Organization of Data in Non-Convex Spatial Domains

Eric Perlman[1], Randal Burns[1], Michael Kazhdan[1],
Rebecca R. Murphy[2], William P. Ball[2], and Nina Amenta[3]

[1] Dept. of Computer Science, Johns Hopkins University
{eric,randal,misha}@cs.jhu.edu
[2] Dept. of Geography and Environmental Engineering, Johns Hopkins University
{rebecca.murphy,bball}@jhu.edu
[3] Dept. of Computer Science, University of California, Davis
amenta@cs.ucdavis.edu

Abstract. We present a technique for organizing data in spatial databases with non-convex domains based on an automatic characterization using the medial-axis transform (MAT). We define a tree based on the MAT and enumerate its branches to partition space and define a linear order on the partitions. This ordering clusters data in a manner that respects the complex shape of the domain. The ordering has the property that all data down any branch of the medial axis, regardless of the geometry of the sub-region, are contiguous on disk. Using this data organization technique, we build a system to provide efficient data discovery and analysis of the observational and model data sets of the Chesapeake Bay Environmental Observatory (CBEO). On typical CBEO workloads in which scientists query contiguous substructures of the Chesapeake Bay, we improve query processing performance by a factor of two when compared with orderings derived from space filling curves.

1 Introduction

We present a system that provides efficient disk access when querying spatial structures across multiple, heterogeneous data sets defined over a non-convex spatial domain. Non-convex means that the line segment connecting two points in the domain is not guaranteed to be contained within the domain's interior. We built the system to support data discovery and analysis of environmental data sets from the Chesapeake Bay, which exhibits non-convexity: estuaries are long and skinny with a large number of winding, tendril-like tributaries. Non-convexity arises in many other systems, such as medical applications (the circulatory system), manufactured systems (road networks and indoor air systems), and other natural systems (turbulent structure and flow in porous media).

Querying spatial data has emerged as an important topic in database management systems, for both scientific databases [1, 2] and geo-spatial services, such as geotagging, urban computing [3], and collaborative sensing [4]. The Chesapeake is the most intensively studied estuary in the United States. Our databases

include more than 15 observational data sets from buoys, shallow-water sensors, cruises, aircraft observations, etc. alongside high-resolution water-quality and hydrodynamic models. Providing scientists with the ability to compare, correlate, and join data among these data sets has transformed our understanding of critical Bay processes [5], such as hypoxia (oxygen depletion) and the variable influence of flow and nitrogen load on water quality and the benthic community. For example, using high-resolution model output to inform the interpolation of water-quality measurements from 40 measurement sites has produced more accurate maps of salinity and dissolved oxygen over 10 years of data than were previously possible [6].

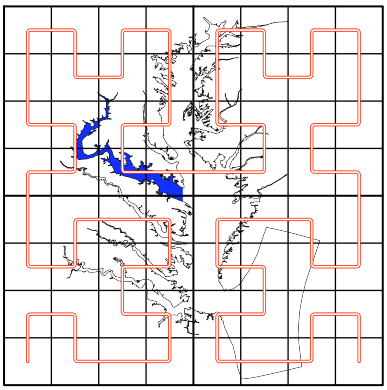

Fig. 1. A Hilbert ordering applied to the Chesapeake Bay. The Potomac River is highlighted.

Traditional approaches for organizing spatial data on disk do not meet the requirements of Chesapeake Bay data sets specifically and data drawn from non-convex spatial domains in general. Two aspects make the problem challenging: (1) the non-convex boundaries define a complex notion of spatial proximity among data and (2) the support of queries across multiple, heterogeneous data sets requires a regular decomposition of space.

Data organizations that do not respect the boundaries of non-convex spatial domain lead to poor I/O performance when querying spatial structures. Most decompositions of space, e.g. tessellations, space-filling curves, region trees, and packed R-trees [7], group data by Euclidean distance. However, data points near each other in Euclidean space may have a weak spatial relationship. For example, Figure 1 shows how a Hilbert curve fails to cluster the region of space associated with the Potomac River (shaded). If one uses the linear ordering following the curve to define the address space, the Hilbert curve will break up data from the river into many disjoint regions.

