A Declarative Spatial Query Processor for Geographic Information Systems

*Sudhakar Menon** and *Terence R. Smith*
Department of Geography and National Center for Geographic Information and Analysis, University of California at Santa Barbara, Santa Barbara, CA 93106

ABSTRACT: The design and implementation of a declarative GIS query processor capable of extracting the locations of complex objects from a spatial database is described. The processor extracts such objects from the database in a single, automated step. It also provides extensible support with respect to spatial operators and is independent of the particular spatial and attribute relationships being satisfied. The processor relies on the application of spatial operators that are stored in an extensible database and is guided both by symbolic definitions of the objects being sought and knowledge of the different operators. Search is based upon an efficient query processing algorithm, named forward constraint propagation, that integrates spatial constraint propagation, geometric search using hierarchical data structures, and an effective heuristic used in solving constraint satisfaction problems. Experimental results show that forward constraint propagation is superior to other constraint satisfaction algorithms for large domain sizes and high degrees of constraint. The nature of queries that may be satisfied is illustrated using an example query involving four sub-objects and ten binary spatial constraints.

INTRODUCTION

WE DESCRIBE THE DESIGN, implementation, and performance of a non-procedural query processor that automatically extracts instances of complex spatial objects from a spatial database given a *declarative* description of the object in terms of multiple spatial and attribute constraints. The principle features of the query processor described in this paper are

- Automated single step complex object search. The task of selecting, ordering, and applying a sequence of spatial operators to the data is performed by the query processor rather than the user.
- An extensible architecture that allows the smooth integration of new spatial operators into the query processing mechanism.
- Dynamic satisfaction of spatial constraints from databases storing only implicit topology.
- New algorithms for spatial search based on embedding geometric search within classical constraint satisfaction algorithms used in the fields of AI and Computer Vision.

The query processor described is designed to operate on both location-based and object-based spatial data. We use the term location-based to describe organizations of spatial data in which the primary entities represented are locations, with each location having a set of object properties. We use the term object-based to describe organizations of spatial data where the primary entities represented are objects, with each object having among its properties a spatial location. Location-based systems organize spatial data as a collection of themes or layers, such as surficial geology, soils, and elevation. Examples of location-based systems using raster data structures include IBIS (Bryant and Zobrist, 1976) and GRASS (C.E.R.L, 1988). Examples of location-based systems using topological vector data structures include ARC-INFO (Morehouse, 1985). We may characterize the bulk of recent efforts to store spatial data in relational database management systems using extended spatial operators and spatial access mechanisms as object-based (Abel and Smith, 1986; Guting, 1988).

The query processing mechanism described in this paper was developed and implemented as part of KBGIS-II, a GIS developed at the University of California, Santa Barbara that uses the frame-based paradigm of knowledge representation. KBGIS-II is part of an ongoing research effort in the area of GIS and spatial database design and implementation (Smith *et al.*, 1987). The paper is organized as follows: The next section provides an overview of the environment within which the query processor operates, including a description of the query language, the spatial operators available in the language, and the manner in which the spatial databases are organized. Following that, an architectural and algorithmic description of the query processing mechanism is provided, and strategies that are useful in optimizing the performance of spatial query processing are described. The processing of an example query using the query processing mechanism developed is then described. Finally, conclusions are presented.

THE QUERY PROCESSING ENVIRONMENT

The spatial query processor operates upon the user's description of the spatial phenomena of interest. The user models the phenomena in terms of a spatial object language (SOL) which is based on the first-order predicate calculus and supports both object-based and location-based descriptions. The models of spatial phenomena input by the user are stored as symbolic definitions in a Spatial Object Knowledge Base implemented in terms of frames. Each frame contains a symbolic description of a spatial object class. Each member of a spatial object class is required to satisfy a set of constraints specified by the user in the description for the class. Each member of a spatial object class has a location, which may be a polygon, line, or point. Examples of Spatial Object Classes that may be defined, given an appropriate spatial database, are *Potential Landslide Site* and *Groundwater Contamination Site.*

The spatial language provides three classes of properties that may be used to describe the members of a spatial object class. PPROPS are pixel properties that apply to each point or cell in space, GPROPS are group properties that characterize groups of cells comprising an object location, and RPROPS are relational properties describing relationships between two groups of cells or object locations. A more detailed description of the SOL may be found in Smith *et al.* (1987) and is included in the Appendix to help clarify the example query described later.

The Spatial Object Knowledge Base stores the definitions and other pertinent information on user-defined object classes. Each spatial object class is stored as a frame with variable length slots (fields) named the **SpatialObject** structure. The names and nature of the information stored in each slot of the structure are

*Presently with Environmental Systems Research Institute, Redlands, CA 92373.

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 55, No. 11, November, 1989, pp. 1593-1600.

0099-1112/89/5511–1593$02.25/0

shown in Table 1. The knowledge base is extensible and an editor allows users to add, modify, delete, and browse through the objects in the Knowledge Base. The actual spatial locations corresponding to objects in the knowledge base are held either implicitly or explicitly in different spatial data bases.

The spatial object language is similar to languages based on the tuple relational calculus and is declarative (or non-procedural) in nature. It does not specify the sequence of operations that must be applied in order to retrieve instances of a spatial object class from a database. The language currently lacks universal and existential quantifiers. For a discussion of the limitations on the representation imposed by the lack of these quantifiers, the reader is referred to Menon (1989).

Spatial Operators

The GPROP and RPROP constraints specified in the SOL are enforced by the application of spatial operators. A spatial operator is a function that takes as operands geometric entities: points (*P*) lines (*L*), or regions (*R*). The result returned by a spatial operator may be a numeric value (*N*), a boolean value (*B*), or another geometric entity (*P,L,R*). There is a clear separation in the language between operators that return boolean or numeric values and operators that return new geometric entities. GPROP and RPROP spatial operators may return **only** numeric or boolean values and take as operands object entities of type point, line or region.

