily expressed and efficiently evaluated using the MLPQ/GIS system [33]. Precise result is generated without any redundant storage.

Another example is in noise removal for fingerprints. Observe that there are a large number of line segments in the fingerprint and it is extremely hard to avoid erroneous intersection of rectangular bounding boxes with polyhedra. In constraint databases, this problem does not occur. The join of two line segments with very close endpoints is guaranteed not to intersect with any other line segments already present.

Researchers have also made efforts on data modeling with constraints and system development. Vandeurzen et. al. [62] proposed a linear spatial database model in which the representation and manipulation of both non-spatial and spatial data are based on first-order logic over the real numbers with addition. By presenting a general, variable- dimensional, linear spatial database model as a formal framework, They tried to bridge the gap and combine the benefits of two main approaches : models based on fixed spatial dimension and models based on variable spatial dimensions. Their proposed model uses seme-linear sets as spatial data type. They propose a declarative, calculus-like query language, FO+linear which can express a lot of practical queries. (However it can not be considered as a fully quantified querying tool for linear spatial database because certain natural linear queries , such as collinearity or computing convex hull of a finite set of points cannot be expressed. This problem is not merely a deficiency for some particular model, but a fundamental issue because no safe extension of FO+linear exists to be complete for linear spatial queries.)

The MLPQ/GIS system [33] developed in the computer science department of UNL is based on linear constraint database. The internal data representation uses linear constraint data model which is proposed to be a common basis for geo-spatial-temporal data because of its expressive power, efficient query evaluation and convenient data integration. MLPQ/GIS will play an important role in next generation database systems. The MLPQ/GIS system provides two kinds of queries, icon-based and Datalog- based queries. The basic queries are input through the graphical user interface, then translated into a procedural algebraic language and optimized using efficient algebraic evaluation algorithms, and finally evaluated. Not only the fundamental operators, such as selection, projection, join, union, and so on are implemented in this system based on linear constraints, but certain aggregation functions like area, buffer, etc. facilitating geographic and temporal data manipulation. Animation of dynamic spatial-temporal information has been deployed and embedded in the system [40]. With constraint relations, the spatial data that change with time passing, are stored non-redundantly and accessed efficiently. Advantages compared with representative commercial GIS products are non-trivial and presented in [33].

1.3 Contributions

The goal of this thesis is to address the problem of similarity-query, where pictures or images in a database that satisfy the specified similarity constraints with respect to the query picture or image must be selected from the database. We propose a novel method which is based on constraint data model.

By applying existed edge detection algorithms in Artificial Intelligence area, we transfer photos or pictures to stick figures, which are represented using constraint database. The global geometric features of a picture are counted for the similarity measure. When a new picture is submitted to find a similar picture in the database, it will be compared and similar pictures from the database will be matched. The following figure illustrates the procedure.

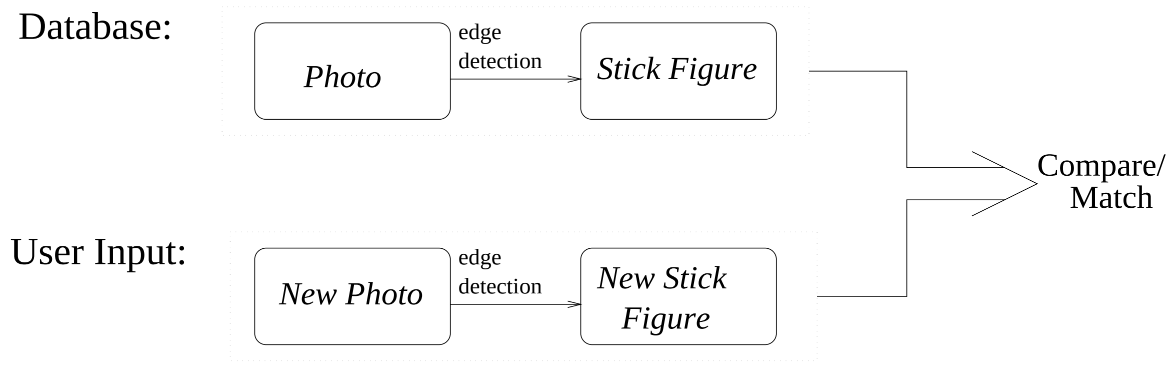

Figure 1.1: Similar Picture Query

A stick figure, or sketch is popularly recognized as an epitome which approximates reality and often is sufficient for conveying meaning and geometries of a picture. An alternative way allows user to construct shapes by drawing the sketches on screen using tools. Based on the distance measure, similarity of pictures is no more ambiguous and similarity query is efficiently evaluated, involving comparing of individual sketch or sets of sketches in the constraint database.

Rarely will two sketches be exactly the same, the shape, the direction, the position and the scaling may be vary significantly or slightly different so that not sensitive to human visual system. To achieve the possibly best measure, this thesis present experimental investigation which quantitates and computes the differences between human intuition and automatic data extraction and measure. Analysis result from the experiments well guides the refinement of the method.

In this thesis, we also study updating of constraint databases. Concrete change operators are presented based on principles of model-theoretic minimal change. We show that the revision, update, and arbitration operators proposed in this study can be characterized by satisfying axioms in this area.

1.4 Organization

The outline of this thesis is as follows.

Chapter 2 gives an in-depth introduction to constraint data models and constraint query languages. Indexing techniques with constraints are also described. Then the representation and manipulation principles for spatial information using constraint databases are illustrated with several examples.

In chapter 3, a novel method of similarity measure is deployed. We give details in how the distance between two pictures are measured. Some assumptions are made which we also describe for the sake of simplicity without loss of genericity. The core technique used in our method is the matching method solving assignment problems. We show by examples. how this method can be useful in definition and evaluation of the measures.

In chapter 4, based on the distance measure proposed in chapter 3, we study the characteristics of change operations for constraint databases. Revision, update and arbitration operators are defined in terms of model-theoretic minimal change. The concrete operators presented are proved to satisfy several axioms commonly accepted by the database community.

In chapter 5, we present the algorithm employed to implement the change operators based on the distance measure. The computational complexity of the algorithm is analyzed. We also make several extension to this algorithm and improve its analysis.

In chapter 6, analysis and discussion of the experimental investigation is shown to study whether our similarity measures agree with human intuition. Using experimental investigation, a large number of pictures are used to compare the difference between our similarity measure and human evaluation of similarity.

In chapter 7, we summerize the current research and point out further directions. Some open problems are listed as an encouragement for other researchers.

Chapter 2

Constraint Databases

Database systems study how to provide good ways to store, manipulate and view data. Relational databases are organized as a set of relations which are represented as tables. Each attribute has an entry in the table. A tuple is an instance of all attributes contained in the table. The relationship between attributes and relations are defined in schemes. Constraint databases extend relational databases and provide significant convenience and efficiency improvement for spatial applications. In this chapter, we review how constraint databases accomplish such a goal by representing, querying and indexing data with constraints, while preserving the merits of relational databases.

2.1 Constraint Databases and Query Languages

For a system to be usable, it must retrieve data efficiently. This concern has led to the design of complex data structure for the representation of data in databases. The complexity is hidden from users through several levels of abstraction. The top most level of abstraction, called the *view level* presents data in particular ways that may be convenient to a specific group of users. The middle level of abstraction, called the *logical level* describes what data are stored in the database and what relationships exist among those data. The lowest level, called the *physical level* describes how the data are actually stored in a computer system. Constraint data models are developed in the logical level. The various levels of data abstraction are linked together by translation methods. Links between the constraint level and the physical level involve implementation of basic storage structures, (e.g. R-trees or B-trees), paging methods, disk location of files etc. Links between the view level and the constraint level provide techniques for displaying the data. Combining constraint relations with data displaying based on constraint queries is very important for the tasks towards practical constraint databases.

A framework for using constraint databases is presented in [34]. The following three definitions are from [34].

Definition 2.1.1 *A* generalized k-tuple *is a quantifier-free conjunction of constraints on k variables ranging over a domain δ. Each generalized k tuple represents in a finite way an infinite set of regular k-tuple.*

A generalized relation of arity k *is a finite set of generalized k-tuples with each k-tuple over the same variables.*

Suppose relation R contains the set of points on the line with slope four. It is impossible for a relational database to enumerate all the points on the line, however the line can be finitely represented by a generalized 2-tuple $R(x, y) : - \ y = 4x$ in a natural sense.

A generalized database *is a finite set of generalized relations.*

The semantics of constraint query languages is also discussed in [34]. An alternative and equivalent semantics presented in [48] is as follows:

Let r_i be the generalized relation assigned to R_i. We associate with each r_i a formula F_{r_i} that is the disjunction of the formulas on the right side of each generalized k-tuple of r_i. Let ϕ be any relational calculus formula. Satisfaction with respect to a domain δ and database d, denoted $<\delta, d>\models$, is defined recursively as follows:

$$<\delta, d>\models R_i(a_1, \ldots, a_k) \quad \text{iff} \quad F_{r_i}(a_1, \ldots, a_k) \text{ is true} \tag{2.1}$$

$$<\delta, d>\models (\phi \wedge \psi) \quad \text{iff} \quad <\delta, d>\models \phi \text{ and } <\delta, d>\models \psi \tag{2.2}$$

$$<\delta, d>\models (\neg\phi) \quad \text{iff} \quad \text{not } <\delta, d>\models \phi \tag{2.3}$$

$$<\delta, d>\models (\exists x_i \phi) \quad \text{iff} \quad <\delta, d>\models \phi[x_i/a_j] \text{ for some } a_j \in \delta \tag{2.4}$$

$$<\delta, d>\models R_i(a_1, \ldots, a_k) \quad \text{iff} \quad (a_1, \ldots, a_k) \in r_i \tag{2.5}$$

$$<\delta, d>\models (\phi \wedge \psi) \quad \text{iff} \quad <\delta, d>\models \phi \text{ and } <\delta, d>\models \psi \tag{2.6}$$

$$<\delta, d>\models (\neg\phi) \quad \text{iff} \quad \text{not } <\delta, d>\models \phi \tag{2.7}$$

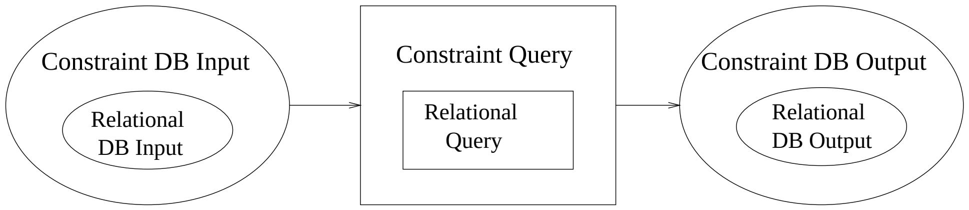

Figure 2.1: The Constraint Data Model

$$< \delta, d > \models (\exists x_i \phi) \quad \text{iff} \quad < \delta, d > \models \phi[x_i/a_j] \text{ for some } a_j \in \delta \tag{2.8}$$

where $[x_i/a_j]$ means the instantiation of the free variable x_i by a_j.

For the relational data model, queries are functions from input relational databases to output relational databases. In the generalized database model of [34], queries are functions from generalized databases to generalized databases using the same type of constraints. This closed-form requirement is the analogue to the termination and the constructibility requirements in relational databases. Constraint query languages are generalizations of relational query languages with constraints. Figure 2.1 illustrates the constraint data model. Constraint query languages are a subset of constraint logic programming [27]. Each constraint logic program is a mapping from a finite set of constraint facts to a least model. There are several criteria for effectively evaluating the least model or whether the least model is finitely representable and efficiently computable. For constraint query languages, these problems are solvable and thus satisfy the requirements that each query must terminate and the query output is given as a database.

It is known that for any first-order theory with constraint relations, $\mathcal{T}$, if quantifier elimination is allowed, then the output of each relational calculus query on generalized relations containing only constraints in $\mathcal{T}$ is evaluable in finite time. Moreover, the output can be represented as a generalized relation containing only constraints in $\mathcal{T}$. This is very helpful with the goal of closed-form evaluation for generalized databases. Computational efficiency for quantifier elimination can be improved by algebrizing the procedure. Algebraic operators provide efficient set-at-a-time computations with generalized relations, and they are considered in studies of constraint databases by many researchers (see some example works in [5, 29, 50]). We discuss the termination of the closed-form evaluation of Datalog queries with constraints in the next section.

2.2 Datalog with Constraints

Datalog [61], the primary example of a deductive query language, utilizes logic as a way to represent knowledge and as a language for expressing operations on relations. Datalog derives new relations from input relations using rules. Each Datalog query consists of a finite set of rules of the form

$$R_0(x_1, \ldots, x_k) : -R_1(x_{1,1} \ldots, x_{1,k_1}), \ldots, R_n(x_{n,1}, \ldots, x_{n,k_1})$$

The x's are either variables or constants in the domain. If the variables on the right side are substituted by constants, which makes the right side true, then the left side must also be true. In general, rules in Datalog define the true instances of certain predicates, with R_0 in the above form, in terms of certain other predicates,

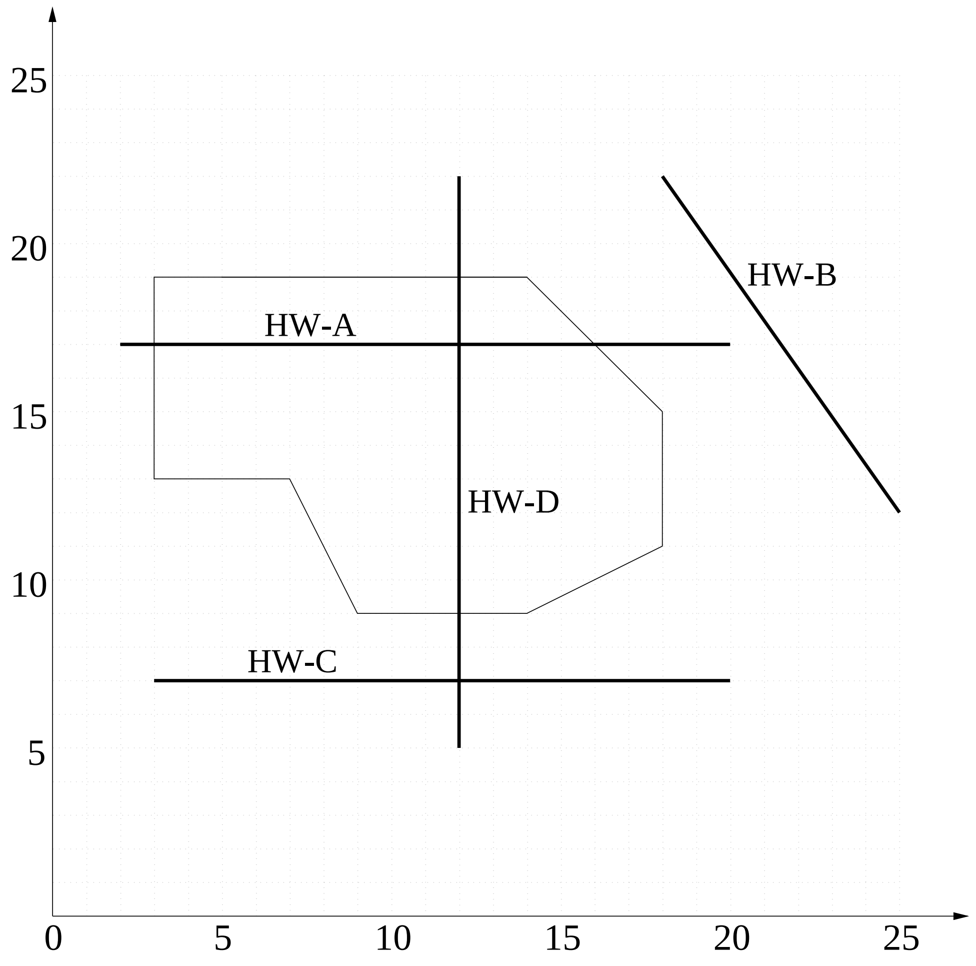

Figure 2.2: The Town Map and Highway Layout

$R_1, \ldots, R_n$, where each R_i is either an input relation name or a derived relation name.

Example 2.2.1 Suppose we have two relations in a constraint database. One relation represents a town region and the other contains the highways around the town as shown by Figure 2.2. The two relations can be defined as follows.

$$Town(x, y) : -x \geq 3, x \leq 18, y \geq 13,$$

$$y \leq 19, 7y - 6x \geq 49.$$

$Town(x, y) : -y \geq 9, 7y - 6x \leq 49, x \leq 18,$

$$y + 2x \geq 27, 2y - x \geq 4, x + y \leq 33.$$

$Highway(HW\_A, x, y) : -y = 17, x \geq 2, x \leq 20.$

$Highway(HW\_B, x, y) : -10x + y = 234, y \geq 12, y \leq 22.$

$Highway(HW\_C, x, y) : -y = 7, x \geq 3, x \leq 20.$

$Highway(HW\_D, x, y) : -x = 12, y \geq 5, y \leq 22.$

The $Town$ relation has two attributes which are represented as two variables, x and y corresponding to the value of x and y coordinates in the map. The town region is the collection of all the point-sets that satisfy the constraints on the right side of the rules. The relationship between relations with same relation names but different rule bodies, is the logical OR. All the (x, y) pairs that satisfy either of the two sets of rules are in relation $Town$. The $Highway$ relation contains four sets of constraints, each of which represents the point-set that lies on a highway which is a line segment in the map. The comma symbols (,)s represent conjunctions between constraints. 'HW_A', 'HW_B', 'HW_C', and 'HW_D' are constants as the name of each highway.

Example 2.2.2 The following query finds all the highways that pass through the town.