A regular decomposition of the spatial domain creates a global addressing scheme that makes comparing and joining data from different data sets possible [8]. The index value (lookup key) of a data point derives from its location in space alone. For this reason, regular decompositions are called *data independent*. Owing to data independence, data will have an index value that only depends on its spatial position, making it possible to compute joins across multiple data sets. Similarly, spatial properties of the index are preserved and we can compute nearest neighbors and clusters across multiple data sets.

We provide a data organization and indexing system based on a regular-decomposition that captures the complex spatial relationships of non-convex domains. We call this RINGS for regular indexing of non-convex geometric spaces. RINGS employs a computational geometry construct, the medial-axis transform

(MAT), to automatically characterize the domain. Based on the MAT, we partition space and create an index on the partitions. RINGS defines a primary index for our databases in that we lay the data out on disk in index order by using RINGS indices as the primary keys in a sorted relation.

RINGS derives its performance benefits from encoding the geometry of the domain in the index and data layout. It represents the medial-axis as a tree and enumerates its branches to generate the indices. In doing so, all data down any branch of the MAT, regardless of the geometry of the sub-region, are contiguous in index space and on disk. For Chesapeake Bay data, this means that any substructure—an estuary, river, or bay—consists of a contiguous index range and can be read sequentially from disk. RINGS preserves index contiguity at multiple scales in support of self-similar structures, e.g. Rock Creek occupies a contiguous sub-region of the Potomac River's contiguous index range.

We evaluate the performance of RINGS in comparison to indexes derived from space filling curves. On workloads derived from queries submitted to the Chesapeake Bay Environmental Observatory, our implementation improves query processing times by a factor of two for queries of contiguous substructure. An evaluation against randomly generated shapes shows further performance improvements as non-convexity increases.

2 Related Work

The Medial Axis Transform: Introduced by Blum more than four decades ago, the Medial Axis Transform [9] is a computational geometric structure characterizing the "central skeleton" of a shape. In 2-d, it comprises a network of curves, whereas in higher dimensions it can be made up of surface patches, volume elements, and higher degree manifolds up to co-dimension one. The MAT characterizes the general structure of the shape and has found applications in varied disciplines, such as path planning [10] and image segmentation in medicine [11], studying drainage patterns in watersheds [12], surface reconstruction in computer graphics [13], and routing in sensor networks [14].

The computation of the MAT has proved difficult for all but the simplest of polyhedral models, and numerous approximating methods have been proposed. These have included using morphological thinning [15, 16], adapting quadtrees to define the MAT [17], approximating the MAT by the vertices of a Voronoi diagram defined by a dense sampling of the boundary [18], modifying the underlying spatial metric [19], pruning the set of interior edges in a Delaunay triangulation computed from the union of boundary and Voronoi vertices [20], and using the center points of a constrained Delaunay triangulation to define the skeleton [21].

Regular Decompositions: The simplest regular decomposition divides space into a grid of square (2-d) or cubic (3-d) cells. To construct an index from a grid, one defines a linear-ordering over the cells. For simplicity, this can be done in a row-major or column-major order. Many applications choose to use space-filling curves, such as the Z-order or Peano-Hilbert curve, which cluster data

spatially [8]. Space-filling curves reduce the number of accesses to indexes for query regions that are square, circular, or simple polygons [22].

Region quadtrees in 2-d [23] and region octrees in 3-d [24] recursively partition space into successively smaller squares or cubes. By themselves, the trees are just spatial search structures and do not define a linear ordering. *Locational codes* derive labels for each leaf based on a Z-ordering of the grid implied by the smallest leaf nodes [8].

The Astrophysics community has adopted the Hierarchical Triangular Mesh (HTM) [25] regular decomposition for indexing and data organization. The HTM defines a space-filling curve for spherical coordinates using a quad tree: it approximates a sphere by starting with an octahedron and recursively refining each triangle into 4 sub-triangles.

Space filling curves and locational codes cannot respect non-convex boundaries. Even if the underlying grid identifies the boundary, e.g. a region tree that has small cells along the boundary and bigger cells elsewhere, the linearization remains insensitive to the structure of the domain.