The operands of GPROP and RPROP spatial operators are obtained by

- searching for and aggregating cells in a location-based spatial database organized as raster or quadtree data structures with the application of operators for connected component labeling (Samet and Tamminnen, 1985; Menon and Smith, 1988) and line following in order to derive object based representations (lines, regions), or
- directly searching object-based spatial databases for lines and regions.

Table. 2 shows the set of operators that may be used to define GPROPS and RPROPS in KBGIS-II. Spatial operators such as UNION and INTERSECTION return geometric entities and will be referred to as geometric operators. A set of such operators is described in Claire and Guptill (1984) and is shown in Table 3.

Locations corresponding to the results of UNION and INTERSECTION operations between thematic layers may be described in the SOL using conjunctions and disjunctions of PPROP predicates. A location that is the result of a COMPLEMENT operation may be described in terms of a PPROP predicate that uses the complement of the set of attribute values. Locations that are the result of a DIFFERENCE operation may be described in terms of the results of COMPLEMENT and INTERSECTION operations. The actual geometric operations are performed dynamically by the query processing mechanism during search for the defined spatial object in the location-based spatial database. The use of hierarchical, quadtree data structures in the KBGIS-II location-based spatial database permits these operations to be performed efficiently. Geometric operators that take object-based region and line operands can also be used within the spatial object language as unary and binary functions.

A Function Knowledge Base is used to store information on different spatial operators. The query processing mechanism makes active use of the information in the Function Knowledge Base during the process of satisfying the constraints specified in the description of an object class. Information on each operator is stored in a frame which has the structure shown in Table 4. This knowledge base is extensible and the user may add new operators to the system. The addition of an operator requires building a procedure that implements the underlying operation,

TABLE 1. THE SPATIAL OBJECT STRUCTURE

Slot	Contents
DefinedBy	The definition of the object : an expression in the Spatial Object Language
Defines	Names of objects defined using this object as a subobject
FeatureType	*point,region,linear*, or *hybrid*
Heuristics	An estimate of the number of examples of the object in the database
Complexity	A measure of the complexity of search for new examples of the object
Size	An estimate of the linear and areal dimensions of the object
Geom-Operator	Name of geometric operator, if any.
Procedures	Name of special purpose search function, if any.

TABLE 2. SPATIAL OPERATORS

Operator	Type	Operand1	Operand2	Result
SIZE-R	GPROP	Region	–	numeric
ECCENTRICITY-R	GPROP	Region	–	numeric
ORIENTATION-R	GPROP	Region	–	numeric
PERIMETER-R	GPROP	Region	–	numeric
LENGTH-L	GPROP	Line	–	numeric
DISTANCE-RR	RPROP	Region	Region	numeric
DISTANCE-LL	RPROP	Line	Line	numeric
DISTANCE-RL	RPROP	Region	Line	numeric
DIRECTION-RR	RPROP	Region	Region	numeric
DIRECTION-RL	RPROP	Region	Line	numeric
DIRECTION-LL	RPROP	Line	Line	numeric
CONTAINS-RR	RPROP	Region	Region	boolean
CONTAINS-RL	RPROP	Region	Line	boolean
OVERLAPS-RR	RPROP	Region	Region	boolean
OVERLAPS-RL	RPROP	Region	Line	boolean
INTERSECTS-LL	RPROP	Line	Line	boolean

TABLE 3. GEOMETRIC OPERATORS

Operator	Operand1	Operand2	Result
BOUNDARY-R	Region	–	Line
COMPLEMENT-R	Region	–	Region
UPPER-END-L	Line	–	Point
LOWER-END-L	Line	–	Point
INTERSECTION-RR	Region	Region	**Region**, Line, Point
INTERSECTION-RL	Region	Line	**Line**, Point
INTERSECTION-LL	Line	Line	**Point**, Line
UNION-RR	Region	Region	Region
UNION-LL	Line	Line	Line
DILATE-R	Region	Numeric	Region
SHRINK-R	Region	Numeric	Region

TABLE 4. THE **FUNCTION** STRUCTURE

Slot	Contents
Propagation	Indicates if the function can be inverted to propagate constraints
Complexity	An index of the relative computational complexity of the function
Symmetry	Indicates if a binary function (RPROP) is symmetric
Range	Boolean, Numeric

and its declaration to the system by means of the addition of a frame to the Function Knowledge Base. The query interface remains non-procedural at all times, the processing mechanism being responsible for the application of all spatial operators during query processing.

Several operators may be required to implement a single RPROP. As an example, three operators are required to implement the concept of DISTANCE. The function DISTANCE-RR is used to compute the distance between two region features, the function DISTANCE-LL computes the distance between two linear features, and the function DISTANCE-RL computes the distance between a region feature and a linear feature. Interface routines to the spatial object knowledge base and to the query processor determine the correct functions to be used and modify the query or definition that is input by the user. The precise geometric definition associated with a particular function (e.g., the minimum distance between the boundaries of two regions or the distance between centroids) may be modified by defining and replacing operators in the Function Knowledge Base.

Spatial Database Organization

The spatial data in the system are stored in two databases: A Location-Based Spatial Database and an Object-Based Spatial Database. These databases are queried for instances of a Spatial Object Class during Query Processing.

The Location-Based Spatial Database is organized as a series of layers or themes, such as Geology, Landuse, and Slope. Each layer is stored as a hierarchical quadtree that explicitly represents both leaf and internal nodes. Peng *et al.* (1987) describe space efficient data structures for managing hierarchical quadtrees on external storage. Menon *et al.* (1988) describe the use of Bit Mapped Multi-Colored quadtrees for the rapid overlay of discrete categorical layers such as Landuse and Geology. The use of hierarchical quadtrees in the Location-Based Spatial Database allows for the rapid overlay of layers in the database during query processing and also permits rapid windowing and selection of areas that possess desired attributes.