$Q_{passthrough}(name) : -Town(x, y), Highway(name, x, y).$

The query $Q_{connect}$ finds all the highways that are connected to the town directly or indirectly (if intersecting any passing through highway).

$Q_{connect}(name1) : -Q_{passthrough}(name1), Highway(name1, x, y).$

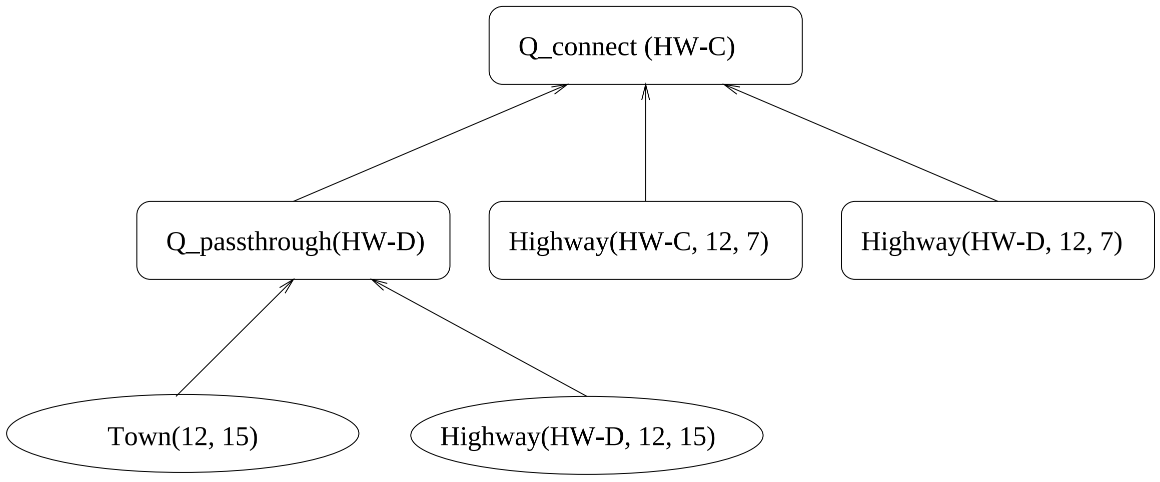

Figure 2.3: The Proof Tree

$Q_{connect}(name) : -Q_{connect}(name1), Highway(name1, x, y), Highway(name, x, y).$

A query is *recursive* is the body of any of its rules constraints a derived relation. Otherwise, the relation is called *non-recursive*. For instance, the query $Q_{passthrough}$ is a non-recursive query and $Q_{connect}$ is a recursive query.

Syntactically, Datalog is a fragment of predicate calculus extending relational calculus with intensionally defined relations. In relational calculus each query defines a single output relation which is not named explicitly and all other relations are input relations. In Datalog, a query may define several output relations which are referred to by names within the query, that is, built-in relations are allowed in Datalog. There are expressions in recursive Datalog , such as the $Q_{connect}$ query, that can not be expressed in relational algebra or relational calculus. The relational calculus underlying most commercial relational query languages is a form a logic that can be obtained from a non-recursive Datalog program with the substitution of logical OR of the rule bodies.

The proof-based semantics of Datalog queries views the input databases as a set of axioms and the rules of the query as a set of inference rules to prove that specific tuples are in some derived relation. The rules can be used only by substituting given or proven facts in the right side and thereby proving the resulting fact on the left. When proof trees are used to show the reasoning of provable tuples, each internal node usually represents the head of an instantiated rule, and its children are the body relations. For example, the proof tree displayed in Figure 2.3 proves that 'HW_C' indirectly connects to the town. It is a fact that the point $(12, 15)$ is within the town region and also on the highway 'HW_D'. So 'HW_D' is one of the proven facts in relation $Q_{passthrough}$. Another given fact is that the point $(12, 7)$, the intersecting point, resides on both highway 'HW_C' and highway 'HW_D'. Applying the rules, it can be derived from the $Q_{passthrough}$ relation and the $Highway$ relation that 'HW_C' is a tuple in the $Q_{connect}$ relation. Similarly, proof trees can be constructed for every tuple in the output database as long as negation is not involved.

Model-theoretic based semantics of Datalog views rules as defining possible worlds or "models". An interpretation assigns truth or falsehood to every possible instance of the predicates, whose arguments are chosen from some infinite domain of constants. Instead of only given facts used as in the proof-based interpretation, a model is an interpretation that makes the rules true, no matter what assignment of values from the domain is made for the variables in each rule. Consider the rules (1) $p(x, y) :- q(x, y)$ and (2) $q(x, y) :- r(x, y, z)$. Suppose the domain of interest is $Reals \times Reals$ and the only given fact is $r(1,2,3)$. Then the true tuples that can be derived using proof-tree are $q(1,2)$ and $p(1,2)$. No other tuple is true. However, not only $\{p(1,2), q(1,2)\}$ is a model, but also $\{r(1,2,3), p(1,2), q(1,2), p(3,4), q(3,4)\}$ which is consistent with the database because the semantics "If the right side is true, then the left side is true." is not violated.

The model-theoretic interpretation of rules can deal with negations in Datalog programs. To interpret negations, the use of a variable in a subgoal is forbidden if that variable does not also appear in another subgoal, which is neither negated nor a built-in predicate. It is possible to rewrite any rule so that this restriction can be satisfied. For example, suppose q and r are given relations, the rule $p(x) :- q(x), \neg\, r(x,y)$ can be rewritten as (1) $s(x) :- r(x,y)$ and (2) $p(x) :- q(x), \neg\, s(x)$.

It seems that the model-theoretic approach can handle a more powerful class of rules. Nevertheless, no matter which meaning to choose, it is essential that equivalent computational meaning can be found, that is, to provide an algorithm for executing the rules and then tell whether a potential fact is true or false. Fortunately, rules can be translated into a sequence of operations in relational algebra. Algorithms are available to compute all and only those tuples such that when variables are substituted by constants, every rule is made true. Recursive rules can be computed correctly using naive or semi-naive evaluation algorithms [61]. Negated rules can be stratified first and then the ? least model is selected from among all possible models.

Constraint databases are parameterized by the type of constraint domains and constraints used. Real polynomial inequality constraints, dense linear order inequality constraints, boolean equality constraints, and set constraints are typical types attracting a great deal of interest in the constraint database community. It is known that Datalog programs with real polynomial constraints can be evaluated bottom-up in closed-form and NC data complexity[34]. Since most basic operations of computational geometry can be described in Datalog with real polynomial constraints, this implies the potential capability of constraint databases to be used in spatial applications.

2.3 Spatial Databases with Linear Constraints

A spatial database is a database system that offers spatial data types and query languages and supports such data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

As a number of data models have been proposed to handle spatial data, there are many studies concerning the finite representation of the infinite and non-enumerable set of points of spatial objects that are described by spatial data models. For example, the raster model describes a spatial object by a finite number of points which are equally distributed following an easy geometric pattern, normally a square. The spaghetti model deduces an spatial object to a set of poly-lines from its contour. Based on the Peano curve, the Peano model represents non-uniformally distributed object-points. The topological model handles topological information without dealing with the exact position and form of the spatial objects. The polynomial model describes spatial properties of an object by *semi-algebraic sets* [32, 63].

It is generally required that a spatial database should contain an elegant framework to combine geometric and thematic information, be as general as possible and not be designed for one particular area of application, have a formally defined semantics that is closed under set theoretic, geometric and topological operations, i.e. defined in terms of finite representations and use efficient implementation techniques, especially for the operations on n-dimensional objects.

The polynomial data model is well suited to model spatial objects which require exact geometrical and geographical information. The task of the spatial database is to store a representation of some geographic area which are typically two-dimensional maps. Such geographical information can be described precisely. Higher dimensional spatial objects, unbounded and topologically non-closed geometric figures which most other models can not handle, can also be represented and manipulated with polynomial inequality constraints.

The linear data model approximates spatial objects using linear representable objects, e.g. points, line segment, polygons, i.e. only linear inequality constraints are allowed. The simplicity of linear data is very attractive to spatial data modeling and there also exist efficient algorithms to implement the variety of operations on spatial data [5, 6, 24, 38]. Spatial database models and prototypes [33] proposed in the literature often focus on one specific type of spatial information such as polygonal line segments for geographic application. This concentration is acceptable because it is sufficient for many situations and allows extensions. In this thesis linear constraint models are applied to our studies of similarity measure and database change operators. We discuss the linear data model more formally as follows. The main idea is equivalent to [32].

A *linear term* is of the form $\sum_{i=1}^{k} a_i x_i$, where $x_1, \ldots, x_n$ are variables over real numbers called *real variables* and $a_1, \ldots, a_n$ are rational constants. An *atomic linear formula* is of the form $t \odot c$ with t a linear term, c a real constant and $\odot \in \{=, <, >, \leq, \geq, \neq\}$. A *linear formula* is recursively defined in first-order logic with addition, i.e.,

(1) an atomic linear formula is a linear formula;

(2) if ϕ and ψ are linear formulae, then $\phi \vee \psi, \phi \wedge \psi, \neg\psi$ are linear formulae;

(3) if x is a real variable and ψ a linear formula in which x is free, the $(\exists x)\psi$ is a linear formula.

Every linear formula ψ with n free real variables, $x_1, \ldots, x_n$, defines a point-set called a *semi-linear set*

$$\{(x_1, \ldots, x_n) \mid \psi(x_1, \ldots, x_n)\}$$

in n-dimensional Euclidean space R^n. Every semi-linear set can be represented by an infinite number of linear formula, while every linear formula defines exactly one seme-linear set.

A *linear tuple* of type $[n, m]$ is defined as a tuple of the form

$$(c_1, \ldots, c_n, \psi(x_1, \ldots, x_m))$$

where $c_1, \ldots, c_n$ are non-spatial values of some domain C and $\psi(x_1, \ldots, x_m)$ is a linear formula with m free real variables. Its semantics is the possibly infinite subset of $\{(c_1, \ldots, c_n)\} \times R^m\}$, which can be interpreted as a possible infinite $(n+m)$-ary relation. A *linear relation* is a finite set of linear tuples of type $[n, m]$. A *linear spatial database* is a finite set of linear relations.

A linear calculus query language called L-calculus is defined based on the linear formulae as follows.

First, we allow non-spatial variable in a linear formula, disjoint with the set of real variables. The form $v_1 = v_2$ with v_1 and v_2 are non-spatial variables, is added to be an atomic formula. Universal and existential quantifications followed by non-spatial variables are atomic formulae. The form $R(v_1, \ldots, v_n, t_1, \ldots, t_m)$ is an atomic formula, where R is a spatial linear relation of type $[n, m]$, $v_1, \ldots, v_n$ are non-spatial variables and $t_1, \ldots, t_m$ are linear terms. Then a query expressed in L-calculus has the form

$$\{(x_1, \ldots, x_n) \mid \psi(x_1, \ldots, x_n)\}$$

where $\psi(x_1, \ldots, x_n$ is an expression of L-calculus with free variables $x_1, \ldots, x_n$.

Finally, we give a simple example query.

Example 2.3.1 The query "Find all the highways that pass through the town." on the constraint database in Example 2.2.1 can be expressed by the following L-calculus expression:

$$\{(n)|(\exists x)(\exists y)(Town(x,y) \wedge Highway(n,x,y))\}$$

For more detailed and deeper studies examining the expressiveness and limitations of calculus, see the references [32, 63].

2.4 Indexing for Constraint Databases

A successful practical database system could never ignore supporting its queries with efficient secondary storage manipulation. For the relational data model, there are many techniques developed, e.g., B-trees, grid files, Hashing indices, etc. All the indexing techniques proposed and applied in database systems have shown advantages as well as limitations, no one technique is best in general since each technique may be best suited for particular applications. Regardless of which technique to use, there are some basic factors to be evaluated for each technique [57].

- *Access types* that are supported efficiently. These types could be searching tuples with a specified attribute value, or if a value falls in a specified range.
- *Access time* it takes to accomplish a searching.
- *Insertion time* it takes to insert new information. The value included the time for finding the correct place to insert it and the time for updating the index structure.

- *Deletion time* it takes to delete old information. The value includes the time for finding the location of the old information and the time for updating index structure.

- *Space overhead* due to the additional space occupied by an index structure. There are always tradeoffs between the improved performance and space overhead.

B^+-trees [8, 7] are representive data structures for implementing relational databases. Let r be a relation with n tuples, each secondary memory access transit B units of data. Let x be the search key. The space used for a B^+-tree index is $O(N)$. The worst case for finding all tuples such that for the x attribute $(a_1 \leq x \leq a_2)$ is $O(log_B n + k/B)$ if the output size is k tuples. To insert or delete a given tuple, the worst case is $O(log_B N)$ secondary memory access.

For spatial databases, indexing is also a central problem. There are some solutions with good performance, e.g., R-trees, quad-trees, K-D-B-trees [47, 59, 60]. It is of great value to study how current spatial database access methods can be applied to indexing constraint query languages. Constraint query languages guarrentees low data complexity and have strong applicability to manipulate spatial data. It is, of course, important to index constraints and thus to support the new language with efficient secondary access as long as the cost of space is moderate.

Analogue to relational databases, there are two operations defined as follows for generalized databases, which is called *one-dimensional searching on generalized* *database attribute x.*

(1) Find a generalized relation that contains all the tuples of the input generalized database such that their x attribute satisfies $(a_1 \leq x \leq a_2)$;

(2) Insert or delete a given generalized tuple.

The problem of *k-dimensional searching on generalized database attributes $(x_1, \ldots, x_k)$* extends one-dimensional searching to k attributes with range searching on k-dimensional intervals.

Under the natural assumption that the projection of any generalized tuple on x is one interval $(a_1 \leq x \leq a_2)$, [34] gives a solution for one-dimensional searching. The idea is to index a generalized database using a set of intervals, where each interval is associated with a generalized tuple. Each interval $(a_1 \leq x \leq a_2)$ in the index is the projection on x attribute of its associated generalized tuple. The two-endpoint a_1, a_2 representation of an interval act as a fixed length generalized key. Finding a generalized relation that represents all the tuples of the input generalized database such that their x attribute satisfies $(a_1 \leq x \leq a_2)$, can be performed by adding the constraint $(a_1 \leq x \leq a_2)$ to only those generalized tuples whose generalized keys have a non-empty intersection with it. The insertion or deletion of a given generalized tuple is performed by computing its projection and insert or delete intervals from a set of intervals. This method transforms the one-dimensional searching into the problem of on-line intersections in a dynamic set of intervals and thus reduces redundancy of representation and improves performance.

The indexing is much harder for two or higher dimensional searching problems. Under the standard assumption that each secondary memory access transmits one page or B units of data and counts it as one I/O, [31] proposes the new data structure *metablock tree* to solve diagonal corner query. This data structure has worst-case space $O(n/B)$ pages, query I/O time $O(log_B n+k/B)$ and $O(log_B n+(log_B n)^2/B)$ amortized insert I/O time. [31] points out that the two-dimensional range searching can be solved using worst-case $O((n/B)log_2 n)$ pages, static query I/O time $O(log_2 n log_B n + k/B)$, and amortized update I/O time $O(log_2 n log_B n)$.

There are also significant other progress made in different directions. It is proved in [22] that two-dimensional queries can be answered in $O((n/B+t/B)(log_{M/B}(n/B)) + k/B)$ I/O's where t is the number of queries being processed and M is the amount of main memory available. [55] shows that it is possible to answer 2-sided queries in optimal $O(log_B n + k/n)$ disk I/O's with storage usage of $O((n/B)log_2 log_2 B)$. The amortized cost of an update is $O(log_B n)$. [54, 55] also consider range searching when several attributes are involved. A more general statement found in [58] shows that any secondary storage data structure that can answer two-dimensional range searching queries in $O(log_B^c n + k/B)$ I/O's in the worst-case (where c is a constant) has to occupy $\Omega((n/Blog(n/B)/(loglog_B n))$ disk blocks. All these techniques aim at efficient constraint queries. Many open problems remain to be solved in this area.

Chapter 3

Similarity Measure of Spatial Databases

There are several similarity notions including changes of scale, position, size, rotation, shape changes, and so on. These are typical geometric transformations. Similarity measure is application-dependent. Study on the changes of a single factor such as area difference, or Euclidean distance, is far from adequate to satisfy numerous applications. It is unlikely possible to find a general measure best for all cases. However, there are still advantages to look at these problems together in an abstract way, as we do in this chapter. We propose a new method to measure the similarities between two-dimensional spatial objects, which are used to represent most spatial scenes and can be easily extended to deal with n-dimensional spatial information.

3.1 Distance Measure

3.1.1 Measure Line Segments

Assume that we are given a linear constraint database which contains a set of line segments in the plane. Each line is defined by its slope, a lower bound and an upper bound constraint on x. An example of a general form could be, $y = ax+c, lb < x < up$ where a, and c are constants, lb and up are the upper and lower bounds respectively.

Given a specific line, any other line segment can be transformed into the given line segment by shifting its mid-point to where the given line segment's mid-point is located, then rotating the line segment a certain degree, and finally extending or shrinking the line segment to the length of the given line segment. Based on this observation, we measure the distance between two line segments by taking into consideration the changes of length, slope and position together, which can give complete comparison between two line segments. Each of the three factors may be of different importance in different applications, so weights are put on each of the three factors and can be assigned with any values flexibly.

First, we define following terms.

MidpnDist is the Euclidean distance of the mid-points of the two line segments.

LengthDiff is the absolute value of the difference of the lengths of the two line segments.

LengthSum is the sum of the lengths of the two line segments.