Other Spatial Indexing Techniques: R-trees [26] index objects in space by providing navigation paths to minimum-bounding rectangles. While the structure is data defined, one could produce a regular decomposition from R-trees by triangulating the spatial domain and building an R-tree over the triangles. However, when indexing triangles, R-trees have many overlapping minimum bounding rectangles [27], which requires the navigation of multiple tree paths. R+-Trees [28] mitigate this by dividing objects into disjoint bounding boxes. As with other spatial search data structures, R-trees do not order data on disk. The preferred technique for placing R-tree data onto disk, called Hilbert-packing [7], uses space-filling curves to define a linear order on the R-tree's bounding rectangles.

Papadomanolakis et al. describe a system for indexing unstructured tetrahedral meshes [27]. When searching for a point (or nearest neighbors), they use a Hilbert-curve to find a mesh cell near the point and then conduct a local search based on mesh connectivity. We adopt similar search techniques, using a pixelation of all space to identify a nearby point on the MAT and then search the local structure of the MAT to identify a point, neighbor, or range. They do not address data placement on disk.

The MAT in Geospatial Applications: The medial-axis transformation has been used for cartographic and hydrological applications, because it preserves topological structure. McAllister and Snoeyink [29] use the medial-axis to characterize rivers, e.g. match shores, calculate width, and estimate volume. Gold et al. [30] use the medial-axis transform to generalize objects while maintaining the spatial relationships between them. Neither previous work applies the MAT to data organization or system design.

3 Motivating Applications

Estuarine studies are increasingly data-intensive, pulling in a heterogeneous mix of observational and model-derived data to gain further understanding of hydrological and biochemical processes. The Chesapeake Bay Environmental Observatory (CBEO) provides the infrastructure for developing new methods for interacting with data [5]. Our collaborators have been using the CBEO to query Bay structures for several years [31]. Our goal in this work is to improve the I/O performance of their workloads.

One project studies how changes in the level of dissolved oxygen effect animal species in benthic (bottom) regions. One of the better datasets for this study comes from the tidal region near Calvert Cliffs Nuclear Power Plant, where detailed monitoring has taken place since 1968. This data is combined with other data, both modeled and measured, to gain a better understanding of how the benthic community composition changes with respect to dissolved oxygen.

Individual estuaries are often studied as their own microcosm. The Patuxent River in Maryland is one such tributary. Testa et al. [32] have studied the long-term changes in the water quality with both measured data and a box model that estimates the estuary's time-dependent water circulation. Using these methods, the authors were able to link anthropogenic and environmental events, e.g. identifying how new sewage treatment techniques resulted in nutrient changes.

Spatial interpolation techniques are widely used in estuarine science to estimate the distribution of variables over all space from a discrete number of measurements. Kriging is one such method for generating spatial predictions by using a model of the spatial correlation of the measurements to predict unknown values as a function of location. Murphy et al. [6] have extended traditional kriging techniques to use the output from a water-quality model as a covariate to improve the interpolation of data collected from fixed locations. Ongoing work includes the integration of non-Euclidian cost functions, such as water-distance and hydrodynamic travel time, with these techniques.

The workloads presented by these applications have several common features that inform the design of Rings. They all perform data fusion, bringing data together from heterogeneous sources to facilitate new scientific discovery. Rings supports this efficiently through a regular decomposition of space which provides for consistent indexing across multiple data sources. Also, many queries focus on regions of interest that correspond to physical structures. This arises naturally, because the different structures express different properties: hydrodynamics, geochemistry, land-use, etc. Rings captures structure through the use of the medial-axis transform, which automatically identifies spatial features and clusters data by structure.

4 Methodology

In this section, we describe how we generate an index from a shape file. We also outline how to process point, range, region, and water-distance queries using this

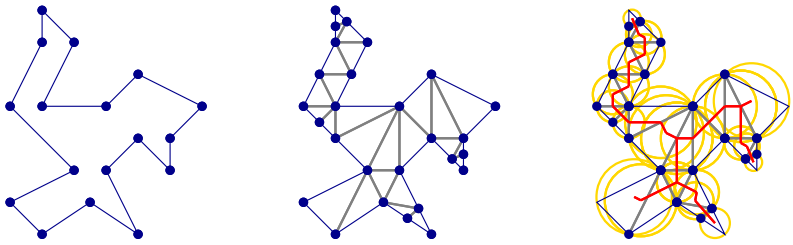

Fig. 2. Finding the MAT: We take simple polygon (left), subdivide edges to create a Gabriel-conforming graph and construct a Delaunay Triangulation (center), and connect the circumcenters of adjacent triangles to define the approximate MAT (right).

index. The input to this process is a polygon that describes the spatial domain. We will output a triangulation of the space, a linear ordering of the triangles, and auxiliary data structures to be used for query processing.