The Object-Based Spatial Database is organized by Object. An Object possesses attributes such as NAME, AREA, and LOCATION. Object Locations can be represented as polygons, lines, or points and can require the use of a variable length field. The examples of each Object are spatially indexed using a hierarchical quadtree key based on the minimum containing rectangle for a tuple (Abel and Smith, 1983, 1986).

THE QUERY PROCESSING MECHANISM

The query processing mechanism is independent of particular spatial relationships and object definitions and is based upon the use of function application, recursion, and spatial constraint propagation.

The principal data structure used by the search procedures responsible for assembling together the locations of a multi-component object is a *semantic network* (Nilsson 1980). Such a structure represents objects and relationships between objects as a labelled graph. A directed arc with name L and value V between nodes X and Y signifies that the relationship $L(X, Y) = V$ holds.

During spatial search, the nodes of the network serve as a temporary database for locations of the corresponding objects that are retrieved from the spatial database. The network representation is then used in the process of spatial constraint satisfaction and guides the selection of spatial operators that must be applied to check the constraints imposed in the query. The network representation is also used in the propagation of spatial constraints to narrow down the area of the spatial database in which a sub-object is sought, thus aiding in geometric inferencing. A node in the semantic network is represented in terms of a structure with variable length slots (fields) named the **SubObject** structure. An arc in the network is represented by a structure named the **Relation** structure. Table 5. shows the slots in the **SubObject** structure. Table 6. shows the slots in the **Relation** structure used to represent arcs in the network.

A query processor that can support both object- and location-based organizations of spatial data requires the ability to overlay multiple thematic layers dynamically during query processing and the ability to process spatial relationships between multiple objects dynamically during query processing. A flexible GIS cannot rely on the explicit representation and storage of every spatial relation that may be of potential significance to a user. A spatial query processing mechanism for a GIS must therefore be capable of handling implicit spatial information in a flexible and automated manner. If the spatial database uses raster structures, then both the connectivity information pertaining to a single object and the topologic and metric information on the spatial relationships between objects are only implicitly represented in the data structure. If the database is organized using topologic vector structures, then connectivity and adjacency relationships are represented explicitly but metric spatial relationships between objects are only represented implicitly.

The Multi-Component Spatial Search Problem

The task faced by the query processor, when asked to retrieve all members of an object class from a large spatial database when given a symbolic description of the class, is the solution of an instance of the multi-component spatial search problem (MCSSP).

Viewed in terms of the relational data model, each sub-object in a multi-component object corresponds to a sub-object relation, the columns of which are the object attributes and the object location. The location entry for each tuple in the relation is a

TABLE 5. THE **SUBOBJECT** STRUCTURE

Slot	Contents
Type	The name of the corresponding Spatial Object
SubObjects	A list of pointers to nodes (**SubObject** Structures) representing the components of the corresponding Spatial Object
Pprops	The set of PPROPS for this node (if any)
Gprops	The set of GPROPS for this node (if any)
Relations	Pointers to the set of **Relations** in which the node is involved
ExampleLoc	Space for locations of examples found for this node that satisfy all imposed unary (PPROP and GPROP) constraints.
CurLoc	The current location of this node that is being investigated by the constraint satisfaction process
TempLoc	Temporary storage space for use by constraint satisfaction procedures
Solutions	Space for locations of examples found for this node that satisfy all imposed unary and binary (RPROP) constraints

TABLE 6. THE **RELATION** STRUCTURE

Slot	Contents
Type	Name of the corresponding RPROP spatial operator
Arg1	Pointer to first RPROP argument
Arg2	Pointer to second RPROP argument
Value	Specified Value for the RPROP

point, line, or polygon. Solution of the query results in an output relation that is a subset of the Cartesian Product set of the input sub-object relations. Selection of a tuple for inclusion in the output relation requires that all unary and binary spatial constraints be satisfied.

We now present a more formal description of the MCSSP. In a multi-component spatial search problem for a spatial object O with n sub-objects $0_1,\ldots, 0_n$, the sub-objects must satisfy specified unary constraints (such as size and shape) and binary constraints (such as distance and direction). Let $v_1, v_2,\ldots, v_n$ be variables characterizing respectively each of the n subobjects and taking values from a finite set of locations, D_i. The location of any sub-object is a point, a piecewise linear curve, or a connected region in the plane whose boundary is a polygon.

Let the predicate P_i represent the conjunction of all k unary constraints on the location variable v_i ($P_i = P_{i1} \wedge P_{i2} \ldots \wedge P_{ik}$) and let the predicate P_{ij} represent the conjunction of all binary constraints on the location variables v_i and v_j. A *location-set* for the object O is then a set of n locations $\{x_1,x_2, \ldots, x_n\}$ with $x_i \epsilon D_i$ for $i=1, \ldots n$ such that

$$P_1(x_1) \wedge \ldots \wedge P_n(x_n) \wedge P_{12}(x_1,x_2) \wedge \ldots \wedge P_{n-1,n}(x_{n-1}x_n).$$

The MCSSP consists of determining all location-sets for the object O, given the domains of the n sub-objects. The MCSSP is a constraint satisfaction problem (CSP) in which the constraints between variables are *spatial* in nature and the values taken by variables are *locations* in space. It is easy to show that if no restrictions are placed on the spatial constraints, then the general MCSSP is NP-Complete (Menon and Smith, 1987).

The Spatial Query Processing Algorithm

The unique feature of the spatial search problem in GIS (in comparison to the general constraint satisfaction problem) is that the locations that must be selected to satisfy the spatial constraints are *already* stored in a spatially referenced format. Hence, inversion of constraints suggests itself as a natural way to achieve constraint satisfaction.