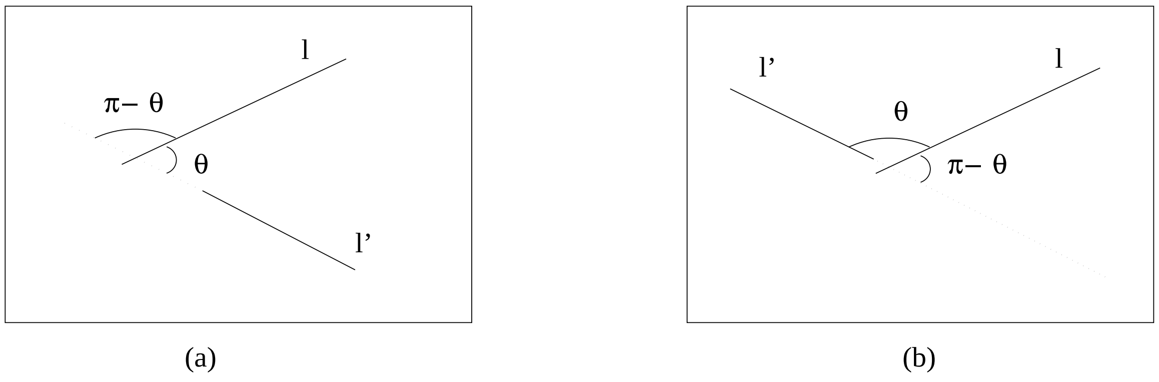

Figure 3.1: Definition of $AngleDiff$

AngleDiff is the rotation degree.

Suppose we have two arbitrary line segments l and l' in the plane, as shown in Figure 3.1(a). The two line segments are either parallel or their extended lines intersect at some point. There exist two inside angles θ and $\pi - \theta$ between these two line segments. The smaller one θ is defined as the $AngleDiff$. If two line segments are parallel, $\theta = 0$. Under this definition, the $AngleDiff$ ranges from 0 to 90. The value of $sin(AngleDiff)$ ranges from 0 to 1. Given a line segment l, all line segments on the line l' have the same $AngleDiff$ with respect to line segment l. As illustrated in Figure 3.1(b), $Anglediff = \pi - \theta$. The degree of $AngleDiff$ in (b) equals that in (a) because $\pi - \theta$ in (b) equals to θ in (a).

Now, we propose the following formula to calculate the distance notated $ldist(l_1, l_2)$ between two line segments l_1 and l_2.

$$ldist(l_1, l_2) = w_p \times MidpnDist + w_s \times LengthDiff + w_r \times LengthSum \times sin(AngleDiff) \tag{3.1}$$

where w_r, w_p, w_s are weights for the transformation of rotation, position and scale, respectively.

The following example shows how the proposed formula can be used to measure the distance between two line segments.

Example 3.1.1 Suppose we have two arbitrary line segments l_1 and l_2 in the plane as shown in Figure 3.2(1). The end points of the two line segments are $(17, 23), (6, 12)$ and $(23, 3), (19, 14)$, respectively. First, we calculate the coordinates of their mid-points m_1 and m_2. For m_1, it is $(11.5, 17.5)$ and for m_2, it is $(21, 8.5)$. So $MidpnDist$ equals to $\sqrt{(11.5 - 21)^2 + (17.5 - 8.5)^2}$ which is 13.09.

Second, we calculate the $LengthDiff$ and $LengthSum$. The length of l_1 is 15.56. The length of l_2 is 11.70. So $LengthDiff = 3.86$ and $LengthSum = 27.26$.

Thirdly, the $AngleDiff$ equals to θ and $sin(\theta) = sin(66°) = 0.91$.

Finally, we assume $w_r = w_s = w_p = 1$ for simplicity. Pluging in the calculation results of former three steps, we get the distance of the two line segments $ldist(l_1, l_2) = 13.09 + 3.86 + 27.26 \times 0.91 = 41.76$.

If we say l_2 is a transformation from l_1, then Figure 3.2 displays how l_1 can be restored from l_2 by the following three steps: parallelly move l_2 along the line between m_1 and m_2 as Figure 3.2(2), then rotate l_2 an angle of $66°$ as Figure 3.2(3) and finally extend its length $15.56/11.70 = 1.33$ times as Figure 3.2(4). Now we see that the two line segments are exactly the same.

The proposed distance measure between two line segments is based on geometric

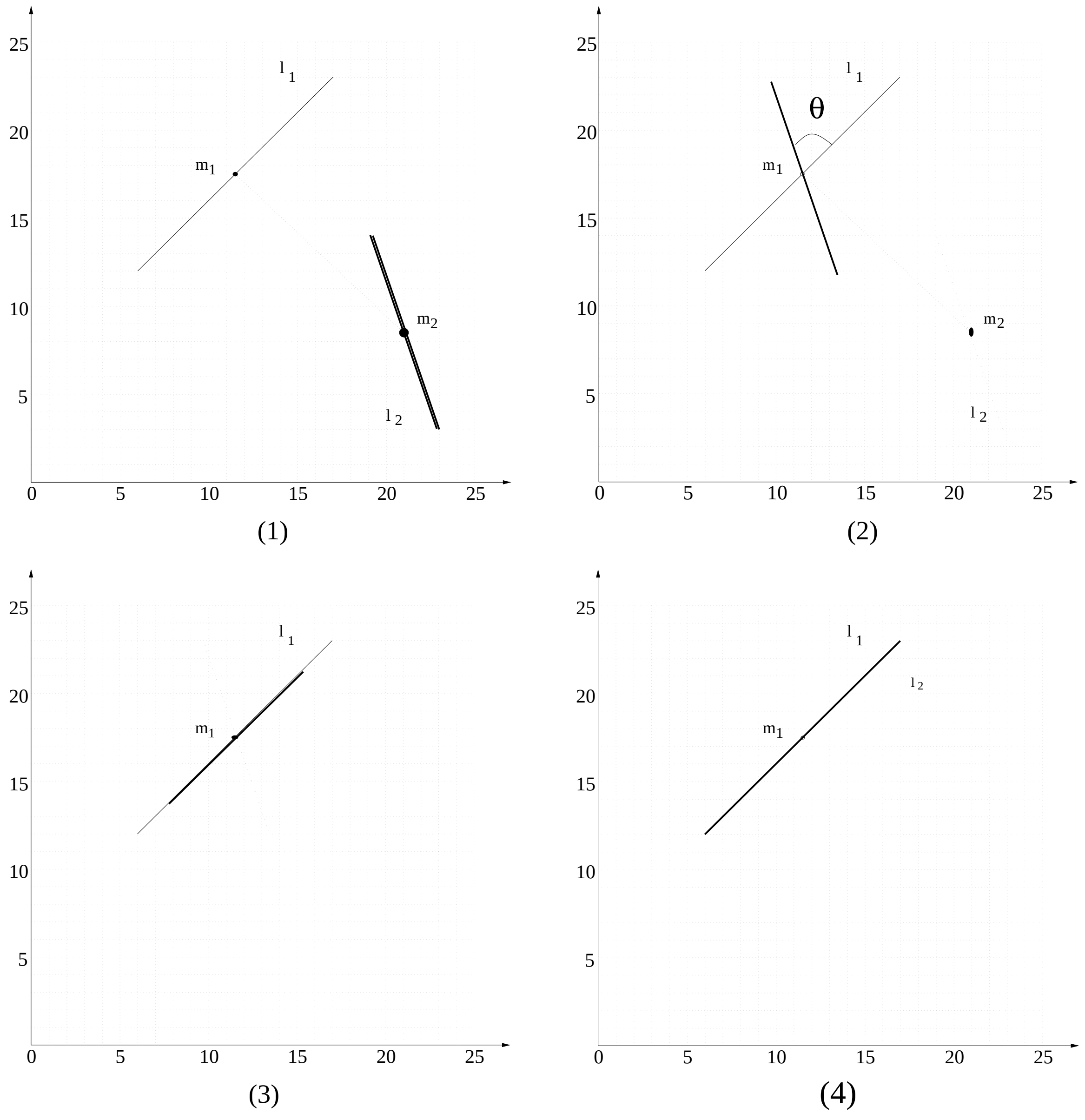

Figure 3.2: Distance between two line segments

transformations. It has following features.

1. *Geometric.* Only geometric features are considered. Non-spatial features of a spatial scene, such as weather information, population information are not involved. The advantage is that the comparison is quantified easily because geometric objects are usually represented in mathematical ways.

2. *Comprehensive.* All of the geometric transformation factors are considered. Two line segments have minimal distance 0 if and only if they are identical. Any translation, rotation, extension or shrink is reflected by a non-zero distance.

3. *Flexible.* Each transformation factor can be considered with different weight. For example, if rotation is considered to be a significant factor, then a much larger weight can be assigned to w_r. If some factor is of little interest, then a very small number or even 0 can be assigned to the corresponding weight coefficient.

4. *Atomic.* Spatial scene are usually represented as collections of points, line segments and polygons. If we let w_r and w_s be zero, then the formula calculates the geometric distance between two points. If we look a polygon as a set of line segments then the formula can be conveniently extended to applicable for polygons. A measure between two polygons is given in next section.

5. *Symmetric.* The functions applied on every factor are symmetric, so that no matter which one is the standard, the distance between two line segments is

always the same, i.e. $ldist(l_1, l_2) \equiv ldist(l_2, l_1)$.

6. *Non-oriented.* The line segments are considered with no direction, which leads to the simplicity of the measure.

7. *N-dimensional.* This formula can be used for line segments in n-dimensional space directly.

8. *Independent.* No matter what type of data model is used to represent the spatial objects, spaghetti model, linear constraint model, vector model or many others, the measure works independently on them.

9. *Efficient.* The formula can be calculated in constant time. Slope, length, mid-point distance involve only simple calculations well known by middle school students.

3.1.2 Measure Polygons

A two-dimensional spatial scene can be looked as a collection of polygons. This is a reasonable assumption because a line segment is a special polygon with one edge and zero area and a point is a special line segment with zero edge length. In addition, any non-linear (curved) objects can be approximated by linear objects. So a measure of distance between polygons can generally be a measure of any two-dimensional linear representable objects.

At first, we define following terms related to our measure.

commonArea is the area of the intersection of the two polygons.

totalArea is the area of the union of the two polygons.

Center point is the geometric center of a polygon.

DistCPnt is the Euclidean distance between the center points of the two polygons. $DistCPnt(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ with p_1 of (x_1, y_1) and p_2 of (x_2, y_2).

edgeLenSum is the average length sum of the two polygons, i.e. $edgeLenSum = \frac{1}{2}\sum l_{e_i}$ where l_{e_i} is the length of edge e_i which belongs to the two polygons.

scaleChange is calculated by: $dtEdgeLenSum/ptEdgeLenSum - 1$ or $ptEdgeLenSum/dtEdgeLenSum - 1$, depending on which one is positive. $dtEdgeLenSum$ denotes the length sum of the edges of one polygon which is called the distorted polygon and $ptEdgeLenSum$ denotes the length sum of the edges of the other polygon which is called the prototype polygon.

rttAngle is the average rotated angle. Each edge in the prototype polygon is paired with an edge in the distorted polygon. If the two polygons have different number of edges, then some edges are repeated used in pairing edges until every edge has a partner. The minimal average $AngleDiff$ of each pair of edges is defined as the $rttAngle$, where the $AngleDiff$ is the same as it is defined in previous section.

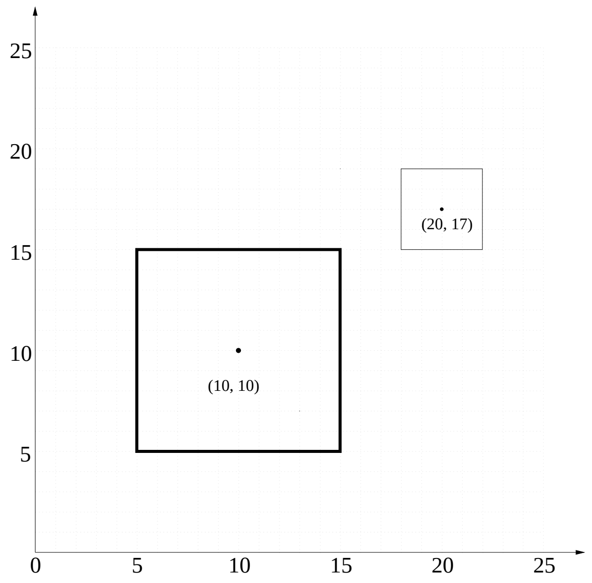

Figure 3.3: Distance between Two Squares

Now, we propose following formula to calculate the distance between two polygons P_1 and P_2,, denoted by $PGDist(P_1, P_2)$.

$$PGDist(P_1, P_2) = DistCPnt + \alpha * sin(rttAngle) * edgeLenSum +$$
$$\beta * scaleChange * edgeLenSum + \gamma * \sqrt{totalArea - commonArea} \qquad (3.2)$$

where α, β, γ may be any positive constants as weights for each component.

To show how we can use the proposed formula, we give following examples, from regular polygons to irregular ones and from simple situations to complicated ones. We assume $\alpha = \beta = \gamma = 1$ in all following examples.

Example 3.1.2 As an simple example shown in Figure 3.3, the two squares have the $DistCPnt$ equals to $\sqrt{(20-10)^2 + (17-10)^2}$ which equals to 12.2. There is no rotation between these two polygons, but some scale change. The $scaleChange$ equals

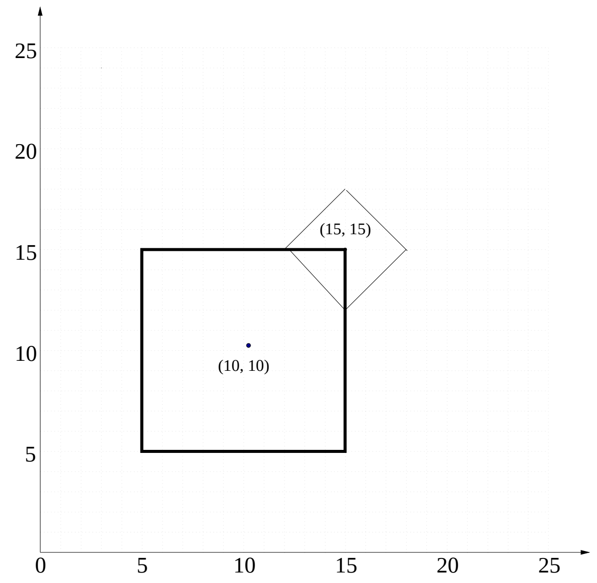

Figure 3.4: Distance between a Square and a Diamond

to $(40/16 - 1)$ which is 1.5. The average sum of edge lengths, $edgeLenSum$ equals to $(40 + 16)/2 = 28$. The $totalArea$ is $100 + 16 = 116$, with zero $commonArea$. So we have $PGDist = 12.2 + 1.5 * 28 + \sqrt{116} = \mathbf{64.97}$.

Example 3.1.3 As shown in Figure 3.4, the distance between two center points, $DistCPnt$ equals to $\sqrt{(15-10)^2 + (15-10)^2}$ which is 7.07. Each edge of the diamond can be paired with an edge of the square and the average rotation is $\frac{\pi}{4}$. The $scaleChange$ equals to $(40/12\sqrt{2} - 1)$ which is 1.38. The $edgeLenSum$ equals to $(40 + 12\sqrt{2})/2$ which is 28.48. These two polygons have the intersection area, $commonArea$ equal to 4.5 and the $totalArea$ equal to 113.5. So we have $PGDist = 7.07 + 1.38 * 28.48 + \frac{\sqrt{2}}{2} * 28.48 + \sqrt{109} = \mathbf{76.95}$.

Example 3.1.4 As shown in Figure 3.5, The $DistCPnt$ is 7.07. The $scaleChange$

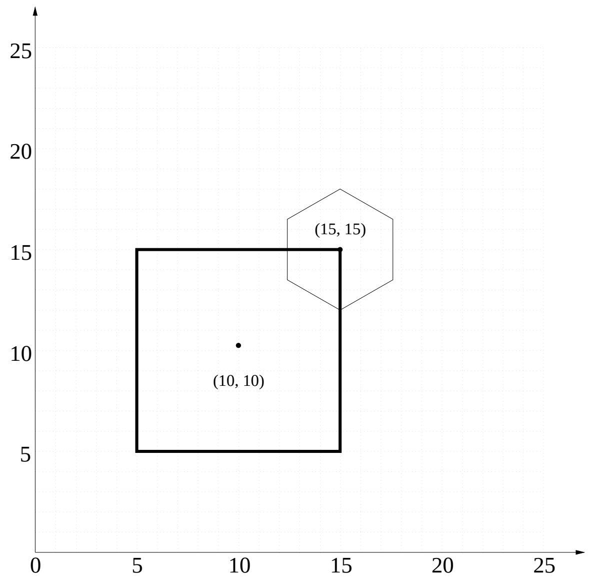

Figure 3.5: Distance between a Square and a Hexagon

equals to $40/18 - 1 = 1.22$. Each edge of the hexagon is paired to an edge of the square with rotation of 0, $\frac{\pi}{6}$, $\frac{\pi}{6}$, 0, $\frac{\pi}{6}$ and $\frac{\pi}{6}$, respectively. So the *rttAngle* equals to $\frac{\pi}{9}$. The *edgeLenSum* equals to $(40+18)/2 = 29$. The *commonArea* equal to 9.0 and the *totalArea* equals to 126.9. So we have $PGDist = 7.07 + sin(\frac{\pi}{9}) * 29 + 1.22 * 29 + \sqrt{117.9} = \mathbf{69.24}$.