4.1 Approximate Medial-Axis

The fundamental principle underlying our approach is the use of the medial-axis transform (MAT) to characterize the spatial domain. The MAT is commonly referred to as the *skeleton*. For 2-d domains, it is a series of curves that represent the "centerline" of the shape. In Rings, we generate a piece-wise linear approximation of the MAT.

Our approach is to construct a Delaunay triangulation of the vertices and use the edges of the dual Voronoi diagram to define the MAT. However, the Delaunay triangulation does not necessarily preserve the edges of the original polygon. To address this, we subdivide the edges of the polygon to guarantee that each sub-edge has the property of being a *Gabriel edge* [33]. Formally, this means the circle that has a Gabriel edge as its diameter contains no other vertex in the polygon. Using Gabriel edges has several advantages:

1. The edges of the original polygon are a subset of the Delaunay triangulation as the end-points of each edge satisfy the *empty circle property* [34].
2. Each Delaunay triangle is entirely interior or entirely exterior to the bounding polygon.
3. The circumcenters of interior Delaunay triangles and the Voronoi edges connecting two such triangles are interior to the polygon.

As a result, the Delaunay triangulation of the vertices provides a well-defined partition of the spatial domain, and our derived approximate MAT, generated by connecting the circumcenters of adjacent triangles, is guaranteed to be contained within the domain.

Figure 2 illustrates our MAT construction. Starting with an initial polygon, we subdivide the edges to ensure that the sub-edges are Gabriel-conforming and compute the Delaunay triangulation. Then, we obtain the approximate medial axis by connecting the circumcenters of adjacent interior Delaunay triangles.

4.2 Linearizing the MAT

Subsequent steps will require the MAT to take the form of a tree with no cycles. This is the case for genus-0 boundaries, i.e. simple, connected, closed polygons. For non-genus-0 boundaries, we fill in all holes interior to the domain, creating a genus-zero boundary. This approach is simple and appropriate when the interior structures are small. For the Chesapeake Bay, the holes are a series of small islands. Leaving the islands in would produce a MAT that encircles every island and may not represent the outline of the shape well. The issue of holes can also be addressed more generally by building a MAT and breaking the cycles to construct a tree (e.g., computing the minimum spanning tree). Many other tendril-like structures, however, are not easily reducible to trees; for instance, road networks are highly interconnected. Nonetheless, it is always possible to find a spanning tree that covers the entire network, the selection of which may be tuned for a specific application.

4.3 Index Generation

Using the MAT to guide the process, we assign a unique index to each triangle. The index serves as a spatial identifier, with data points inside and objects intersecting the triangle labeled by the triangle's index. Using the fact that adjacent vertices in the MAT correspond to adjacent Delaunay triangles in the triangulation of the polygon, we index the triangles using a simple tree traversal. Starting at a leaf of the tree, we traverse the tree in depth-first order and label each triangle we encounter with a unique, consecutive identifier. At branch points, we index all of the vertices on one branch before proceeding to the indexing of triangles on the second branch. (Figure 3 shows an example where the indexing was started at the top-left triangle.)

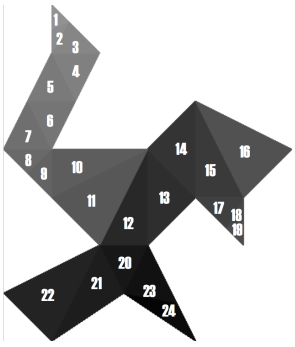

Fig. 3. Spatial index generated from the polygon in Figure 2.

This generates an indexing in which the triangles within a branch of the MAT are assigned consecutive addresses. Since the branches correspond to logically distinct regions (estuaries, rivers, bays, etc.) we obtain an indexing that characterizes the spatial domain.