Backtracking is the basic tree search algorithm for the general constraint satisfaction problem, and corresponds to a depth first search of the variable domains. The performance of backtracking may be improved by the use of consistency algorithms either as a preprocessing step (see Mackworth (1977) and Mackworth and Freuder (1985) who describe node, arc, and path consistency algorithms) or during tree search, in which case Forward Checking (FC), Partial Looking Ahead, and Backmarking (Haralick and Elliott, 1980) are three algorithms that dynamically enforce arc consistency conditions. In these algorithms for the general CSP, the domain of each variable is represented as an ordered set. During tree search, the values for any variable are examined sequentially and constraints are explicitly computed in order to check whether the value selected from the domain satisfies the constraints on the variable.

Unlike these variants of the CSP algorithms, the query processing mechanism developed implements a new algorithm developed for MCSSP, called the Forward Constraint Propagation algorithm (FCP). The FCP algorithm exploits the spatial nature of both the domain and constraints by using appropriate spatial data structures and retrieval algorithms. FCP inverts spatial constraints and embeds spatial constraint propagation within the Forward Checking heuristic tree search algorithm developed for the general CSP. An algorithmic description of FCP, including in embedding in a spatial database using hierarchical quadtree data structures, may be found in Menon (1989). Spatial constraint propagation is used to replace the explicit checking of constraints during backtracking by geometric search within constrained areas of the database. Depending on the geometric nature of object locations (points, lines, regions) and of the propagated constraints, the spatial window in the database in which search occurs may have an arbitrarily complex shape, which may be difficult to search efficiently. It may then be necessary to search within a larger area which is more efficiently accessed under the geometric data structure selected, and apply *filtering search* with explicit constraint checking being used to filter out retrieved examples that do not satisfy the required constraints.

Inversion of constraints is an efficient approach when there are only two sub-objects involved in a relationship. The replacement of explicit constraint checking by constraint propagation is, however, insufficient to prevent "thrashing" during backtracking when the number of sub-objects in a problem exceeds two. FCP therefore incorporates the "looking into the future" heuristic used by the FC algorithm of Haralick and Elliott (1980). The forward checks that enforce arc consistency between a current variable and future variables in the FC algorithm are replaced by forward constraint propagations that result in a search for future variables within constrained areas of the database in FCP. In order to apply spatial constraint propagation, it is necessary to decide on a search order for the sub-objects. Order criteria include domain size (smallest first), the structure of the underlying constraint graph (e.g., minimum width ordering (Freuder, 1982)), and object complexity (simplest first). Spatial Constraint Propagation requires the use of data structures that permit efficient retrieval of the subset of locations within a constrained window from the total set of locations in the database. Examples of such structures include quadtrees (Samet *et al.*, 1984), R-trees (Guttman, 1989), and Cell trees (Gunther, 1987). The efficiency of this retrieval process depends on the specific data structures used for spatial indexing and storage of component locations within the spatial database.

Experimental Comparison of Search Algorithms

An experimental evaluation of FCP and other Algorithms for the MCSSP was carried out using procedures embedded in the spatial search module of KBGIS-II. The algorithms were tested on a database provided by the US Geological Survey, organized using Bit-Mapped MultiColored Quadtrees (Menon *et al.*, 1988). Nine algorithms were compared experimentally on a set of multi-component spatial search problems: Backtracking, Unidirectional Arc-Consistency followed by Backtracking, Arc-Consistency followed by Backtracking, Forward Checking, Unidirectional Arc-Consistency followed by Forward Checking, Arc-Consistency followed by Forward Checking, Partial Looking Ahead, Backmarking, and FCP. Of the above algorithms, only FCP explicitly takes the spatial nature of the problem into account. FCP and FC were compared to determine the utility of spatial constraint propagation. Each algorithm was investigated on the same set of multi-component spatial objects, consisting of up to five region and linear sub-objects related by distance constraints. The following measures were made during each test run in order to investigate the time efficiency of the different procedures:

- The total number of consistency checks used to verify spatial relationships between different sub-objects;
- the total number of computations of constrained search windows;
- the total number of database searches for single component sub-objects;
- the total area of the spatial database searched; and
- the total CPU time taken by the system to answer the complete query.

The multi-component object models corresponding to the test queries were characterized by the following parameters:

- The Degree Of Constraint (DOC) imposed by tghe spatial relations in the model. Three values of this parameter were used (HIGH, MEDIUM, LOW).
- The Constraint Graph Topology (CGT) for the model. CLIQUE and STAR topologies were investigated.

Of the eight constraint checking algorithms investigated, Forward Checking used the smallest number of consistency checks for all test cases. FC was also the fastest algorithm, in terms of CPU time (confirming results obtained by Haralick (1981) for the 8-Queens problem). Forward Constraint Propagation (FCP) was superior to Forward Checking for queries with a high degree of constraint, and for queries which involve costly constraint checking operators. The number of constraint propagations made by FCP is a small fraction of the number of constraint checks made by FC. This fraction grows smaller as domain sizes increase, but is independent of the number of sub-objects. A more detailed discussion of experimental parameters and results may be found in Menon (1989).

QUERY PROCESSING WITH MULTIPLE SPATIAL CONSTRAINTS: AN EXAMPLE

This section describes a query used to test the ability of the query processor implemented in KBGIS-II to handle queries that involve multiple objects, multiple spatial relations, and more than one kind of spatial relation between two objects. The symbolic definition of the query in the Spatial Object Language is

```
(AND
((GEOL O__1)[DEADWOOD__SANDSTONE])
((SIZE O__1) [5 200])
((TYPE O__2)RIDGE)
((GEOL O__3) [ENGLEWOOD__LIMESTONE ])
((SIZE O__3) [5 5000])
(AND
((LAND O__4) [EVERGREEN]))
((ASPECT O__4) [0 60])
)
((SIZE O__4) [10 5000])
((DISTANCE O__1 O__3) [1 20])
((DISTANCE O__1 O__4) [1 20])
((DISTANCE O__3 O__4) [1 20])
((>(SIZE O__1) (SIZE O__4)) [TRUE])
((>(SIZE O__4) (SIZE O__3)) [TRUE])
((DIRECTION O__1 O__3) [NE E SE])
((DIRECTION O__1 O__4) [NE E SE])
((DISTANCE O__1 O__2) [1 20])
((DISTANCE O__3 O__2) [1 20])
((DISTANCE O__4 O__2) [1 20])
)
```