Example 3.1.5 As shown in Figure 3.6, the center points of the two polygons are overlapped, so $DistCPnt$ equals to 0. The *scaleChange* equals to $42/40 - 1 = 0.05$. The *rttAngle* is $\frac{\pi}{4}$, the same as that of Example 3.1.4. The *edgeLenSum* equals to 41. The *commonArea* is $(100 - 4 * \frac{\sqrt{3}}{2})$ which equals to 96.5, and the *totalArea* equals to 129.5. So we have $PGDist = 0.05 * 41 + sin(\frac{\pi}{9}) * 41 + \sqrt{129.5 - 96.5} = \mathbf{28.46}$.

All the four examples we showed above are between two polygons. It is interesting

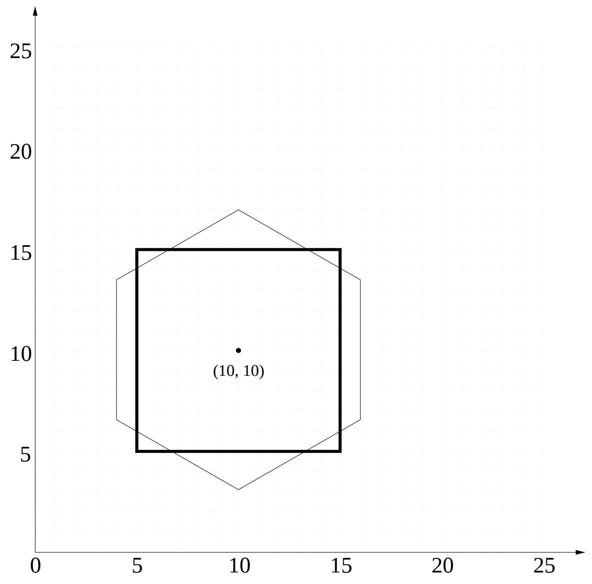

Figure 3.6: Distance between Two Polygons

to compare this measure and the measure of line segments with human intuitions of similarity evaluation. We will present this work in Chapter 6.

3.2 Matching Method for Similarity Measure

In this section, we introduce the problem of weighted bipartite matching and refer to some algorithms which solve this problem in polynomial time. Based on the distance measure presented in above section, we propose a new similarity measure for spatial databases by applying the ideas of the bipartite matching.

3.2.1 Weighted Bipartite Matching

A *matching* M in a graph $G = (V, E)$ is a set of edges with no two edges sharing a vertex. Given a graph $G = (V, E)$, the *matching problem* is to find a matching M that has as many edges as possible, i.e. the maximum matching of G. When the cardinality of a matching is $\lfloor |V|/2 \rfloor$, the largest one possible in a graph with $|V|$ nodes, the matching is *complete*.

If we are given, besides the graph $G = (V, E)$, a number $w_{ij} \geq 0$ for each edge $[v_i, v_j] \in E$, called the *weight* of $[v_i, v_j]$, then the matching problem turns to be finding a matching of G with the largest possible sum of the weights. Clearly, the unweighted matching problem is a special case of the weighted matching problem: Just let $w_{ij} = 1$ for all edges in E.

A graph is *bipartite* if its vertex set V can be partitioned into (at most) two independent sets V_1 and V_2, which means that there are no edges formed by two vertices which are in the same set. A *complete bipartite graph* is a bipartite graph in which the edge set consists of all pairs having a vertex from each of the two independent sets in the vertex partition. It is clear that bipartite graphs is a special case in the problem class of weighted matching.

We may assume that a bipartite graph G is a complete bipartite graph with two sets of vertices that are of equal size — otherwise let the weights of those missing edges in G be equal to zero or add new vertices with edges of weight zero incident upon them. The optimal solutions at this point will always be complete matchings (since

$w_{ij} \geq 0$. Considering the *cost* $c_{ij} = W - w_{ij}$ where W is larger than all the w_{ij}'s, the bipartite weighted problem is also known as the *assignment problem* which is stated as: For what man-job assignment is the total cost minimized, assuming the cost of assigning man i to job j is a_{ij}. The assignment problem can be described by a concise linear program, which is a special case of the Hitchcock problem [39], and is solved by the primal-dual method called the Hungarian Method [30, 39]. The Hungarian method solves the weighted matching problem for a complete bipartite graph with $2|V|$ vertices in $O(|V|^3)$ arithmetic operations. The matching problem has attracted the interest of researchers during decades of years. Some recent improvements can be found in [18, 46].

3.2.2 Similarity Measure

The distance measures proposed in Section 3.1 are between two line segments or two polygons. Now we assume that the two spatial objects are represented by two sets of polygons. For each polygon in one set, we calculate the distances between this polygon and every polygon in the other set using the proposed distance measure of two polygons.

Now, there are two different cases. In the first case we have the same number of polygons in the two pictures. In this case, we pair the polygons in the first picture with the polygons in the second picture such that the sum of the differences of the pairs is minimal. Such a pairing can be found in polynomial time in the number of polygons using weighted bipartite matching algorithms referred above.

If the number of polygons are different (say n and m number of polygons in the two spatial objects with $n < m$), then we pair the n elements of the smaller set of polygons with the closest elements of the other polygon. For each remaining unmatched polygon in the larger set ($m - n$ polygons) we assume some maximum constant C distance in the calculations.

Now let us look at an example which calculates the distance between two sets of polygons.

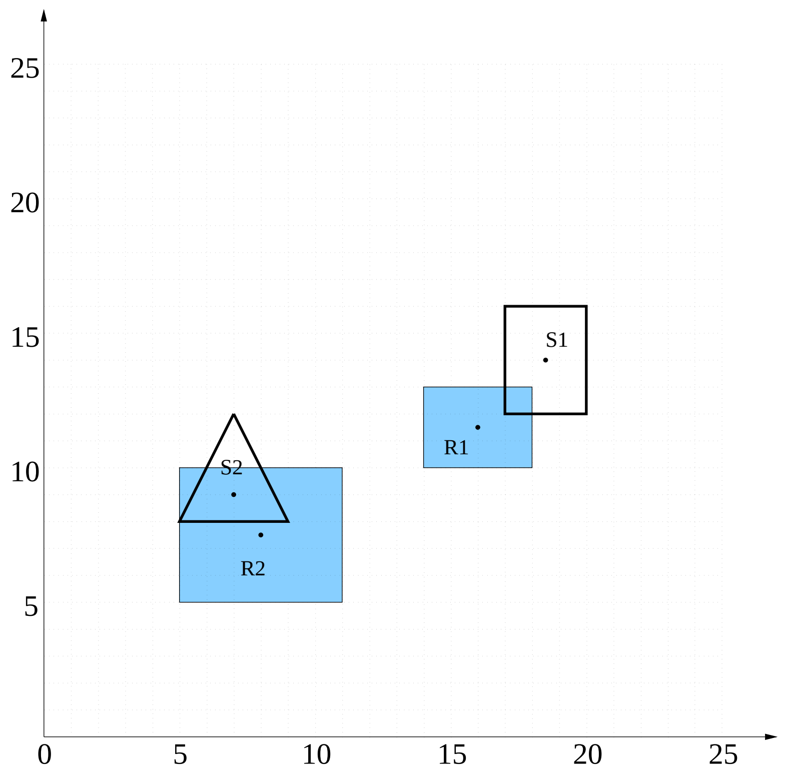

Figure 3.7: Distance between Two Sets of Polygons

Example 3.2.1 Suppose we have two databases S and R. Each contains two polygons as displayed in Figure 3.7. Using, Formula (3.2), we calculated the $PGDist$ between each pair of the polygons as below.

$PGDist(R1, S1) = 8.0$

$PGDist(R1, S2) = 19.52$

$PGDist(R2, S1) = 29.12$

$PGDist(R2, S2) = 25.10$

So the distance between database S and database R is $PGDist(R1, S1) + PGDist(R2, S2) = 8.0 + 25.10 = 33.10$, which is the minimal sum of the $PGDist$ across all possible polygon pairing.

Chapter 4

Change Operators for Constraint Databases

4.1 Introduction

Database systems must allow changes to the database. In most current database systems, specific commands are provided to insert and delete tuples or to update the values of the attributes. These low-level operations however often lead to inconsistencies and other errors. For example, if only additional information is to be acquired, one may simply insert a tuple into the database. If, however, the new tuple conflicts with the current contents of the database, we have to resolve the conflicts based on some rules and principles. This consistency requirement brings out the well-known problem of database changes: if some data is expected to change, then what other data must change with them and what should remain the same? The goal of high-level operations is to allow database users to be concerned only with what is the new information they want to insert into the database and to leave the burdon of resolving inconsistencies between the new and old data to the system.

There are three types of high-level change operations: *revision*, *update* and *arbitration*. Revision changes the database by saying that something in the database is incorrect in the sense that it was never true and a new information is to be added to replace the old one. For example, suppose there is a database containing information about the class registration of students in the University of Nebraska. A student wants to add class CSCE913 for the next semester. However, the database records show that he has registered CSCE861 which is scheduled at the same time as CSCE913. This is conflict to the constraint that students may not take two classes arranged at the same time. Assume that the latest registration is considered as the correct one and the former registration due to incorrect operation. The database systems revises the class registration for this student by triggering a series of low-level operations (i. e., delete the registration record of CSCE861 and insert the new registration of CSCE913) which leave the integrity constraint satisfied.

Update deals with the situation when some facts have changed over time, i.e., what was true before, is not true any more. For example, the student registered CSCE913 in the above example gets an ‘I’ grade by the end of the semester. Later a grade ‘A’ is to replace the old grade. Not only the low-level update operation is needed to set the value of grade attribute from ‘I’ to ‘A’, but also the change of the total credit hours earned by the student is required because an ‘I’ grade earns zero credit, while an ‘A’ grade earns 3 credit hours. Also the grade point average has to be changed as well. So the relational database is to be updated to keep the grades information up-to-date by applying a number of lower-level operations that satisfy the request of the user and leave all the usual database integrity constraints satisfied.

Arbitration settles arguments between sets of various statements. It is useful when making judgments in a trial, negotiating an international trade contract etc. For example, imagine that there is a database containing testimonies of several witnesses. The police wants to find all the consistent statements from the database. An arbitration operation over the set of testimonies draws out the results by applying a series of deletion of inconsistent information and combining the remaining statements into some consistent assertions. The three types of high-level change operations, revision, update and arbitration complement with each other. They can be used alternatively in complex situations as well as independently in specific cases.

A *declarative language* is a language in which one can express what one wants without explaining how the desired result is to be computed. *Knowledge-base systems* are generally systems supporting declarative, logic-based languages. A *knowledge-base management system* (KBMS) is a programming system that has the capabilities of both a database management system (DBMS) and a knowledge system [61]. High-level change operations over knowledge-bases preserve the property that users are not involved in how to maintain the consistency of the system but only describe the problem using provided declarative languages. The system automatically check the inconsistencies and knows how to solve them according to some rules.

Frameworks on generalizing the three types of change operations have attracted a number of researchers. Fundamental work on knowledge base revision can be found in [1] which proposes a set of rationality postulates that the revision and update operator must satisfy and it explores the implications of their postulates. Katsuno and Mendelzon [35] studies the revision and update operators for on propositional knowledge-bases. They discuss the differences between operators where the result of a change depends on the particular syntax of the sentences in the knowledge-base, and operators where it depends on the possible worlds described by sentences.

Katsuno and Mendelzon also give a model-theoretic characterization of all revision operators that satisfy the postulates in [1]. They also show that such operators accomplish a minimal change to the sets of models of the knowledge base. A total pre-order among the interpretations induced by the propositional knowledge-base is defined to give a meaningful sense to the concept of minimality.

Revesz [51] analyzes the model-theoretic properties of arbitration operators. A set of axioms is defined that arbitration operators must satisfy to accomplish a minimal change in a different sense. These three classes of axioms and minimal change principles are commonly accepted by most researchers. They relate the postulates with their methods of change operators, some of which are declared to satisfy all the axioms, while others satisfy part and fail on the other parts [3, 9, 15, 52, 56, 64, 65].

Each new information described in a logical form allows several models of the world which are currently thought as being true. Under the principle of minimal change, the result of changing any information in a knowledge-base should be the set of models that are closest to some existing models. Therefore, finding a distance measure between models is important for the precise definition and implementation of database change operations.

In this chapter, we address the issues of database change operators for linear constraint databases. By applying the distance measure proposed in the previous chapter, we show that, revision, update and arbitration operators for constraint databases can be defined and proved to satisfy the axioms proposed by Katsuno, Mendelzon and Revesz. All the operators also have a model-theoretic characterization that imply well-defined model-based minimal changes to constraint databases. Some examples are presented to show how the operators are applicable for spatial data manipulation.

The rest of this chapter is organized as follows. Section 4.2 presents the basic definitions and notations in constraint databases and model-fitting change operators. The fundamental axioms proposed in [35] and [51] are described. Next three sections present definitions of the three types of change operators. They are shown to satisfy all the recognized axioms. We also analyze their model-theoretic characteristics and give examples for various applications. Section 4.6 summarizes our contributions and mentions open problems.

4.2 Preliminaries

Alchourron and his colleagues [1] propose a set of postulates that any reasonable revision functions must satisfy. The postulates do not assume any specific languages to model the knowledge-bases but a deductively closed set of formulas. Katsuno and Mendelzon [35] consider knowledge-bases represented by a finite set of propositional sentences and proposes a set of postulates that a revision operator on knowledge-bases must satisfy. A direct correspondence between the two sets of postulates is also given and proved in [35].

Given a knowledge-base ψ and a sentence μ. Let $\psi \circ \mu$ denotes the revision of ψ by μ. The postulates of [35] are as following.

(KM1) $\psi \circ \mu$ implies μ.

(KM2) If $\psi \wedge \mu$ is satisfiable, then $\psi \circ \mu \equiv \psi \wedge \mu$.

(KM3) If μ is satisfiable, then $\psi \circ \mu$ is also satisfiable.

(KM4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$, then $\psi_1 \circ \mu_1 \equiv \psi_2 \circ \mu_2$.

(KM5) $(\psi \circ \mu) \wedge \phi$ implies $\psi \circ (\mu \wedge \phi)$.

(KM6) If $(\psi \circ \mu) \wedge \phi$ is satisfiable, then $\psi \circ (\mu \wedge \phi)$ implies $(\psi \circ \mu) \wedge \phi$.

The main meaning of the six postulates is that the new knowledge μ should be retained in the updated knowledge-base (KM1), when there is no conflict, it is guaranteed to take the obvious path(KM2), the revision introduces no unwarranted inconsistency (KM3), the principle of irrelevance of syntax [9] is satisfied (KM4), and (KM5) together with (KM6) represent the condition that revision should be accomplished with minimal change.

[35] gives a model-theoretic characterization of revision which satisfies the postulates. A function that assigns to each propositional formula ψ a total pre-order $\leq_\psi$ is a $faithfulassignment$ if the following conditions hold:

(1) If $I, I' \in Mod(\psi)$, then $I <_\psi J$ does not hold.

(2) If $I \in Mod(\psi)$ and $I' \notin Mod(\psi)$ then $I <_\psi I'$.

(3) If $\psi \equiv \phi$ then $\leq_\psi \equiv \leq_\phi$.

That is, a model of ψ can not be strictly less than any other model of ψ and must be strictly less than any non-model of ψ.

It is proved in [35] that a revision operator $\circ$ satisfies (KM1) - (KM6) if and only if there exists a faithful assignment that maps each knowledge-base ψ to a total pre-order $\leq_\psi$ such that $Mod(\psi \circ \mu) = Min(Mod(\mu), \leq_\psi)$. (see [35] for precise definitions of the notations appear above.)

[51] gives a formal definition of the set of model-fitting operations and also a model-theoretic characterization of the operations. [51] generalizes revision to model-fitting by allowing the knowledge-base to be inconsistent. Corresponding to that, the *loyal assignment* in [51] is a generalization of faithful assignment in [35].

Arbitration is originally proposed by [51] to be used as a change operator for knowledge-bases. Analogous to revision and update, [51] gives a set of axioms that an arbitration operator must satisfy and characterizes arbitration using *generalized loyal assignment.* The main idea behind the model-theoretic characterizations of a change operator is that the models closest to the whole set of formulas of a knowledge-base are selected based on some total pre-order over all the models of the possible world.

The studies on change operators in [35] and [51] are applicable to constraint databases. The operators satisfying relevant axioms also have a model-theoretic characterization. Before we show these, we give following basic definitions and notations used in following sections.

We adopt the definitions of constraint databases from [34].

Recall that a $generalized k-tuple$ over variables $x_1, \ldots, x_k$ is of the form:

$$r(x_1, \ldots, x_k) : - \ \phi_1 \wedge \ldots \wedge \phi_n$$

where r is a relation symbol, ϕ_i (for all $1 \leq i \leq n$) is an atomic constraints of some constraint theory and $x_1, \ldots, x_k$ are the only variables.

A $generalized relation r with arity k$ is a set of generalized k-tuples with symbol r on left side.

A $generalized database$ is a finite set of generalized relations.

In [35], a knowledge-base is a set of propositional formulas and the models of the formulas are interpretations that make the formulas true. Contrast to that, we define that a $generalized knowledge-base$ is a finite set of generalized databases. In stead of defining and comparing distance between interpretations, our task is to define distance between pairs of generalized databases. We restrict our generalized database to be a *linear constraint database* which contains a set of finite number of polygons in the plane. Each polygon is represented by a generalized relation, i.e., we assume that all spatial objects are represented by sets of polygons. This a reasonable assumption because a line segment is a special polygon with only one edge and zero area, and a point is a special line segment with zero edge length. Also, spatial objects with non-linear (curved) edges can be approximated by high-degree polygons.