Order of Medial Axis Traversal:

Once a starting node is selected, the indexing of triangles remains deterministic until we reach a branch point. The choice of which branch to traverse first can give rise to markedly different indexing of the domain. We consider two different approaches.

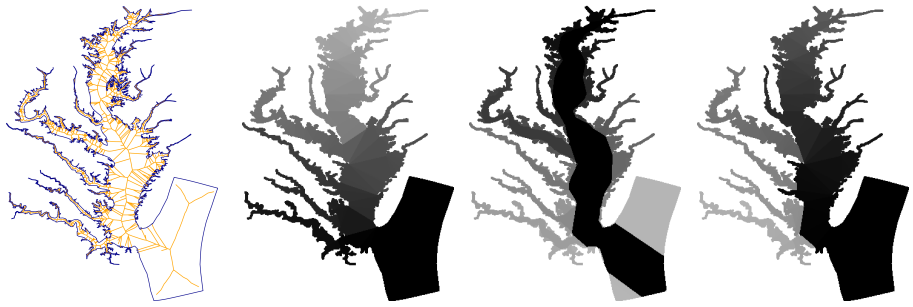

Fig. 4. Our primary shape, a union of the Chesapeake Bay outline combined with the outline of the water-quality models with an extra 1 km buffer. The approximate medial-axis is shown in (a). We also show several gray-scale visualizations of the indexes generated by traversing the MAT, including the results of using a least-weight-first ordering for branch traversal (b), using heaviest-weight-first ordering (c), and using heaviest-weight-first with a threshold for minimal area branches (d).

When encountering a branch in the MAT, we select which branch to follow based on the weight of the subtree down that branch, i.e. the sum of the area of the descendant triangles. In the first technique (Figure 4(b)), we traverse the least-weighted subtree first. In the second (Figure 4(c)), we traverse the heaviest subtree first. Traversing the least-weighted subtrees first gives a smoother indexing of the shape, including all of the micro-structure on the sides of each tributary. Smoothness would seem to be best, by our intuition, but does not cluster large structure. Traversing the heaviest subtrees first, yields a consecutive "main stem" of the bay and each of the tributaries. However, it introduces discontinuities in the index where small structures break off from larger ones.

We strike a compromise by including small, border areas into the heaviest first policy (Figure 4(d)). Specifically, we choose a threshold weight (area) and use heaviest if both branches are above the threshold and least-weighted otherwise. We choose this approach for the Chesapeake Bay with a threshold of 2.5% of the total weight of the shape, which has the effect of combining the shallows on the boundary of the main stem with the main stem itself.

Auxiliary Data Structures:

The index encodes spatial locality in a linear ordering of the Delaunay triangulation and defines an organization on disk based on that ordering. However, it does not naturally associate points in space with that index.

We use an additional lookup table to determine the triangle(s) associated with a point or region of space. We generate the lookup table by rasterizing the domain onto a low-resolution pixel grid. For each pixel, we store a list of the triangles that intersect the pixel. The table is generated once for each shape. The resolution process for a point within this pixel traverses the list until it finds the triangle that contains the point.

The size of the lookup table needs to be chosen with consideration. Too fine a pixelation produces an overly large lookup table, which consumes cache space. Too coarse a pixelation would result in too many triangles intersecting each pixel, which could slows query evaluation. For the Chesapeake Bay, we chose a grid of 150,000 (350x500) pixels to render the 22,729 indexed regions (triangles). The data structure consumes only 1.17MB of memory. The domain covers 29.6% of the pixelated area with an average list length of 2.54 triangles per covered pixel. The maximum length of the list is 54 triangles and only 53 pixels have 25 or more candidates. In practice, these long lists are not a concern as they occur in narrow regions with many small triangles which rarely contain data.

We also maintain an adjacency map that lists the adjacent triangles for each triangle. We use the adjacency map to quickly traverse the local structure of the MAT and the triangulation.

4.4 Query Processing

We use the MAT index, lookup table, and adjacency map to implement spatial selection queries.

Index Lookup/Point Query: As described in the previous section, this basic operation finds the index value for a coordinate location, and is used most frequently with data ingest.