Figure 1 shows the semantic network representation of the query object. Sub-Object O__1 represents connected regions where the Geology is Deadwood Sandstone and that are between 5 and 200 units in size (area). Sub-Object O__2 represents linear features that are ridge lines. Sub-Object O__3 represents connected regions where the Geology is Englewood Limestone and that are between 5 and 5000 units in size (area). Sub-object O__4 is described in terms of the intersection of the Landuse and Aspect layers of the database and corresponds to regions of evergreen forests located on slopes with aspects between 0 and 60 degrees east of north, with sizes between 5 and 10000. A total of ten binary constraints are imposed between the four sub-objects.

If the query processing mechanism uses no spatial constraint propagation, then solution of the above query begins with the extraction of the locations of the individual region and linear sub-objects from the spatial databases. All unary constraints are satisfied as part of this extraction process. This phase is very fast and uses quadtree overlay and connected component labeling algorithms. The location of each connected region that satisfies all unary constraints is approximated using an oriented rectangular approximation and returned to a high level spatial constraint satisfaction processor. The relation between the oriented rectangular approximation of a region and its true shape is shown in Figures 2a and 2b. The spatial distributions of the examples of sub-objects O__1, O__2, O__3, and O__4 that satisfy all unary constraints are shown in Figures 3a, b, c, and d, respectively, using oriented rectangular approximations to represent the regions.

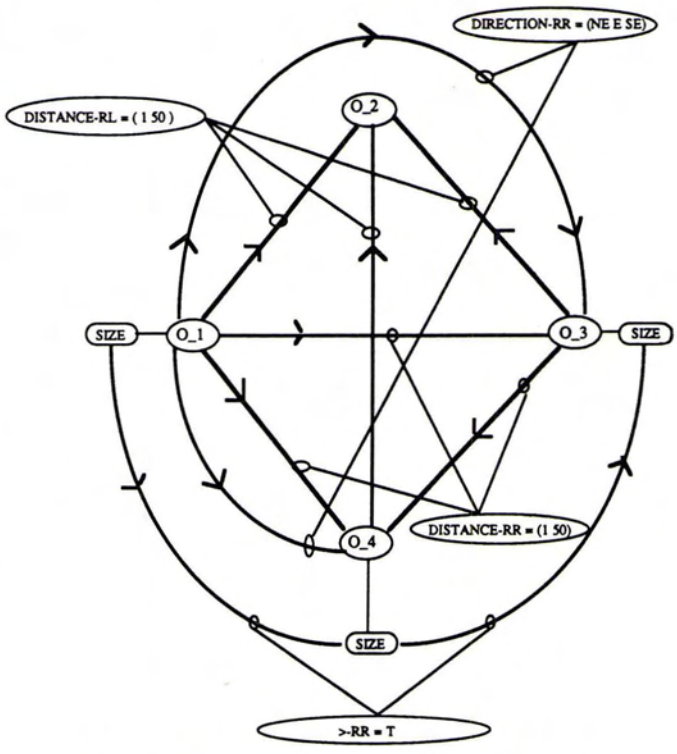


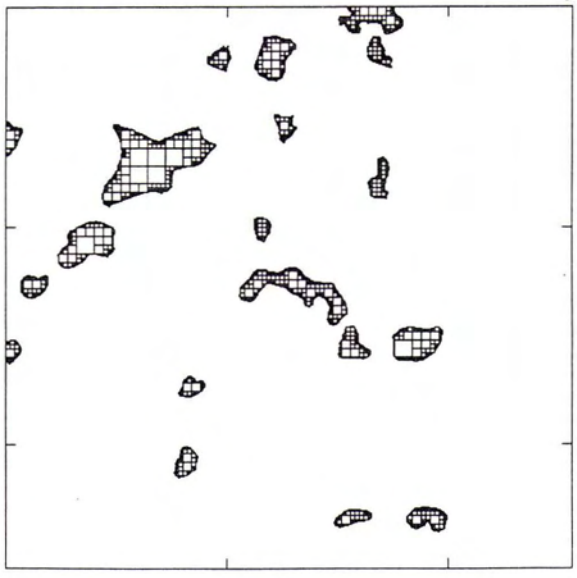
FIG. 1. Semantic Network corresponding to Test Query.

The next phase involves the application of the FC spatial constraint checking algorithm to the sub-object locations extracted in the first phase and is computationally more intensive. The size of the Cartesian product set of the sub-object domains for the above query, after rejecting locations that do not satisfy the unary constraints, is 6.25×10^8. The number of tuples from this set that satisfy all binary constraints is 12. Constraint checking operators for distance and direction were point-based and constraints were computed using the centroids of object locations.

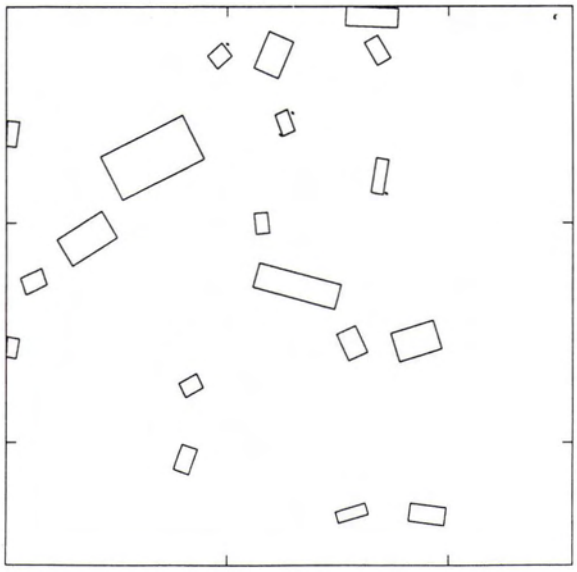
If the query processing mechanism uses spatial constraint propagation, then the processes of object location extraction and constraint satisfaction are interleaved, using the FCP algorithm. Spatial search and overlay take place within reduced sections of the database, resulting in faster query execution.