We denote the models of A by $Mod(A)$ where A is a generalized relation, a generalized database or a generalized knowledge-base. The formal definitions are as following.

Let D be the domain over which variables in the database are interpreted.

Then the *model of a generalized k-tuple* with variables $x_1, \ldots, x_k$ is the k-ary relation

$$\{(a_1, \ldots, a_k) \,:\, (a_1, \ldots, a_k) \in D^k \; and\, the\, substitution\, of\, a_i\, for\, x_i\, satisfies\, the\, right\, side.\}$$

The *model of a generalized relation* is the union of the models of its generalized tuples.

The *model of a generalized database* is the set of the models of its generalized relations.

The *model of a generalized knowledge-base* is the set of the models of its generalized databases.

Let K_1 and K_2 be two knowledge-bases.

If μ is a generalized database, we take $\mu \in K_1$ to be true if and only if $Mod(\mu) \in Mod(K_1)$ in the regular sense.

$Mod(K_1) = \{\{Mod(\mu)\} : \mu \in K_1\}$.

$K_1 \subseteq K_2$ is true if and only if $Mod(K_1) \subseteq Mod(K_2)$ in the regular sense.

$K_1 \cap K_2 = \{\mu_1 \in K_1 : \exists \mu_2 \in K_2$ such that $Mod(\mu_1) = Mod(\mu_2)\}$. Note that $Mod(K_1 \cap K_2) = Mod(K_1) \cap Mod(K_2)$.

A *pre-order* relation $\leq$ over M is a reflexive and transitive relation.

A pre-order is *total* if for all pairs $I, J \in M$, either $I \leq J$, or $J \leq I$ holds.

$I < J$ holds if and only if $I, J \in M$, and $I \not\leq J$.

The set of *minimal elements*, S of M with respect to the pre-order $\leq_\psi$ is defined as:

$$Min(S, \leq_\psi) = \{I \in S : \nexists I' \in S \; where \; I' <_\psi I\}$$

The *distance between two generalized relation* is calculated using the proposed formula in Section 3.1.2 since each relation represents a polygon. We pair each polygon of one database with a polygon of the other database such that the sum of *PGDist* of all the polygon pairs is minimal. This minimal sum is defined as the *distance of the two generalized databases*, denoted by $DBdist(DB_1, DB_2)$. We define the distance between a knowledge-base K and a generalized database I as following:

$$Dist(K, I) = \min_{J \in K} \; DBdist(I, J)$$

We define the *overall distance* between a knowledge-base K and a generalized database I as following:

$$Odist(K, I) = \max_{J \in K} \; DBdist(I, J)$$

4.3 Revision

[52] proposes a set of axioms out of the postulates in [1] that a revision operator on generalized knowledge-bases must satisfy, which says that for generalized knowledge-bases ψ, ϕ and μ, the following holds:

(R1) $Mod(\psi \circ \mu) \subseteq Mod(\mu)$.

(R2) If $\psi \cap \mu$ is non-empty, then $Mod(\psi \circ \mu) = Mod(\psi \cap \mu)$.

(R3) If μ is non-empty, then $\psi \circ \mu$ is non-empty.

(R4) If $Mod(\psi_1) = Mod(\psi_2)$ and $Mod(\mu_1) = Mod(\mu_2)$, then $Mod(\psi_1 \circ \mu_1) = Mod(\psi_2 \circ \mu_2)$.

(R5) $Mod((\psi \circ \mu) \cap \phi) \subseteq Mod(\psi \circ (\mu \cap \phi))$.

(R6) If $Mod((\psi \circ \mu) \cap \phi)$ is non-empty, then $Mod(\psi \circ (\mu \cap \phi)) \subseteq Mod((\psi \circ \mu) \cap \phi)$.

For generalized knowledge-bases, a *faithful* assignment is defined as a function that assigns for each knowledge-base ψ a total pre-order $\leq_\psi$ such that the following conditions hold:

(1) If $Mod(I), Mod(J) \in Mod(\psi)$ then $I <_\psi J$ does not hold.

(2) If $Mod(I) \in Mod(\psi)$ and $Mod(J) \notin Mod(\psi)$ then $I <_\psi J$.

(3) If $Mod(\psi_1) = Mod(\psi_2)$ then $\leq \psi_1 = \leq \psi_2$.

There exists a direct relationship between the axioms and a faithful assignment which gives the model-theoretic characterization for revising generalized databases.

Theorem 4.3.1 *A revision operator satisfies axioms (R1) - (R6) if and only if there exists a faithful assignment that maps each generalized knowledge-base ψ to a total* *pre-order* $\leq_\psi$ *such that for every other generalized knowledge-base* μ, $Mod(\psi \circ \mu) = Mod(Min(\mu, \leq_\psi))$.

Now we define a concrete revision operator as follows.

For each pair of generalized databases I and J, we define $I \leq_K J$ holds if and only if $Dist(K, I) \leq Dist(K, J)$ where K is a generalized knowledge-base. Then the revision operator $\circ$ on knowledge-bases K and ψ is defined as:

$$K \circ \psi = Min(\psi, \leq_K)$$

Lemma 4.3.2 *The revision operator* $\circ$ *satisfies* $(R1) - (R6)$.

Proof Prove by showing that the three faithful assignment conditions are satisfied by the revision operator $\circ$.

For condition (1), if $Mod(I), Mod(J) \in Mod(\psi)$, then $Dist(\psi, I) = 0$ and $Dist(\psi, J) = 0$, i.e., $Dist(\psi, I) = Dist(\psi, J)$. So $I <_\psi J$ does not hold.

If $Mod(I) \in Mod(\psi)$ and $Mod(J) \notin Mod(\psi)$, then $Dist(\psi, I) = 0$ and $Dist(\psi, J) > 0$. So we have $I <_\psi J$. Condition (2) is satisfied.

It is obvious that condition (3) is also satisfied.

Hence, we have defined a *faithful assignment* that maps each knowledge-base K to a total pre-order $\leq_K$ such that for every other knowledge-base ψ, $Mod(K \circ \psi) = Mod(Min(\psi, \leq_K))$. ∎

The theorem and proof can be explained in a regular sense as follows. Suppose we have a set of pictures containing spatial objects. Each picture is represented by a generalized database. Relations in a generalized database represent the polygons which consist the picture. Given a new picture, the revision operation selects those pictures that have minimal distance with respect to the given picture, i.e., the pictures that are most similar to the given picture are the result of the revision operation. If any picture is the same as the given one, then this picture has the minimal distance zero, while other pictures have a positive distance.This property guarrentees a faithful assignment.

Example 4.3.1 Suppose there are a set of pictures each of which represents the shape of a house owned by a real estate agent, as shown in Figure 4.1(1)-(3). A customer who is interested in buying a house from the agent provides a picture which shows the kind of shape that he is favorite, as shown in Figure 4.1(4). Then which house is better for the agent to exhibit to the customer?

We represent the agent's pictures by generalized knowledge-base ψ and the customer's picture by knowledge-base K. Now applying $K \circ \psi$, it is easy to calculate and find out that picture (2) is closest to picture (4). So the agent will choose the one that most possibly satisfiable to his customer's interest. It is not hard to expect the success of the agent who uses the databases that provide such kind of similar queries and revision operations.

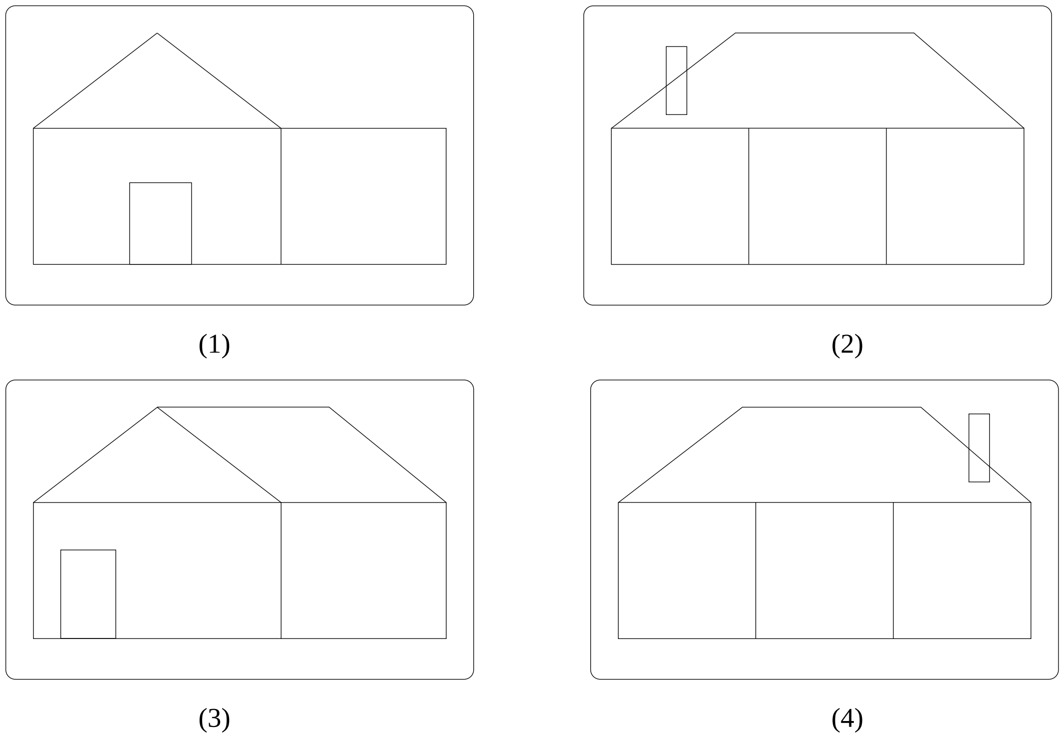

Figure 4.1: Revision Operation

4.4 Update

When information previously believed true becomes obsolete and has to be given up, new information is to be added. The operation makes this change is called an update. The following axioms must be satisfied for an update operation on generalized knowledge-bases ψ, μ and generalized database I [52].

(U1) $Mod(\psi \diamond \mu) \subseteq Mod(\mu)$.

(U2) If $Mod(\psi) \subseteq Mod(\mu)$, then $Mod(\psi \diamond \mu) = Mod(\psi)$.

(U3) If ψ and μ are non-empty, then $\psi \diamond \mu$ is non-empty.

(U4) If $Mod(\psi_1) = Mod(\psi_2)$ and $Mod(\mu_1) = Mod(\mu_2)$, then $Mod(\psi_1 \diamond \mu_1) =$

$Mod(\psi_2 \diamond \mu_2)$.

(U5) $Mod((\psi \diamond \mu) \cap I) \subseteq Mod(\psi \diamond (\mu \cap I))$.

(U6) If $Mod(\psi \diamond \mu_1) \subseteq mod(\mu_2)$ and $Mod(\psi \diamond \mu_2) \subseteq mod(\mu_1)$, then $Mod(\psi \diamond \mu_1) = Mod(\psi \diamond \mu_2)$.

(U7) $(Mod((I \diamond \mu_1) \cap (I \diamond \mu_2))) \subseteq Mod(I \diamond (\mu_1 \cup \mu_2))$.

(U8) $Mod((\psi_1 \cup \psi_2) \diamond \mu) = Mod((\psi_1 \diamond \mu) \cup (\psi_2 \diamond \mu))$.

A *faithful assignment* for update satisfies the following condition: For any generalized database I, if $I \neq J$ then $I <_I J$.

The direct relationship between the axioms and a faithful assignment which gives the model-theoretic characterization for updating generalized databases is described as Theorem 4.4.1.

Theorem 4.4.1 *A update operator satisfies axioms (U1) - (U8) if and only if there exists a faithful assignment that maps each generalized database I to a total pre-order $\leq_I$ such that for every other generalized knowledge-base ψ, μ, $Mod(\psi \diamond \mu) = \bigcup_{I \in Mod(\psi)} Mod(Min(\mu, \leq_I))$.*

Now we can define a concrete update operator as follows.

With respect to any generalized database I, a total pre-order $\leq_I$ is defined as: For each pair of generalized databases J_1 and J_2, let $J_1 \leq_I J_2$ if and only if $DBdist(I, J_1) \leq DBdist(I, J_2)$. Then an update operator $\diamond$ on knowledge-bases K and ψ can be defined as:

$$K \diamond \psi = \bigcup_{I \in K} Min(\psi, \leq_I)$$

Lemma 4.4.2 *The update operator* $\diamond$ *satisfies* $(U1) - (U8)$.

Proof Prove by showing that the update operator defined above satisfies the *faithful assignment* condition. For any generalized databases I and J, if $I \neq J$, then $DBdist(I, J) > 0$, but $DBdist(I, I) = 0$. So we have $DBdist(I, J) > DBdist(I, I)$. By definition, $I <_I J$ holds. Hence, we have defined a *faithful assignment* that maps each linear constraint database I to a pre-order $\leq_I$ such that for every knowledge-bases ψ, K, $Mod(K \diamond \psi) = \bigcup_{I \in Mod(K)} Mod(Min(\psi, \leq_I))$. ∎

Suppose we have a set of pictures represented by a generalized knowledge-base. Given a new set of pictures, also represented by a knowledge-base, the update operation results in those pictures that have minimal distance with respect to one of the pictures in the second set, i.e., as long as it is most similar to a certain picture, not necessarily to closest to all the pictures. The faithful assignment condition is shown to be satisfied because, if any picture in the second set appears also in the original set, then this picture has the minimal distance zero to the same picture in the original set, while other pictures do not appear in the original set have a positive distance, which is greater than zero.

Example 4.4.1 Suppose we have a knowledge-base μ contains summer weather information, average high temperature and average low temperature of three cities, Lincoln, Boston, and Miami. Another knowledge-base ψ contains the summer weather information of Beijing. The pictures are shown in Figure 4.2. We want to find which city has the most similar summer weather to Beijing.

The query can be solved using the update operator $\diamond$. Applying $\psi \diamond \mu$, the result is that the average high temperature of Lincoln is most similar to that of Beijing and the average low temperature of Boston is most similar to that of Beijing. So picture (1) and (4) are returned as the updated knowledge-base.

4.5 Arbitration

Arbitration is another kind of change operation related to revision and update. [51] proposes a set of axioms that an arbitration operator over knowledge-bases must satisfy. Considering the generalized knowledge-bases, we adopt the tuned axioms from [52] as following.

(A1) $Mod(\psi \triangleright \mu) \subseteq Mod(\mu)$.

(A2) If ψ is empty, then $\psi \triangleright \mu$ is empty.

(A3) If μ is non-empty, then $\psi \triangleright \mu$ is non-empty.

(A4) If $Mod(\psi_1) = Mod(\psi_2)$ and $Mod(\mu_1) = Mod(\mu_2)$, then $Mod(\psi_1 \triangleright \mu_1) = Mod(\psi_2 \triangleright \mu_2)$.

(A5) $Mod((\psi \triangleright \mu) \cap \phi) \subseteq Mod(\psi \triangleright (\mu \cap \phi))$.

(A6) If $Mod((\psi \triangleright \mu) \cap \phi)$ is non-empty, then $Mod(\psi \triangleright (\mu \cap \phi)) \subseteq Mod((\psi \triangleright \mu) \cap \phi)$.

(A7) $Mod((\psi_1 \triangleright \mu) \cap (\psi_1 \triangleright \mu)) \subseteq Mod((\psi_1 \cup \psi_2) \triangleright \mu)$.

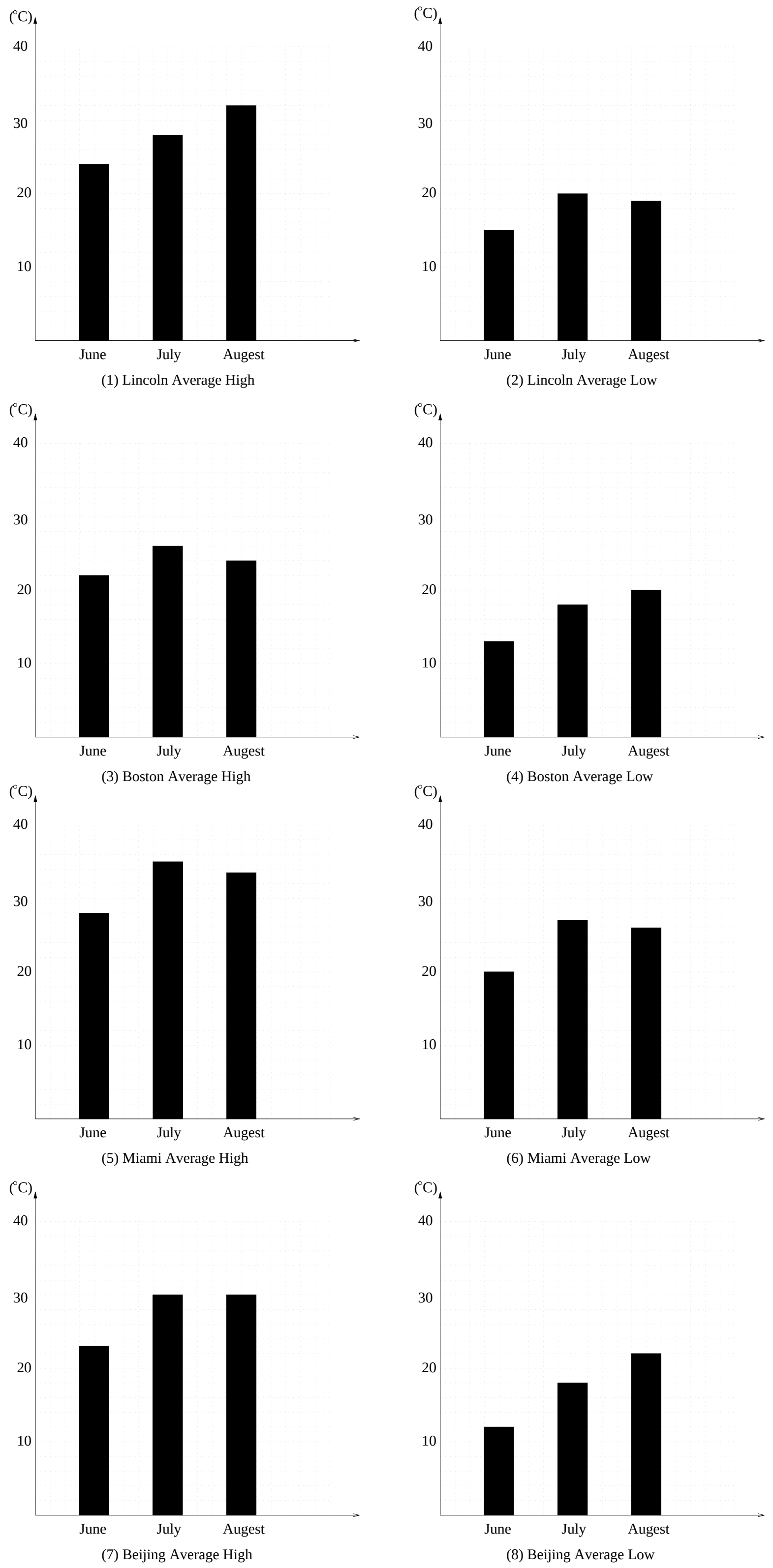

Figure 4.2: Update Operation

The two conditions that a *faithful assignment* should satisfy are as following:

(1) If $Mod(\psi_1) = Mod(\psi_2)$ then $\leq \psi_1 = \leq \psi_2$.