Range and Polygon Queries: Queries in which the user specifies either a Euclidean distance from a point or a polygon of interest are processed by identifying candidate triangles that intersect the query. We consult the lookup table to find pixels that overlap the query. We include all the data from triangles that are entirely inside the query. Additionally, the data from triangles that intersect the query boundary are evaluated to determine if they are inside the range.

Approximate Water-Distance Range Query: Users may specify a range query in terms of water distance, i.e. the length of the shortest path between two points that lies entirely inside the shape (domain). This query has real-world implications, e.g. travel time between two points. It also could be a more useful distance metric for interpolation than Euclidean distance [35]. For simplicity, we implement water distance approximately. We sum the length of the medial-axis in the traversal between two points. Having selected the portion of the medial-axis, we process the query as a range query.

The accuracy of approximate water distance depends upon the aspect ratio and size of the triangles. Our value will never be *less* than the actual water distance. In the future, we will consider supporting shortest path queries [36] for obtaining the exact distance.

5 Implementation

Our software consists of a utility that generates the spatial index from a shape file and database routines that use the precomputed index to process queries.

```
SELECT IndexFromLatLong(@map, 39.46678, -75.87466)
```

(a) Finding the index value for a specific location

```
SELECT data.* FROM MedialRangeWalk(@map, 39.0257, -76.2017, @range)
AS m JOIN wqm57k.dbo.WQM_data AS data ON m.IndexID = data.IndexID
ORDER BY range
```

(b) Table-valued function that performs a water-distance range query.

```
SELECT * FROM DirectionalMedialWalk (@map, 37.97155, -76.330637, +1)
AS m JOIN eotb.EOTB_data WHERE m.IndexID = eotb.IndexID
```

(c) Retrieve all EOTB data upstream from a specified point.

```
SELECT cims.*,MedialDistanceBetweenPoints
(@map, 39.2833, -76.6097, cims.Latitude, cims.Longitude)
AS Distance FROM cims.Station_info AS cims ORDER BY Distance
```

(d) Get the approximate water-distance between Baltimore and each station.

Fig. 5. Sample SQL Queries

The preprocessor was written in C++ and uses triangulation routines from CGAL [37]. Our input files are typically ArcGIS shape files containing a polygon with the region of interest. The index generation process on our Chesapeake Bay outline of 80,000 points takes several seconds to run on a standard PC.

Compiling map files: We "compile" all of the generated data into C# class files. This includes all metadata, the pixelated look-up table, triangle vertices, and adjacency map. We chose these data structures for ease of implementation, requiring only basic array indexing to retrieve the vertices for a triangle. We will explore a winged-edge [38] structure in the future for a more compact storage format.

Database Implementation: We implement query processing routines that use these data structures in Microsoft's SQL Server. We use a combination of user-defined scalar and table-value functions (TVFs). All functions are written in C# using SQL's common language runtime interface [39].

We store data in sorted database tables, using the MAT-index value as the primary key for a clustered index. Our largest datasets have a large volume of data output for all data points at regular time intervals. We cluster these datasets first by time and then by index. Queries to these databases tend to select distinct regions of data across specific time-ranges for comparison.

Range Queries: Range queries, both regular and water-distance, are implemented as table-valued functions. A list of all candidate regions which overlap the range query are returned. We obtain the requested data by joining the candidate indexes with the data tables. Finally, we apply a filter to the result to eliminate data located within the border index regions yet outside the query range. Water-distance queries are done through a similar interface (Figure 5(b)). Another variation of the range query allows for a complete traversal from a point on the medial-axis. For example, a simple query can be written to select all data contained within the Potomac river (Figure 5(c)) by traversing upstream from the mouth.

Calculated Columns: We also implement the water-distance query in a user-defined function. This allows it to be called as a subroutine to a larger query that performs further data analysis. Figure 5(d) shows an SQL query that retrieves a list of CIMS monitoring stations sorted by their approximate water-distance from the Baltimore Inner Harbor.

6 Evaluation on CBEO Data

We have implemented Rings on the Chesapeake Bay Environmental Observatory (CBEO) and evaluate its performance using workloads derived from the queries submitted to the CBEO. We characterize the CBEO's workload by query type and built workloads that cover the entire spatial domain for each query type. The CBEO's users tend to study a specific region. For example, one scientist queries the Patuxent River, an entire estuary. For this, we have generated workloads that randomly select space-constrained water-distance ranges. We also examine the performance of circular range queries that are unconstrained by boundaries, because they are the most fundamental geospatial range query.