The use of more accurate constraint checking operators, based on the minimum distance between polygons and polylines and the directions between polygons, is to increase the cost of the FC algorithm by a factor that grows linearly with the cost of the checking operator. The rate of growth in the cost of the FCP algorithm is smaller because the FCP algorithm makes only a small fraction of the number of checks made by the FC algorithm (checks are restricted to filtering retrieved examples). Experiments showed this factor to range from between 0.1 and 0.25 at high degrees of constraint.

For fixed spatial operators, the overall time taken to compute the query increases when the degree of constraint is decreased. The degree of constraint was decreased by increasing the upper values specified in the distance constraints from 20 to 50. The number of tuples satisfying all constraints increased from 12 to 523. The cost of finding these tuples using the FC algorithm



(a)



(b)

FIG. 2. (a) Sub-Object Locations represented using Linear Quadtrees. (b) Sub-Object Locations represented using Oriented Rectangles.

increased by a factor of (approximately) 5. The FC and the FCP algorithm require roughly the same computational resources because the value of constraint propagation diminishes as the degree of constraint in the query decreases.

Plate la shows the location of all four sub-objects color-coded and overlaid on the same display surface. Plate 1b shows the locations of the four sub-objects that participate in the 523 tuples that correspond to the locations of the multi-component object sought for in the problem. Note that an individual sub-object location may participate in more than one output tuple. In relational terms, once these tuples have been selected, it may be desirable to project them in order to obtain attributes of interest. For example, in the above query it may be only the owners of the selected forest regions that are of interest.

CONCLUSIONS

In this paper we have described the design of an effective and efficient query processing mechanism for use in GIS. The query proccessing mechanism operates in an environment that integrates both object- and location-based organizations of spatial data using hierarchical data structures, and supports the dynamic enforcement of user specified spatial relationships.

The task of assembling sets of locations from a map or image that satisfy multiple imposed attribute and spatial constraints is a difficult computational problem. As a result, most GIS do not support the automated single-step retrieval of information based on multiple, implicitly-stored spatial relationships and attributes. The complexity of the MCSSP derives from the nature of spatial constraints, and the requirement that many of these spatial constraints be computed and verified "on the fly." Unextended implementations of the relational model are thus not suitable for such queries. To solve the MCSSP efficiently using relational systems requires an extension of the relational algebra to include geometric data types and operators and an implementation of the extended algebra that integrates geometric algorithms and data structures (Guting, 1988). Reasonable performance cannot be obtained unless geometric algorithms and data structures are thoroughly integrated into the query optimization and query processing units of relational systems. Query optimization in such an extended relational system requires insight into geometric algorithms and the ability to efficiently exploit spatial constraints in an efficient manner.

The research described in this paper has resulted in a query processor that automates the proccessing of declarative spatial queries, that is extensible with respect to the operators used to enforce spatial constraints, and that can dynamically verify implicitly stored spatial relationships. The query processor developed can operate on both location-based and object-based data organizations and is based on the use of hierarchical data structures, geometric search, and heuristic constraint satisfaction algorithms.

Current developments of the system (now called KBGIS III) include the development of a natural language front-end; the augmentation of the query processor, particularly with the use of knowledge-based techniques for query optimization; an extension of the system's inferential learning capabilities; and the implementation of efficient file structures for organizing the known instances of spatial objects.

REFERENCES

Abel, D. J., and J. L. Smith, 1983. A Data Structure and Algorithm Based on a Linear Key for Rectangle Retrieval, *Computer Vision, Graphics and Image Processing*, Vol. 24: No. 1, pp. 1–13.

———, 1986. A Relational GIS accommodating Independent Partitionings of the Region, *Proceedings of the Second International Symposium on Spatial Data Handling*, Seattle, Washington, pp. 213–224.

Bryant, N. A., and A. L. Zobrist, 1976. IBIS: A Geographic Information System Based on Digital Image Processing and Raster Data Type, *Symposium on Machine Processing of Remotely Sensed Data*, Purdue University.

C.E.R.L., 1988 *GRASS Users Manual, Version 3.0*, Corps of Engineers Research Laboratory, Champaign, Illinois.

Claire, R. W., 1984. Algorithm Development for Spatial Operators, *IEEE Pecora 9 Symposium*, Sioux Falls, SD, USA, 213–221. Seattle,Washington, pp. 5–14.

Freuder, E. C., 1982. A Sufficient Condition for Backtrack Free Search, *Journal of the ACM*, Vol. 29: pp. 24–32.

Garey, M. R., and D. S. Johnson, 1979. *Computers and Intractability - A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., New York.

Gunther, O., 1987. *Efficient Data Structures for Geometric Data Management*, PhD Dissertation, Memorandum UCB/ERL M87/77, Electronics