(2) If $I \leq_{\psi_1} J$ and $I \leq_{\psi_2} J$ then $I \leq_{\psi_1 \cup \psi_2} J$.

The second condition asserts that any generalized database that is closer to another generalized database in both ψ_1 and ψ_2 must also be closer in $\psi_1 \cup \psi_2$.

In [51], arbitration operators are also characterized as accomplishing a minimal change in terms of model-theoretic methods.

Theorem 4.5.1 *The knowledge base operator $\rhd$ satisfies axioms (A1) - (A7) if and only if there exists a faithful assignment that maps each knowledge base K to a total pre-order $\leq_K$ such that $Mod(K \rhd \psi) = Mod(Min(\psi, \leq_K))$.*

To define a concrete arbitration operator, we first define with respect to any knowledge-base K a total pre-order $\leq_K$ as follows. For each pair of generalized databases I and J, let $I \leq_K J$ if and only if $Odist(K, I) \leq Odist(K, J)$. Then the arbitration operator $\rhd$ can be defined as:

$$K \rhd \psi = Min(\psi, \leq_K)$$

Lemma 4.5.2 *The arbitration operator $\rhd$ satisfies $(A1) - (A7)$.*

Proof It is obvious that condition (1) is satisfied.

For condition (2), we prove by contradiction.

Assume $I \leq_{\psi_1} J$ and $I \leq_{\psi_2} J$ but $I \not\leq_{\psi_1 \cup \psi_2} J$. By the definitions of $Odist$ and $\leq$, it holds that $Odist(\psi_1 \cup \psi_2, I) \not\leq Odist(\psi_1 \cup \psi_2, J)$, which means that there exists some I', $I' \in \psi_1$ or $I \in \psi_2$, such that $DBdist(I, I') = max_{I'' \in \psi_1 \cup \psi_2} DBdist(I, I'')$ and $DBdist(I, I') \not\leq max_{J'' \in \psi_1 \cup \psi_2} DBdist(J, J'')$. So either $I \not\leq_{\psi_1} J$ or $I \not\leq_{\psi_2} J$ holds. This is a contradiction. Hence we proved that condition (2) is also satisfied. ∎

Suppose we have a set of pictures represented by a generalized knowledge-base. Given a new set of pictures represented by another knowledge-base, the arbitration operation gives out those pictures whose maximum distance with respect to the second set of pictures is minimal, i.e., pictures that are less different from every pictures in the original set. If there are two sets of identical pictures, then any other picture has same *overall distance* with respect to the two sets. The pictures which are result of arbitration operations on both picture sets, are also contained in the result of arbitration operation on the union of the two picture sets.

Example 4.5.1 Suppose several neighborhood families plan to construct a swimming pool in their resident area. There are several pieces of idle land around their houses. The shapes of the lands are shown as Figure 4.3(a1)-(a3). Each family provides a blueprint of the pool which they are favorite, as shown in Figure 4.3(b1)-(b3). They have to negotiate to decide which land is good for all.

Now we can use the arbitration operation on this problem. First we calculate the similarity between each pair of pictures of set (a) and set (b). Since Picture (a1) is close to any of the three pictures in set (b), the result of arbitration is (a1), which means if the land look like picture (a1) is chosen, all the families will not be too much unsatisfied.

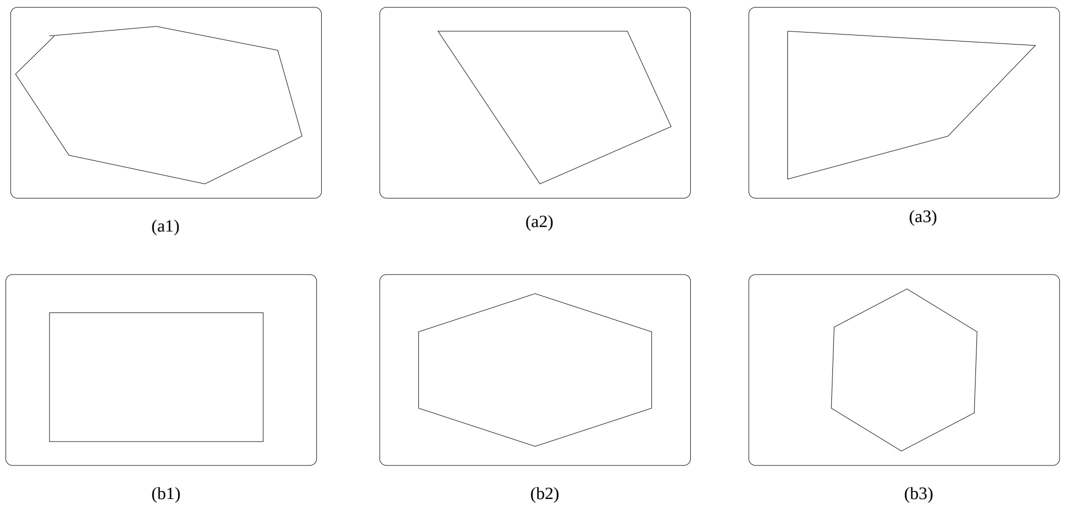

Figure 4.3: Arbitration Operation

4.6 Conclusion

The aim of this chapter is to address the issues of model-based change operators for linear constraint databases. The problems have been studied in the framework of linear constraint databases which is used to represent two-dimensional spatial objects. The models of a knowledge-base obtaining satisfaction of the principles are also those accomplishing a minimal change underlying model-theoretic characrizations. We have defined the revision, update and arbitration operators for linear constraint databases. The key idea is to find a good measure of the distance between possible models. We have applied our similarity measure methods as the basis of our change operators which are proved to carry the minimal change characteristics. It is possible to design algorithms based on this approach to make the operators amenable to computer solutions. We would like to point out the importance of our study because in addition to the promising of constraint databases and the wide application areas of similarity queries, our change operators provides the characterizations that general database change operators must satisfy. This is an indispensable step towards practical constraint databases.

This research can be extended in various direction. First, the comparison of the distance measure and operators with other proposals. Second, the efficient implementation of the algorithms aimed at the distance measures and operator evaluations. Finally, the extension of the approach to a general framework of constraint database changes.

Chapter 5

Computational Complexity

5.1 Preliminaries

The similarity measure and change operators are defined based on the distance computation of two polygons. Recall that the distance between two polygons is defined as follows:

$$PGDist(P_1, P_2) = DistCPnt + \alpha * sin(rttAngle) * edgeLenSum +$$

$$\beta * scaleChange * edgeLenSum + \gamma * \sqrt{totalArea - commonArea}$$

where α, β, γ may be any positive constants as weights for each component.

To consider the computational complexity of the similarity measure and change operations, we first give the computational complexity of computing the distance between two polygons. To calculate the $PGDist$, we give a method for calculating $rttAngle$, $edgeLenSum$, $scaleChange$, $DistCPnt$, area of a polygon and the intersection area or union area of two polygons. In this section, we introduce some existing algorithms which solve the above problems efficiently. The computational complexity of these algorithms are presented.

A polygon is *simple* if there is no pair of nonconsecutive edges sharing a point. We consider only simple polygons, so we use polygon and simple polygon interchangeably in the rest of the chapter. A polygon is *convex* if, for any two points p and q in the polygon, the segment pq is entirely contained in the polygon. Each polygon (not necessarily convex) can be represented as a set of convex polygons. So we take into consideration convex polygons in this section as a basis for further analysis on general polygons in the next sections.

5.1.1 Basic Algorithms

In linear constraint databases, a two-dimensional N-vertex polygon is represented by a set of N linear inequalities (constraints) of the form

$$a_i x + b_i y + c_i \geq 0$$

where x, y are variables, a_i, b_i, c_i are rational constants for $i = 1, 2, \ldots, N$. Note that here we assume there are no redundant inequalities in the constraint databases. Otherwise each inequality represents a half-plane, then we first need to find the intersection of the N half-planes which consists a convex polygonal region. It has been proved in [47] that the intersection of N half-planes can be found in $O(NlogN)$ time, and this is optimal.

To compute the geometric properties of a polygon, we usually need to know the vertex set or the edge set which defines the polygon hull. We need to represent a polygon by a sequence of vertex pairs (v_i, v_j), with v_i, v_j consecutive vertices and $1 \leq i, j, \leq N$. The edge $v_i v_j$ is implied by the pairing relationship between v_i and v_j. The order of the vertex pairs is fixed to be either in clockwise or in counterclockwise direction. For instance, a polygon with vertices $v_1, v_2, \ldots, v_N$ in the order of clockwise is represented by the sequence $(v_1, v_2), (v_2, v_3), \ldots, (v_{N-1}, v_N), (v_N, v_1)$.

To transform a set of N linear inequalities into a sequence of N vertex pairs, we may use the following algorithm.

Algorithm 5.1.1

```
Let $C$ be the set of $N$ linear inequalities.
Procedure TRANSLATION($C$)
Begin
1. For each inequality
2.         calculate the intersect point with all the other inequalities;
3. For each intersection point
4.         If it is external to the polygon
5.         discard it;
6.         Else
7.         keep it as a vertex of the polygon;
8. Let $b$ be a vertex of the polygon;
9. While not all the vertices found their neighbors
     Begin_while
10.           Find a neighbor of $b$;
11.           Keep the pair, $b$ and $b$'s neighbor as a vertex pair of the polygon;
12.           Let $b$'s neighbor be $b$;
     End_while
End.
```

Example 5.1.1 *Suppose we are given the following set of linear inequalities:*

$x - 3y + 48 \geq 0$

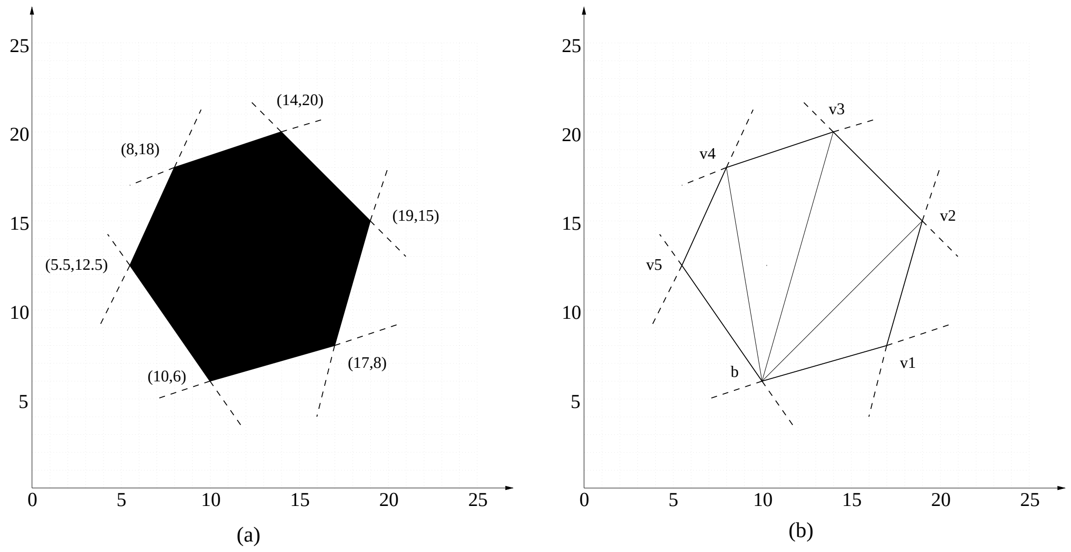

Figure 5.1: Translation and Triangulation

$-x - y + 34 \geq 0$

$-7x + 2y + 103 \geq 0$

$-2x + 7y - 22 \geq 0$

$13x + 9y - 182 \geq 0$

$11x - 5y + 2 \geq 0$

The polygon and the intersection points of all these inequalities which are internal to the polygon are shown in Figure 5.1(a). Let's begin with point b and go along the clockwise direction to do the while loop. Finally, the vertex pairs $((10,6),(5.5,12.5))$, $((5.5,12.5),(8,18))$, $((8,18),(14,20))$, $((14,20),(19,15))$, $((19,15),(17,8))$, $((17,8),(10,6))$ *are found. We can then use them to calculate the related geometric properties of the polygon instead of the set of inequalities.*

The performance analysis of above algorithm is straightforward. The first two steps access each of the N inequalities $N-1$ times, so steps 1-2 cost $N(N-1)$ time. For N lines, there exist at most N^2 intersection points. Step 4 costs $O(N)$ time. So steps 3-7 use $O(N^3)$ time. Step 9 repeats N times because it is an N-vertex polygon. Step 10 costs at most $O(N)$ time. So it is clear that steps 9-13 use $O(N^2)$ time. In summary, a set of N linear constraints can be translated into its vertex pair representation in at most $O(N^3)$ time.

We believe that Algorithm 5.1.1 is not optimal, but it is a simple and easy to implement algorithm. In the rest of this chapter, our analysis of the computational complexity is based on the vertex-pair sequence representation of polygons.

The center point (Centroid) of a finite set of points $p_1, p_2, \ldots, p_N$ is their arithmetic mean $(p_1 + p_2 + \ldots + p_N)/N$. The centroid of a set of N points in k dimensions an be computed trivially in $O(kN)$ arithmetic operations. In our case, $k = 2$.

Each vertex pair (v_i, v_j) represents an edge of the polygon. The length of this edge can be computed using the formula $\sqrt{(x_{v_i} - x_{v_j})^2 + (y_{v_i} - y_{v_j})^2}$. This computation takes constant time. For a polygon with N edges, the sum of the length of its edges can be computed in $O(N)$ time. If another polygon has M edges, without losing generality, assume $N > M$, then the $edgeLenSum$ can be computed in at most $O(N)$ time.

The $scaleChange$ calculates the difference between the sum of the length of two polygons. For two polygons with at most N edges each, it is obvious that the $scaleChange$ can be computed in $O(N)$ time.

For two lines i and j, their slopes are $-\frac{a_i}{b_i}$ and $-\frac{a_j}{b_j}$, respectively. So the inside angle can be calculated using the formula $|arctan(-\frac{a_i}{b_i}) - arctan(-\frac{a_j}{b_j})|\ mod\ \frac{2}{\pi}$. For two polygons with at most N edges for each of them, the average inside angle can be computed in $O(N)$ time. Finally, to find the minimal average value, *rttAngle* , $O(N^2)$ time is needed.

5.1.2 Area Computation

By E^d we denote the *d-dimensional Euclidean space.* The *convex hull* of a set of points S in E^d is the boundary of the smallest convex domain in E^d containing S. The *triangulation* problem is stated as follows: Given N points in the plane, join them by nonintersecting straight line segments so that every region internal to the convex hull is a triangle [47].

Now we have a sequence of N vertex pairs representing a convex polygon. The triangulation of the polygon can be done in $O(N)$ time by following simple algorithm.

Algorithm 5.1.2

```
Let $S$ be the vertex pair sequence $(v_1, v_2), (v_2, v_3), \ldots, (v_N, v_1)$.
The function $next(b)$ returns the $m$ if $(b, m)$ is an element of $S$.
**Procedure** TRIANGULATION($S$)
**Begin**
1. Let $b = v_2$, $e = v_N$;
2. **While** $b \neq e$
     Begin_while
3.          Let $m = next(b)$;
4.          The triple $(v_1, b, m)$ is a triangle internal to the polygon;
5.          Let $b = m$;
      End_while
**End.**
```

Example 5.1.2 *Consider the set of vertices in Figure 5.1. The convex hull S in this case is* $(b, v_5), (v_5, v4), (v_4, v_3), (v_3, v_2), (v_2, v_1)$ *and* (v_1, b). *After calling the triangulation*(S) *algorithm, we get a triangulation* $\{(b, v_5, v_4), (b, v_4, v_3), (b, v_3, v_2), (b, v_2, v_1)\}$ *as shown in Figure 5.1(b).*

The area of a triangle can be calculated using the formula $\sqrt{s(s-a)(s-b)(s-c)}$ where a, b and c are the length of each edge of the triangle and s is half of the length sum of the three edges. The calculation of the length of an edge costs constant time. The triangulation of an N-vertex polygon produces $N-2$ triangles. So the area of an N-vertex polygon can be computed in $O(N)$ time using the above algorithm.