We start by looking at the indexes generated on our primary dataset, the Chesapeake Bay. Figure 4 shows the medial-axis and visualizations of the indexes generated on the spatial domain. The shape comprises the union of the outline of the Chesapeake Bay with the cells of the water quality models. We then expand the boundary by 1 km in order to include the stations that take tidal samples.

Our experiments use the CBEO's most popular data set: the 56,920-cell water quality model. The water-quality model has cells of roughly equal surface area that cover the entire bay and major estuaries. We use a 1 month subset of the output totaling 1.7 million rows of 296 bytes each for a total data size of 522MB.

We compare Rings against other spatial organization techniques. We implement two space filling curves: our implementation of a Z-order curve and the hierarchical triangular mesh (HTM) index [1, 25] used widely by the Astronomy community. HTM supports geospatial as well as celestial coordinates. We also provide latitude-longitude addressing (approximately row major ordering).

For each ordering, we create a database table that contain the same data. Tables are sorted by $(time, index)$. For Rings and HTM, the index is derived from the index address of the triangle containing the data point. For Z-order, it is the locational code of the square containing the data point.

For all results, we measure the time spent reading the data table. For each query, we consult the index in memory and retrieve a list of the index elements that could satisfy the query; triangles for Rings and HTM, and squares for Z-order that intersect the query region. Our timed operation is the `JOIN` between the list of specific index values and the data tables sorted by spatial index. SQL optimizes this as a `HASH JOIN` between this list and a non-clustered index created on each data table for all $(time, index)$ pairs.

Range Queries: This experiment compares the performance of all data organization schemes on circular range queries. The circular range ignores boundaries and returns data points from multiple non-connected water regions. Thus, we

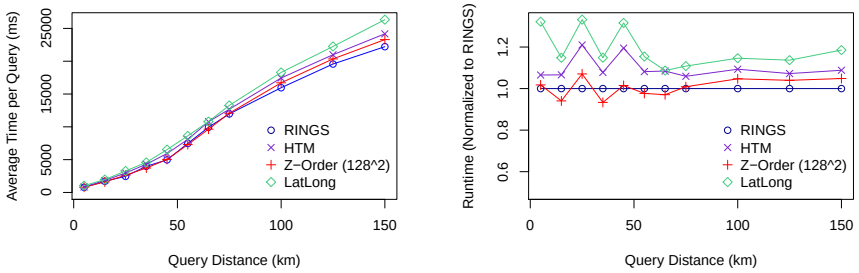

Fig. 6. Index performance for range queries on the water-quality model. RINGS' performance is comparable to the other spatial organization techniques.

expect RINGS to realize no advantage when compared with HTM or space-filling curves. The query geometry more closely matches the shape of the recursive triangle or rectangular decomposition of space used by HTM and space-filling curves respectively than it does the MAT used by RINGS.

The experiment selects 200 points uniformly at random in the interior of the Chesapeake Bay and its tributaries. The same points and ranges are used for all indexes. We do this for ranges from 5 km to 250 km in 10-50 km increments. The Chesapeake Bay outline extends roughly 350 km from north to south and 230 km from east to west.

We show both the average runtime per query and the runtime normalized to that of RINGS in Figure 6. Even though these range queries which do not employ the structure of the Bay, RINGS' performance matches the performance of Z-order and exceeds the performance of HTM.

For some queries, RINGS performs more work than the Z-order curve because it has to read more data. It generates a list of candidate triangles, reads all data within those candidate triangles, and filters out data not within the range. This can be problematic when the triangles are as large or larger than the range of the query. This occurs with smaller ranges in the main stem.

In contrast, RINGS benefits when the query region is long and skinny and includes space exterior to the boundary. HTM and Z-order curve grow their index recursively in triangles and squares respectively, which prevents them from indexing long skinny structures contiguously. In contrast, RINGS adapts to the local geometry.