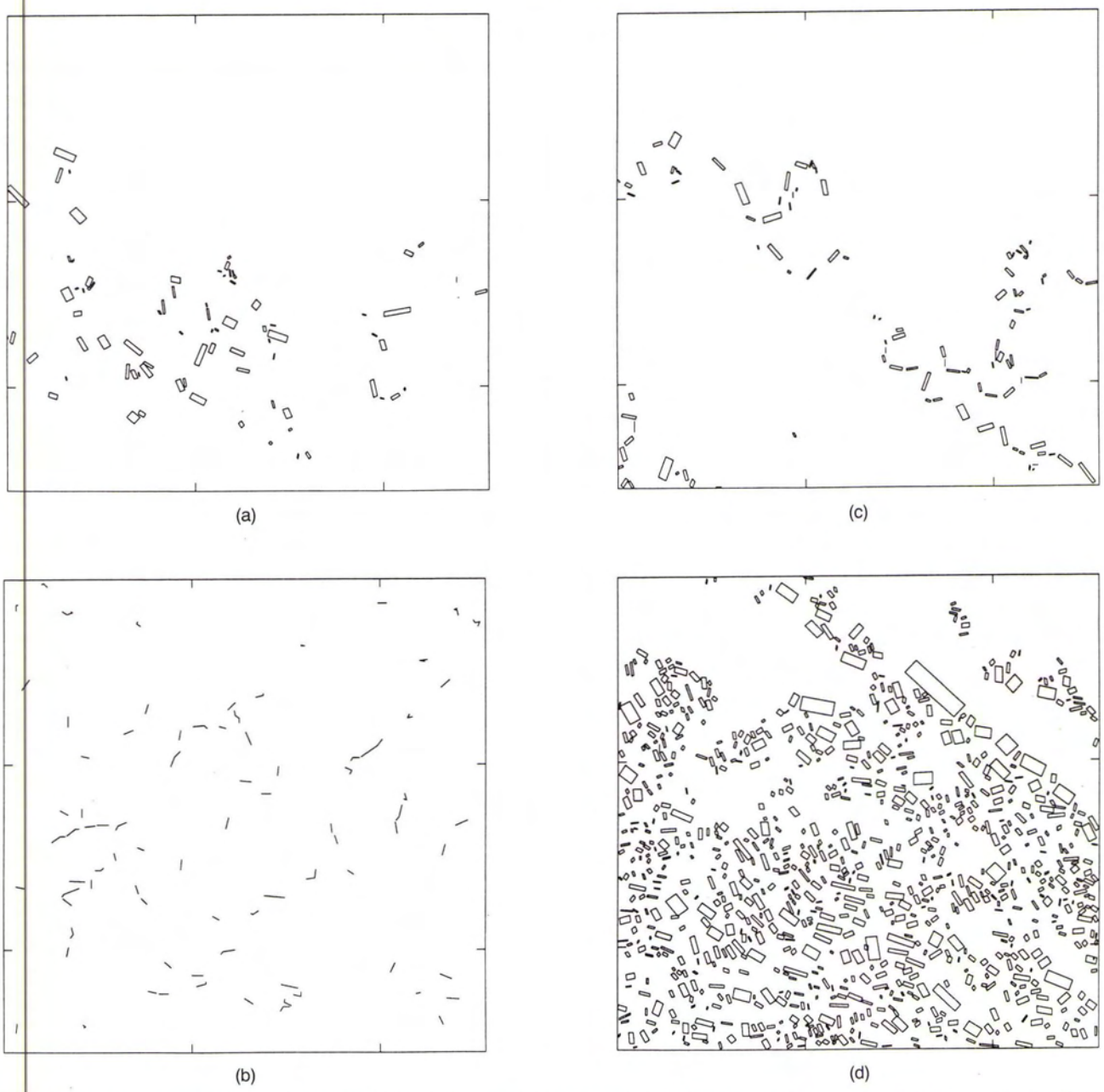


FIG. 3. (a) Rectangles corresponding to Sub-Object O__1 (Deadwood Sandstone). (b) Polylines corresponding to Sub-Object O__2 (Ridge). (c) Rectangles corresponding to Sub-Object O__3 (Englewood Limestone). (d) Rectangles corresponding to Sub-Object O__4 (North East Conifer).

Research Laboratory, College of Engineering, University of California, Berkeley, California.

Guptill, S. C., R. G. Feageas, and M. A. Domaratz, 1988. Designing an Enhanced Digital Line Graph, *ACSM-ASPRS Technical Papers*, American Congress on Surveying and Mapping, Vol. 12, pp. 252–611.

Guting, R. H., 1988. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems, *Proceedings International Conference on Extending Database Technology*, Venice, Italy, Springer-Verlag, pp. 506–527.

Guttman, A., 1984. R trees: A Dynamic Index Structure for Spatial Searching, *Proceedings of the ACM SIGMOD Conference on Management of Data*, Boston, Massachusetts.

Haralick, R. M., and Elliott, G. L., 1980. Increasing Tree Search Efficiency for Constraint Satisfaction Problems, *Artificial Intelligence*, Vol. 14, pp. 263–313.

Mackworth, A. K., 1977. Consistency in Networks of Relations, *Artificial Intelligence*, Vol. 8, pp. 99–118.