Given two convex polygons P with N vertices and Q with M vertices, it is proved in [47] that the intersection of P and Q is a convex polygon having at most $(N+M)$ vertices. [47] also illustrates a method that provides the proof that the intersection of a N-vertex convex polygon and a M-vertex convex polygon can be found in $O(N+M)$ time. Additional $O(N+M)$ time is sufficient to decide, if necessary, among the alternatives $P \subseteq Q$, $Q \subseteq P$, or $P \cap Q = \emptyset$. Thus the intersection area of two polygons with N and M vertices, respectively, can be computed in $O(N+M)$ time.

The union area of two polygons equals to the area sum of the two polygons minus their intersection area, which can be calculated in a constant time once we calculated the union and the intersection areas of the two polygons.

5.1.3 The Assignment Problem

The *Assignment Problem* is generally stated as: For what man-job assignment is the total cost minimized, assuming the cost of assigning man i to job j is a_{ij} [39]. A complete bipartite graph, $B = (V, U, E)$, with $|V| = n$, $|U| = m$, $m \leq n$, $|E| = n \times m$, is used to formalize the man-job assignment problem. Here the vertices in V represent men and vertices in U represent jobs. The cost a_{ij} is the weight of the edge ij. then the assignment problem is to find a maximum match that minimizes the sum of the weights.

In the last decade, there are a large number of studies aiming at efficient implementations of the assignment problem. Currently the best known strongly polynomial time bound of $O(n^3)$ is achieved using the classical Hungarian method [36, 39, 46]. [30] gives a primal method which is practical for implementation and also achieves $O(n^3)$ complexity. Under the assumption that the cost are integers ranging $[-C, \ldots, C]$, an $O(l\sqrt{n}log(nC))$ time algorithm where l denotes the number of edges is obtained using cost scaling and blocking flow techniques by [19, 20]. [18] studies the implementation of the scaling push-relabel method for the assignment problem aimed at improving practical performance. Based on different heuristics, they develop several codes and improve the code performance on many problem classes.

5.2 Computational Complexity of the Similarity Measure

5.2.1 Computing Distance between Two Polygons

Based on the analysis presented in the previous section, we know that the distance between two polygons $PGDist$ can be computed in time $O(N)$ for $DistCPnt$, plus $O(N)$ for $rttAngle$, plus $O(N)$ for $edgeLenSum$, plus $O(N)$ for $scalechange$, plus $O(N)$ for the $commonArea$ and $totalArea$. The sum equals to $O(N)$. This is a linear complexity in terms of the number of edges of the convex polygon.

However, polygons are not always convex. Fortunately, the proposed distance measure can also handle polygons that are concave. In linear constraint databases, concave polygons are represented as the union of a set of convex polygons in the same relation name, which is normally used to denote the polygon. The method of dividing a concave polygon into a set of convex polygons is not unique. Nevertheless, the maximum number of convex sub-polygons is not greater than $N - 2$ with N the number of edges (or vertices) of the concave polygon. The maximum is achieved if the polygon is triangulated because a triangle is a minimal polygon. The way of breaking up a polygon does not change the value of the factors involved in the distance measure formula. Now there are two extreme cases to consider.

Given a N-vertex polygon and a M-vertex polygon, $M \leq N$. First we translate each polygon into a set of convex polygons by triangulation. We get a set of $N - 2$

triangles and a set of $M - 2$ triangles, respectively. For an edge which is not a component of the polygon's hull, always appears twice in the set of inequalities. Since a triangle consists of a constant number, three, of edge the factors for distance computation can be obtained in $O(N)$ time. So the distance can be computed in $O(N)$ time.

Instead of triangulation, we may represent the two polygons by a set of minimal number convex polygons. Now each polygon is translated into a set of no more than k convex polygons with k some constant, and each convex sub-polygon may contain at most $N - 2$ edges. It is clear that, in this case, the distance between the two polygons can also be computed in $O(N)$ time.

In conclusion, the computational complexity of the distance between two polygons with at most N edges each, is linear in the number of edges of the two polygons, i.e., the computation has complexity $O(N)$.

5.2.2 Computing Similarity of Two Linear Constraint Databases

Suppose we have two linear constraint databases, D_1 and D_2, each of which contains B polygons and each polygon has at most N vertices. At first, we translate each set of constraints in the same relation name that represents a polygon into the representation of a sequence of vertex pairs. The translation can be done in $O(B) \times O(N^3)$ time.

Now we construct a weighted complete bipartite graph $G = (V_1, V_2, E)$ as follows.

Each polygon in D_1 is a vertex in V_1. Each polygon in D_2 is a vertex in V_2. $|V_1| = |V_2| = B$. E is the set of all the edges with one end point in V_1 and the other in V_2. $|E| = B \times B$. The distance between two polygons $PGDist$ is assigned as the weight of the corresponding edges. The construction of the bipartite graph can be done in $O(B^2) \times O(N)$ time.

To find the match with minimal sum of weights, we adopt the algorithms described in [18] which presents an algorithm with complexity of $O(B^2\sqrt{B}log(BC))$ with C some constant integer no less than any weight of the edges.

In our case, it is hard to look ahead and find a C which bounds the maximum distance between any two polygons. Suppose there does exist some C, if it has to be a very huge number compared to the B, then the performance improvement by using the scaling push-relabel method is overshadowed. However, when we are dealing with spatial scenes, it is reasonable to assume that the scenes are displayed in a piece of plane such as a map, or a picture. Then we can say that the size of the piece of plane is never larger than a certain size, otherwise we can change the scale of the map, for instance, and adjust the spatial scenes to fit the plane. In this sense, with k denoting the length of the boundary of the plane, the distance of any two polygons is within $O(k^2)$, because the distance measure formula is a quadratic function in terms of the length of polygon's edge. Thus the algorithm proposed in [18] is profitable and feasible to our measure.

[47] proves the following:

Proposition 5.2.1 If problem ψ can be solved in $T(n)$ time and problem ϕ is $\tau(n)$-transformable to ψ ($\phi \propto_{\tau(n)} \psi$), then ϕ can be solved in at most $T(n) + O(\tau(n))$ time.

According to the proposition, the distance between two linear constraint databases can be computed in at most $O(BN^3 + B^2N + B^2\sqrt{B}log(BC))$ time.

5.3 Computational Complexity of the Knowledge-base Change Operators

The definitions of the change operators are based on the distance between a knowledge-base K and a generalized database I denoted as $Dist(K, I)$ which equals to $\min_{J \in K}\ DBdist(I, J)$ and the overall distance denoted as $Odist(K, I)$ equals to $\max_{J \in K}\ DBdist(I, J)$. We denote the computational complexity of the distance measure $DBdist$ between two linear constraint databases as $O(L\_CDB)$.

Given knowledge-base K and knowledge-base ψ, each of which is a set of at most P generalized databases. For each $I \in \psi$, we first calculate the $Dist(K, I)$, which costs at most $O(P) \times O(L\_CDB)$ time. To accomplish a revision operation $K \circ \psi$, we need to calculate all the $Dist(K, I)$ with $I \in \psi$ and find the minimal value. So the revision operation costs $O(P^2) \times O(L\_CDB)$ time.

For the update operation $K \diamond \psi$, each generalized database in ψ is used to calculate $DBdist$ corresponding to every generalized database in K. There are totally

P^2 calculations of $DBdist$, meanwhile the minimal value is selected. So an update operation costs $O(P^2) \times O(L\_CDB)$ time.

Since the calculation of $Odist$ has the same complexity as that of $Dist$, similar to the revision operation, an arbitration operation $K \triangleright \psi$ also costs $O(P^2) \times O(L\_CDB)$ time.

5.4 Conclusion

We have considered the computational complexity of finding the most similar spatial scenes represented by linear constraint databases, from a knowledge-base which contains a set of spatial scenes that have to be revised, updated or arbitrated. The operations yield the set of spatial scenes with minimal deformations required to transform one to another. The main discovery is that for all three change operators, the evaluations are in the complexity level of polynomial time solvable in terms of the number of objects in a knowledge-base and the number of edges of each object. The complexity result does not change even for handling complex scenes, concave objects with holes, irregular shapes and line-region relations because any complex scenes can be represented as sets of convex polygons and the number of convex polygons do not grow beyond a polynomial in the number of vertices in the complex scene. As we mentioned earlier in Section 5.2.1, the way of breaking up a complex scene does not effect the distance measure.

For most problems in constraint databases, it is unrealistic to expect a polynomial solution. Due to finding suitable restrictions for the various constraint formalisms, our approach eliminates the sources of intractability. The complexity results give strong evidence that the similarity measure and change operations can be evaluated efficiently. At the same time the operations accomplish a model-theoretic minimal change. Zeng [67] implements a model-based arbitration operator over knowledge-bases containing sets of formulas. Models are represented by vectors. In terms of the number of vectors, it gives a naive approach in complexity $O(n^2)$. The computer experiments for propositional databases show that the operators can be evaluated efficiently.

Chapter 6

Perceptual Experiments on Similarity Measures

It is easy to come up with a large number of similarity measures that are computable using computers. It is easy to distinguish two different objects by human vision. What is difficult is teaching computers to tell the similarity of two things in a manner intuitive to humans. This chapter describes some investigations on how similarity measures correlated with human intuitions, with an emphasis on the experiments engaged by Brian Boon [2].

6.1 Previous Research

A picture may be represented by points, lines or polygons. The distortions of position, extension and rotation of each component are frequently used factors to measure similarities.

With a picture consisted of a collection of points, Knapp [28] does some research to see how the distortion of individual points in a picture would affect human's perception of similarity. Presented with a prototype image composed of ten randomly placed points and then a distortion image of the same ten points, the individual establishes the similarity between the two images. Research in [28] shows that as more of the points are perturbed then one's perception of similarity decreases.

Dodwell's research [10] takes consideration of the effects of rotation across different vector pattern classifications. Two properties of each vector pattern are counted in, position and orientation. Their findings indicate that the effects of rotation depend upon the type of the vector pattern.

Pigeons are thought of capable of attending to common aspects of drawings in Kirtpatric-Steger's study [37]. After some training, it is shown that pigeons are able to discriminate among the line drawings of several rearranged different objects. The pigeons display different degrees of generalization decrement to the different scrambled versions of the objects.

Studies and experiments done by the psychology community provide some insight into what the perceiver finds important in a picture, as above researches. However, they do little in the way of providing applications. Applications for similarities are widely studied in the database and image retrieval areas, and can be found in [4, 14, 17, 23, 25, 26] and [41]. These studies were mentioned or discussed in previous chapters. Each article contains good ideas about what similarity is and the best ways to use the similarity measure in image retrieval. However, they study only design and implementation issues but no testing on how intuitive to human perception these ideas are.

6.2 Investigations on Linear Similarity Measures

Recall the similarity measure proposed in Chapter 3. The similarity measure has the basic form

$$Sim(V_1, V_2) = \sum_{i=1}^{n}(\alpha Rotation_i + \beta Translation_i + \gamma Extension_i)$$

Where V_1 and V_2 are two pictures each with n line segments and i indicates the ith line segment of V_1 and V_2. To discover a similarity measure intuitive to human perception, perceptual experiments are established and tests are proceeded by Brian Boon and Dr. Revesz [2]. The values of α, β and γ are studied to find a best combination of the amounts that each of rotation, translation and extension contribute to perceived similarity. The results also show how well the proposed similarity measure correlated to human intuition.

6.2.1 Methods and Procedures

Linear regression is employed in [2] to estimate the parameters for the linear model

$$E(Y) = \beta_0 + \beta_1 x_1 + \Lambda + \beta_k x_k$$

. Here we can look Y as Sim, x_i as the factors contributing to Sim and β_i as the constants to be associated with each factor. Let

$$\text{Y} = \begin{vmatrix} y_1 \\ y_2 \\ \Lambda \\ y_n \end{vmatrix} \quad \beta = \begin{vmatrix} \beta_0 \\ \beta_1 \\ \Lambda \\ \beta_k \end{vmatrix} \quad \epsilon = \begin{vmatrix} \epsilon_1 \\ \epsilon_2 \\ \Lambda \\ \epsilon_n \end{vmatrix} \quad \text{X} = \begin{vmatrix} x_0 & x_{11} & x_{12} & \Lambda & x_{1k} \\ x_0 & x_{21} & x_{12} & \Lambda & x_{2k} \\ x_0 & \Lambda & \Lambda & \Lambda & \Lambda \\ x_0 & x_{n1} & x_{n2} & \Lambda & x_{nk} \end{vmatrix}$$

with $x_0 = 1$. In this way, the equations representing each of the testing data sets are expressed as

$$Y = X\beta + \epsilon$$

. Where ϵ possesses some probability distribution with $E(\epsilon) = 0$. Then the estimators $\hat{\beta}$ for the parameters $\beta_0, \beta_1, \Lambda, \beta_k$ are given as

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Examples of finding a regression formula can be found in [2]. Now it is interesting to find out how accurate a regression formula is, that is, how well the regression formula is correlated with the actual data. To do this, a r-value is defined and calculated in [2]. Let $SS_{error} = \sum(Y - \hat{Y})^2$, $SS_y = \sum(Y - \bar{Y})^2$ where $\hat{Y}$ is the estimator of Y and $\bar{Y}$ is the average of Y. Another definition of SS_{error} adapted in [2] is

$$SS_{error} = (1 - r^2) SS_y$$

So r is solved as

$$r = \sqrt{1 - SS_{error}/SS_y}$$

The range of r-value is from 0 (no correlation) to 1 (perfect correlation).

Three experiments are designed and tested upon the similarity measure proposed in Chapter 3 and upon the following measure:

$$Dist(l_1, l_2) = \alpha * DistCPnt + \beta * LengthDiff + \gamma * AngDiff$$

where $LengthDiff$ is the difference in length of the two line segments and $AngDiff$ equals to the sum of two line segments' length times $\sin\theta$ with θ being the minimal inside angle of the two line segments.

In the first experiment, two figures, constellations Ursa Minor and Columba are used as prototype pictures. They are made up of line segments. Distortions to three randomly chosen line segments are in the range of 0, 15, and 30 degrees/pixels. All possible combinations of distortion across one, two or three line segments with distortions of 0, 15 and 30 are generated and presented to the subject for comparison to the prototype picture. The two pictures are displayed on a computer monitor, and order of display is randomized for each subject. As the subject are presented each distortion/prototype pair they are asked to decide how similar the two pictures are. Similarity is given a rank from one to ten where one represents very similar, and ten means very dissimilar. This ranking is used over through the next two experiments also.

The second experiment is aimed at testing which is more significant, rotation and contraction or rotation and extension of a line segment. A random set of line segments and their distortions, either rotation and contraction or rotation and extension, are presented to each subject. The distortion is generated using the same method as the first experiment and each subject is asked to decide the similarity, given a rank from one to ten.

In the third experiment, rotation , translation and extension are considered. The prototype picture consists of a set of ten random line segments. Distortions are created across rotation , translation and extension in the way that the first experiment employs. The subject is presented with all the combinations of distortions and is asked to decide the similarity.

Thus, values of Sim, in the other word, Y is collected from each subject's similarity measure. The factors contributing to the similarity measure equations are computed across each distortion. In this way, applying the linear regression method, the constants α, β, and γ are estimated and the r-value for each equation can be calculated and analyzed.

6.2.2 Results

Data collected from the first experiment reveal a close association between how the similarity measures predict similarity of the two chosen pictures. In following table, Measure 1 is the measure introduced in this chapter, Measure 2 is the measure proposed in Chapter 3. The numbers are r-values corresponding to each picture and measure.

Measure 1		Measure 2	
Ursa	Columba	Ursa	Columba
0.635	0.639	0.718	0.753

It appears that a linear regression equation, which incorporates the weighted distance measure is a very accurate model for predicting similarity among stick figures.

The most interesting observation from the result of the second experiment is that the accuracy of Measure 1 is improved when translation is not considered. Although Measure 1 and Measure 2 perform 100% satisfiable, it does perform well enough. This means that we were able to develop similarity measure that is simple and fast.

The following table contains the r-values for the third experiment.

Measure 1		Measure 2	
Ursa	Columba	Ursa	Columba
0.729	0.735	0.733	0.739

From this table, we notice that the two measures are performing closely well and they have improved over the previous two experiments.

6.3 Discussions

It is hard to say which measure is a best one. From the Boon's experiments, we can find that the proposed similarity measure performs well and it is interesting to discover different means to improve the measure. For example, what is the role that the orientation of a picture plays in similarity? What if the prototype and distortion pictures are displayed with different sizes, and then what is the significance of each factor?

Further, based on the linear regression model, it is possible to assign the constants to each factor of the equation dynamically. A user can have a customized similarity measure tailored to what that user thinks is important in the pictures. This can be integrated into MLPQ/GIS system and enable a personalized measure environment, which in return, will improve the precision of similarity retrieval.

Chapter 7

Implementations

7.1 A Review of MLPQ/GIS System

The MLPQ/GIS system is a constraint database system that handles various spatial, temporal and GIS data [?]. The system allows constraint relations with any number of attributes to be represented using linear constraints. The graphic user interface provides visual and convenient geo-spatial-temporal data querying. For beginning users, icon-based queries are available to do basic spatial queries, e.g., the intersection of several relations, which are displayed as sets of polygons or line segments using different colors. For advanced users, more general queries based on Datalog query rules are provided. The basic queries are internally translated into conjunction queries and conjunctive queries are translated into a procedural algebraic language optimized using algebraic optimizations and then evaluated based on efficient imple-

Figure 7.1: Steps in Query Processing

mentations of basic queries, selection, projection, join, union and intersection etc. of constraint databases. Figure 7.1 depicts the steps involved in query processing of the MLPQ/GIS system. The elements shown as dash lines are under research and not fully implemented in the system.

In MLPQ/GIS, data are stored in a constraint database format. The user is not concerned with the storage but the visual display of the objects in the database. Various data entering tools are provided by the system. For example, the system allows the user to draw pieces of land in the form of polylines, or polygons. The drawings are saved as constraint relations automatically. The user is able to save directly constraint relations in specific constraint database format. When related queries are presented, the files are loaded and spatial objects represented in this file are displayed.

The MLPQ/GIS system has developed great functionality to facilitate GIS applications. The Area query takes a bound as input and output the total area falling within the bounds. The Buffer query finds out the points that locate inside a specified buffer region. Aggregation functions like Max or Min are also implemented. Using Datalog rules, recursive queries which relational database are not capable to process, are realized in the MLPQ/GIS system [?]. One of the advantages of constraint database is that handling spatial-temporal information is much convenient and efficient against other data models. Another feature of the MLPQ/GIS system is its animation functionality [40] - changing objects and events can be simulated, snapshot of a given time point can be caught and displayed on screen. Further, similarity queries and database updates are implemented, which makes the system even more powerful and unique.

7.2 Implementation of Similarity Queries

Given a template object, select all objects that are similar to the given one is called a similarity query. Similar-based retrieval has a wide range of applications. Based on the proposed similarity measure, the similarity query is implemented as a module within the MLPQ/GIS system. Figure 7.2 shows the graphic user interface of the system.

After import a constraint database which contains a large number of constraint relations with each relation represents a picture, and then the user clicks the 'S' icon on the function bar, a dialog window is prompt to specify the prototype picture.

Figure 7.2: GUI for Similarity Query

The constraint relations are then translated into 2-Spaggheti model, in which each line segment is represented using its two end points. The user is not concerned with this translation. What they see is only the result of the distance calculation. By representing the line segments using end points, the algorithms employed to compute the distance between two line segments are efficiently implemented. To solve the bipartite matching problem, we study the practical implementations among variants of the algorithms [18, 20, 30]. The scaling push-relabel method [18] is employed in our implementation.

The number beside each relation name is the distance between the picture and the prototype picture. The pictures are sorted according to the distance, as shown in Figure 7.2. A click on the relation name of the prototype picture brings up the picture on the right window. Another click on any other relation name displays the picture over the prototype using different color. In this way, the user can easily see the difference between the two pictures. Please be aware that only the stick figures or the sketch of a picture is considered at current implementation stage. Further functional improvement can be made to rank the distance or similarity and allow the user to apply some filters to display the top ranked ones. It also may be friendly to display each picture in a separate window when the pictures are complex.

Chapter 8

Conclusions

A design for similarity queries for spatial databases covers three main areas of database research. The three areas are data representation method, similarity measure and query processing technique. Each area has been examined in the light of the unique problems of a linear constraint database system, and new ways of exploiting the various aspects of constraint databases have been addressed.

An important feature of constraint databases is the ability to represent spatial information conveniently. Finitely representing the infinite and non-enumerable set of points of spatial objets allows more powerful spatial queries in constraint databases. In Chapter 1, previous research work on similarity-base spatial information retrieval was investigated and some disadvantages were addressed. The concepts and merits of constraint databases are described in Chapter 2, showing how spatial objects can be represented under constraint data model, and how constraint queries can be expressed to exploit the advantages of a spatial database with constraints.

Chapter 3 proposed a new similarity measure for spatial objects. Concerning the typical geometric transformations such as transition, rotation and scaling, a new method to measure the similarities between 2-dimentional spatial objects, which are assumed to contain a set of line segments in the plane. Further extensions to measure the similarity between any spatial scenes took into consideration of the polygonal representation of spatial information. An extended similarity measure was presented and examples were given. The similarity measure proposed is independent of the data model that the spatial object representation uses.

The measure of the distance between two databases is a key issue not only for similarity queries, but also for developing change operators carrying the minimal change model-theoretic characteristics. Chapter 4 studied the challenges of high-level change operators for linear constraint databases. To solve the inconsistency due to database changes, the meaning of *minimal change* is worked out upon the measure of distance between possible models for the databases. Revision, update and arbitration operators were defined and the their model-theoretic characteristics were analyzed. It has been proved the three operators satisfy the axioms commonly recognized by the database research community. Examples presented in this chapter are an epitome of the wide application areas of high-level change operations.

Chapter 5 presented algorithms for computing the similarity measure and change operations in a linear constraint database system. This is an important step towards a practical implementation of the similarity queries and change operators. Analysis of the computational complexities showed that for a linear constraint database, efficient evaluation of similarity measures and change operations is feasible and potential.

Chapter 6 described several perceptual experiments on similarity queries that act upon a linear constraint database. A similarity measure needs to be not only computable efficiently but also intuitive to human perception. The experiments attempt to discover the values of the ratio put on each factor contributing to the similarity measure proposed in Chapter 3. The results of the experiments show that the proposed similarity measure performs an over 70% correlation on average with experiment data collected from actual human perception. It is observed that a personalized similarity measure environment can be constructed by dynamically assigning coefficients to each portion of the similarity measure equation.

The similarity query has been integrated into a linear constraint database system, the MLPQ/GIS system. Chapter 7 described the core functionality of the MLPQ/GIS system and the query processing techniques developed in this system. Implementation strategies of the similarity query and extension potentialities are addressed.

A basic theme of this dissertation has been to study the various issues to accomplish the advantages of a practical constraint database, particularly the ability of handling spatial information. One important application is similarity-based spatial retrieval. The other one is high-level change operations which enhance the intelligence of a database system. It is valuable to discover a similarity measure not only intuitive to human insight, but also leading to change operators that achieve model-theoretic minimal change property. The essence of linear constraint database for representing and accessing spatial information in a natural way and efficiently makes it a promising and exciting research area.

8.1 Future Work

The area of constraint database systems is relatively new, so there are many possible avenues for future work. In each of the areas presented in this dissertation there is much more to be done.

In the area of spatial information modeling, various techniques have been exploited upon 2-dimentional objects. As the demand of applications on multi-dimensional objects has increased rapidly, the multi-dimensional spatial modeling favors constraint databases. The ability of representing n-dimension objects and the expressive power of constraint databases have been conceived in this dissertation. It is attractive to extend the similarity measure method to 3-dimensional spatial scenes and develop efficient as well as general change operators of constraint databases. It is not obvious that, to accomplish this mission, how the data complexity would be and how efficient algorithms could be developed with diversity spatial information.

Indexing has been studied extensively for relational database systems and query optimization techniques are well established. In this dissertation, we examined the costs of similarity query and change operations, but somehow we need to employ indexing technique and constraint query optimizations to enhance the performance in terms of optimal worst-case access to secondary storage.

Finally, a simple analysis of our proposed similarity query method gives a fair estimation of computational complexity, but a very detailed test or simulation is needed to determine what are the precision and recall that can be reached in a real database system. Also, it would be interesting to see exactly to what extend we can take advantage of constraint data modeling in the data representation and query processing.

Bibliography

[1] C. E. Alchourron, P. Gardenfors, D. Makinson. On the Logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, Vol. 50, pp.510-530, 1985.

[2] Brian Boon. *Perceptual Experiments on Similarity Queries.* Master Thesis, University of Nebraska-Lincoln, 1999.

[3] A. Borgida. Language Features for Flexible Handling of Exceptions in Information Systems. *ACM Trans. Database Systems*, Vol. 10, 563-603, 1985.

[4] H. Tom Bruns, Max J. Egenhofer. Similarity of Spatial Scenes. *Seventh international symposium on spatial data handling*, Taylor & Francis, London, pp173 - 184.

[5] A. Brodsky, J. Jaffar, M.J. Maher. Toward Practical Constraint Databases. *Proc. 19th VLDB*, 322–331, 1993.

[6] A. Brodsky, Y. Kornatzky. The *Lyric* Language: Querying Constraint Objects. *Proc. SIGMOD*, 35–46, 1995.

[7] R. Bayer, E. McCreight. Organization of Large Ordered Indexes. *Acta Informatica*, 1:173–189, 1972.

[8] D. Comer. The Ubiquitous B-Tree. *Computing Surveys*, 11:2:121–137, 1979.

[9] M. Dalal. Investigations into a Theory of Knowledge Base Revision: Prelimilary report. *Proceedings AAAI-88, St. Paul, MN*, 475-479, 1988.

[10] P. Dodwell, T. Caelli. Recognition of Vector Patterns Under Transformations: Local and Global Determinants. *The Quarterly Journal of Experimental Psycology.* 1-23, 1985.

[11] Y. Deng, P.Z.Revesz. A Similarity Measure of Spatial Databases. *30th Jubilee International Conference of the Banki Donat Polytechnic.* Hungary, Sep. 1999.

[12] Y. Deng, P.Z.Revesz. Revision and Update Operators in Linear Constraint Databases. *Proc. First Midwest Conference of the American Association for the Advancement of Science*, Omaha, Nebraska, Nov. 1999.

[13] T. Eiter, G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updats, and Counterfactuals. *Artificial Intelligence*, Vol 57, 227-270, 1992.

[14] Myron Flickner, Harpreet Sawhney et. al. Query by Image and Vedeo Content: the QBIC System. *IEEE Compute* 28, 9, Sep. 1995. pp. 23-32.

[15] R. Fagin, J. D. Ullman, M. Y. Vardi. On the Semantics of updates in Databases. *Proc. 2nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 352-365, 1983.

[16] Andrew U. Frank, Mark Wallace. Constraint Based Modeling in a GIS: Road Design as a Case Study. *Cartography* pp. 177-186

[17] Amaranth Gupta, Ramesh Jain. Visual Information Retrieval. *Communications of the ACM*, Vol.40, No.5, pp.70-79.

[18] Andrew V. Goldberg, R. Kennedy. An Efficient Cost Scaling Algorithm for the Assignment Problem. *Mathematical Programming*, 71, (1995) 153-177.

[19] A. V. Goldberg, S. A. Plotkin, P. M. Vaidya. Sublinear Time Parallel Algorithms for Matching and Related Problems. *Journal of Algorithms*, 14, 180-213, 1993.

[20] H. N. Gabow, R. E. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM Journal on Computing*, 18, 1013-1036, 1989.

[21] V. N. Gudivada, V. V. Raghavan. Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity. *ACM Transactions on Information Systems*, 13, 2, 115-144, 1995.

[22] M. T. Goodrich, J. J. Tsay, D. E. Vengroff, J. S. Vitter. External Memory Computational Geometry. *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science*, pp.714-723, 1993.

[23] Venkat N. Gudivada, Vijay V. Raghavan. Content-Based Image Retrieval Systems. *IEEE Compute* 28, 9, Sep. 1995. pp.18-22.

[24] T. Huynh,C. Lassez, J. Lassez. Fourier Algorithm revisited. *Lecture notes in computer Science*, No.463, pp.117-131, 1990.

[25] H.V.Jagadish. Spatial Search with Polyhedra. *Sixth international conference on data engineering*, IEEE 1990. pp311-319

[26] H.V.Jagadish. A retrieval technique for similar shapes. *ACM SIGMOD*, 1991.

[27] J. Laffar,J. L. Lassez. Constraint Logic Programming: A Survey. *Journal of Logic Programming*, Vol.19&20, pp.503-581, 1994.

[28] A. Knapp, J. Anderson. Theory of Categorization Based on Distributed Mememory Storage. *Journal of experimental Psycologu: Learning , Memory, and Cognition*, 588-609, 1984.

[29] P.C. Kanellakis, D.Q. Goldin. Constraint Programming and Database Query Languages. *Proc. 2nd TACS*, 1994.

[30] M. Klein, A Primal Method for Minimal Cost Flows with Applications to the Assignment and transportation Problem, *Management Science*, 14, (1967) 205-220.

[31] P. Kanellakis, S. Ramaswamy, D. E. Vengroff, S. Vitter. Indexing for Data Models with Constraints and Classes. *Journal of Computer and System Sciences*, Vol.52, pp.589-612, 1996.

[32] Kuijpers,B.; Paredaens,J.; Vandeurzen,L., Semantics in Spatial Databases. *Lecture notes in Computer Science*, No. 1358, pp.114-135, 1998.

[33] Pradip Kanjamala, Peter Z. Revesz, Yonghui Wang. MLPQ/GIS: A GIS using Linear Constraint Databases. *Proc. Ninth International Conference on Management of Data*, pp. 389-392, Hyderabad, India, December 1998.

[34] P.C. Kanellakis, G.M.Kuper and P.Z.Revesz. Constraint Query Languages. *Journal of computer and system sciences*, vol. 51, pp.26-52, 1995.

[35] H. Katsuno, A.Mendelzon, Propositional Knowledge Base Revision and Minimal Change, *Artificial Intelligence*, 52, (1991) 263-294.

[36] H. W. Kuhn. The Hungarian Method for The Assignment Problem. *Naval Research Logistics Quarterly*, (2) 83-97, 1955.

[37] K. Kirtpatric-Steger, E. Wasserman, I. Biederman. Effects of Spatial Rearrangement of Object Components on Picture Recognition in Pigeons. *Journal of the Experimental Analysis of Behavior.* Vol 65, 465-475,1996.

[38] Lassez,J.-L., Querying Constraints. *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database systems*, pp.288-298, 1990.

[39] E.L.Lawler, *Combinatorial optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1976.

[40] Yuguo Liu. Animation of Spatio-Temporal Database. Master Thesis, Mar. 99. computer science department, UNL

[41] Rajiv Mehrotra, James E. Gary. Similar-shape retrieval in shape data management. *IEEE Compute.* 28, 9, Sep. 1995. pp. 57-62

[42] Paredaens,Jan; Bussche,Jan V.; Gucht,Dirk V. Towards a theory of Spatial Database Queries, *Proc. 13th ACM Symp. on Principles of Database Systems*, pp.297-288, 1994.

[43] F.P. Preparata, M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[44] Jan Paredaens, Bart Kuijpers. Genericity in spatial Databases. ???

[45] Jan Paredaens. Spatial Databases, The final Frontier. *Database Theory - ICDT'95*, LNCS No.893, Springer-Verlag

[46] C.H. Papadimitriou, K.Steiglitz, Combinatorial Optimization: Algorithms and Complexity, *Prentice-Hall Inc.*, 1982.

[47] Franco P. Preparata, Michael I. Shamos. Computational Geometry: an Introduction. Springer-Verlag Inc. New York, 1985.

[48] Peter Z.Revesz. Constraint Databases. Handouts for CSCE913 Adv. topics on Databases, spring'98

[49] H. Rafat, Z. Yang, D. Gauthier. Relational Spatial Topologies for History Geographic Information, *International Journal of GIS*, Vol.8, no.2, pp.163-173, 1994.

[50] P. Z. Revesz. Constraint Query Languages. Ph.D. Thesis. Brown University, 1991.

[51] Peter Z. Revesz. On the Semantics of Arbitration. *International Journal of Algebra and Computation*, 7, (1997) 133-160.

[52] Peter Z. Revesz. Model-Theoretic Minimal Change Operators for Constraint Databases. *International conference on Database Theory*, Jan, 1997.

[53] Peter Z. Revesz. Constraint Databases: A Survey. *Semantics in Databases*, L. Libkin and B. Thalheim, eds.,Springer-Verlag LNCS 1358, pp. 209-246, 1998.

[54] R. Ramaswamy. Efficient Indexing for Constraint and Temporal Databases. *Proc. 6th Int. Conf. on Database Theory*, 419–431, Springer-Verlag, LNCS 1186, 1997.

[55] S. Ramaswamy, S. Submaranian. Path Catching: A Technique for Optimal External Searching, *Proc. 13th ACM PODS*, pp.25-35, 1994.

[56] K. Satoh. Nonmonotonic Reasoning by Minimal Belief Revision. *Proc. Inter. Conf. on 5th Generation Computer systems*, Tokyo, 455-462, 1988.

[57] A. Silberschatz, H. F. Korth, S. Sudarshan. Database System Concepts,3rd edition. McGraw-Hill, 1996

[58] S. Submaranian, S. Ramaswamy. The P-range Tree: A New Data Structure for Range Searching in Secondary Memory. *Proc. 6th Annual ACM-SIAM Symposium on Discret Algorithms*, 1995.

[59] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* Addison-Wesley, Reading MA, 1990.

[60] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading MA, 1990.

[61] J. D. Ullman. Database and Knowledge-base Systems, Volume I: Classical Database Systems. Computer Science Press, 1988

[62] Luc Vandeuzen, Marc Gyssens, Dirk Van Gucht. On the Desirability and Limitations of Linear Spatial Database Models. *Lecture Notes in Computer Science*, No.951, 1995

[63] L. Vandeurzen, M. Gyssens, D. V. Gucht. An Expressive Language for Linear Spatial Database Queries. *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Seattle, WA., 1998.

[64] A. Weber. Updating Propositional Formulas. *Proc. First conf. on Expert Database systems*, 487-500, 1986.

[65] M. Winslett. Reasoning about Action Using a Possible Models Approach. *Proc. AAAI-88, st. Paul, MN*, 89-93, 1988.

[66] M. Zeiler. Inside ARC/INFO. OnWord Press, 1994.

[67] Sheng Zeng. Implementation of Model-Based Arbitration and Update Operators. Master Thesis, University of Nebraska, 1998.

List of Figures