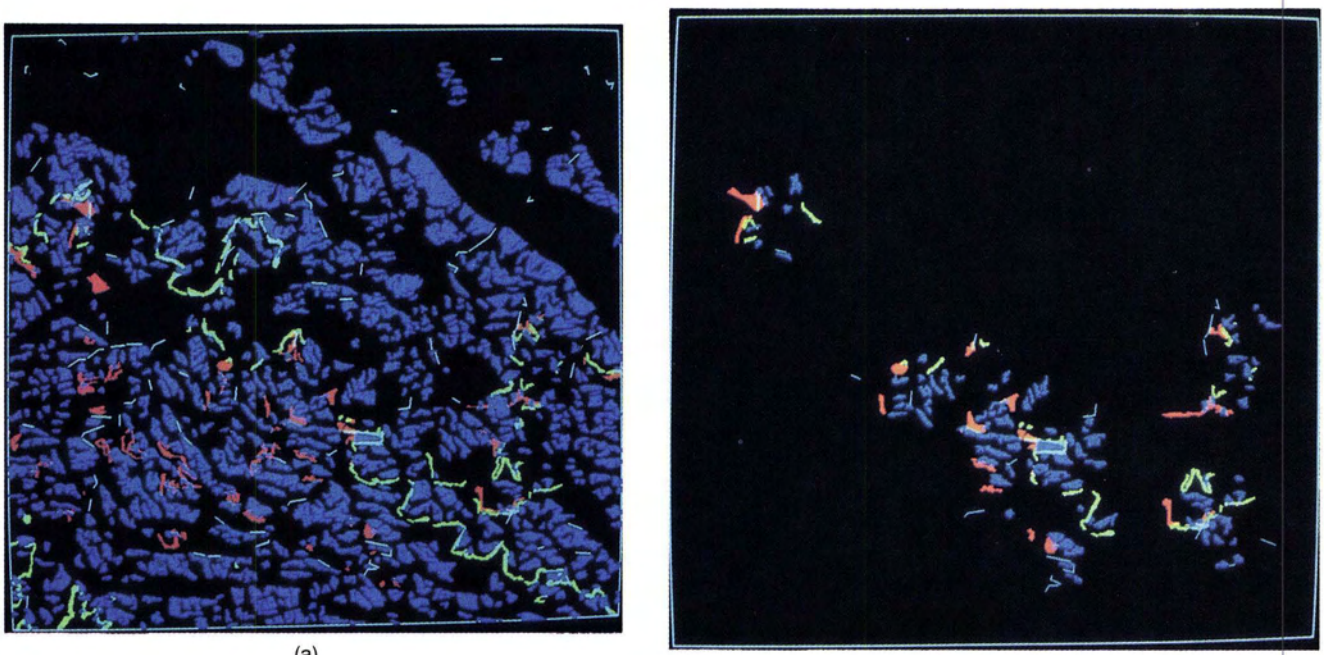
Mackworth, A. K., and E. C. Freuder, 1985. The Complexity of some Polynomial Network Constraint Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence*, Vol. 25, pp. 65–74.

Menon, S., 1989. *Spatial Search for Multi-Component Objects in a Geographic Information System Using Symbolic Models and Hierarchical Data Structures*, PhD Dissertation, University of California, Santa Barbara.

Menon, S., G. Peng, and T. R. Smith, 1988. Multi-Colored Quadtrees for GIS: Exploiting Bit-Parallelism for Rapid Boolean Overlay, *Pattern Recognition Letters*, Vol. 8, pp. 171–170.

Menon S., and T. R. Smith, 1987. Multi-Component Object Search Using Spatial Constraint Propagation, *International Geographic Information Systems Symposium*, Association of American Geographers, Arlington, Virginia.

———, 1988. A Boundary Matching Algorithm for Connected Component



(a) (b)

PLATE 1. (a) Color-Coded and Overlaid Sub-Object Locations. (b) Sub-Object Locations participating in Solution Tuples.

Labeling Using Linear Quadtrees, *Image and Vision Computing*, Vol. 6, No. 4, pp. 215–224.

Morehouse, S, 1985. Arc-Info: A Geo-Relational Model for Spatial Information *Proceedings: Auto-Carto 7*, Washington, D.C., pp. 388–397.

Nilsson, N., 1980. *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, California.

Peng, G., and Smith, T. R., 1987. Space Efficient Hierarchical Structures: Relatively Addressed Compact Quadtrees for GIS, *International Geographic Information Systems Symposium*, Association of American Geographers, Arlington, Virginia.

Samet, H., 1984. The Quadtree and Related Hierarchical Data Structures, *ACM Computing Surveys*, Vol. 16, pp. 187–260.

Samet, H., and M. Tamminen, 1985. Computing Geometric Properties of Images Represented by Linear Quadtrees, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-7, Vol. 2, pp. 229–240.

Smith, T. R., D. J. Peuquet, S. Menon, and P. Agarwal, 1987. KBGIS-II: A Knowledge-Based Geographic Information System, *International Journal of Geographical Information Systems*, Vol. 1, No. 2, pp. 149–172.

Tomlin, C. D., 1983. *Digital Cartographic Modelling Techniques in Environmental Planning*, PhD Dissertation, Yale University, Connecticut.

Winston, P. H., 1977. *Artificial Intelligence*, Addison Wesley, New York.

APPENDIX

The spatial object language (SOL) provides three classes of properties that may be used to describe the members of a spatial object class:

- Pixel properties, PPROPS, are properties that apply to each point in space. Each spatial variable or layer in the database is associated with a PPROP. PPROPs are used to support a location based view of the data. Examples of PPROPs are elevation, aspect, and landuse. In a tesellation-based database PPROPs characterize individual cells in the database.
- Group properties, or GPROPs, are properties that characterize the locations of members of a spatial object class, but that do not apply to the individual points or cells that constitute the location. Examples of GPROPS are AREA, LENGTH, and PARCEL-ID. In a tesellated database a GPROP is a property of the groups of cells that comprise the location of some member of an object class, and not a property of each cell. Implicit in the definition of a GPROP is the aggregation of cells into geometric entities (polygons or lines) where each geometric entity corresponds to the location of some member of an object class. GPROPS are used to support an object-based view of the data.
- Relational properties, or RPROPs, are properties that describe binary relationships between two spatial objects. The relation may be on the locations of the two spatial objects or on the attributes (GPROPS) of the two spatial objects. Examples of RPROPs include DISTANCE, DIRECTION, and GREATER-THAN. RPROPS are used to support an object based view of the data.

A spatial object class is described using a conjunction of properties that characterize its members. An example of a spatial object description is as follows:

```
(AND
((LANDUSE O1) [DECIDUOUS CONIFER])
((SIZE O1)   [10 1000])
((TYPE O2)      PERENNIAL-STREAM)
((LENGTH O2) [5 100])
((DISTANCE O1 O2) [1 20])
)
```

The above example describes an object in terms of two sub-objects and their spatial relationships. The first sub-object is described in terms of the pixel property LANDUSE and the group property SIZE. The second object is described using the group properties TYPE and LENGTH. The relational property DISTANCE is used to express the desired spatial relation. The GPROP TYPE permits reference to members of other spatial object classes that have been defined in the knowledge base. The object PERENNIAL-STREAM thus corresponds to a separate frame in the knowledge base with its own definition.