Water-Distance Range Queries: When constraining the range by the geometry of the domain, RINGS improves performance. As with the previous experiment, we select 200 points uniformly at random, however this time we use the water-distance instead of the Euclidean distance. That is, the query selects all data for which the shortest path to the query point *interior* to the Chesapeake Bay is less than the specified range. We do this for ranges from 5 to 500km in

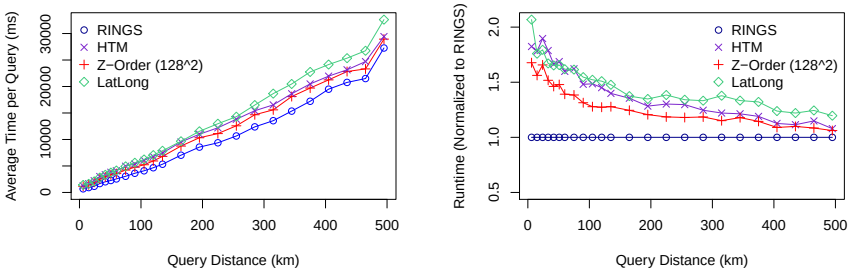

Fig. 7. Index performance for water-distance range queries on the water-quality model.

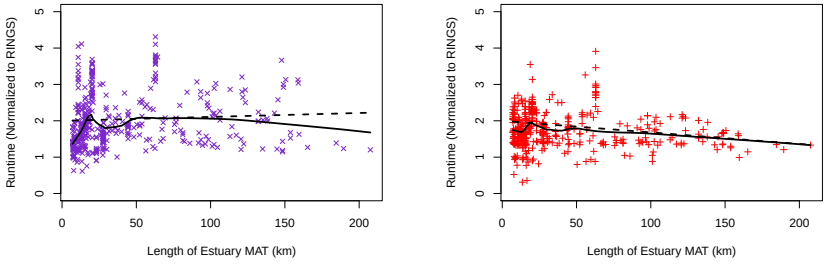

Fig. 8. Scatter plots comparing index performance for estuary queries between Rings and both HTM (left) and Z-Order (right), each with linear regression and locally weighted smooth (lowess) curve. Both plots are normalized to Rings.

increments of 10-30km. As Figure 7 shows, Rings improves performance in all cases. The improvement is most dramatic for smaller ranges.

Rings cuts the query time in half for for ranges up to 50 km. The improvements are most dramatic for queries that select a substantial amount of data from tributaries. It is not uncommon to see an individual query speedup a factor of six when comparing Rings with the Z-order. For queries in the main stem, the structure of the MAT is less helpful. Similarly, larger ranges reduce the benefit, because the queries are less selective. For a range that covers the entire Bay, the query will result in a table scan. Our results reflect this as the performance of all indexes begin to merge at large ranges.

Estuary Queries: Scientists frequently use the CBEO database to query contiguous substructures, e.g. the Potomac or the Patuxtent River (see Section 3). This usage pattern was the original motivation for the design of Rings.

To evaluate the performance of Rings when querying such substructures, we create a random workload based on the `DirectionalMedialWalk` query (see Figure 5(c)). In this experiment, we select points from the interior of the domain

uniformly at random. If the point is on the main stem or in the ocean region, we discard it. Otherwise, we query all data upstream from that point. Our scientists use this query to select an entire tributary by picking a point on the MAT at the mouth of that tributary. The query is particularly convenient, because the scientist does not need to specify geographic boundaries.

For estuary queries, RINGS reduces the query time in half, on average, when compared with the Z-order or HTM. Figure 8 shows these results as a function of the length of the estuary from the randomly selected point to the farthest point upstream. The length is, in some sense, the query diameter of a water-distance query. The results show a tremendous amount of variance, because performance depends upon the geometry of the query region. The few queries in which HTM and Z-order outperformed RINGS are short queries, most of which were centered near the source of the rivers.

7 Evaluation on Random Shapes

In order to demonstrate the applicability of RINGS beyond our prototypical application, we evaluate its performance on randomly generated shapes of varying complexity. We use a 64^2 (4096-cell) Z-order curve as a basis for comparison. We execute two queries on the random data: (1) a rectangular range query shows that RINGS preserves performance even when the query space matches the index space and (2) a water-distance distance query indicates the performance benefits when the query conforms to the spatial structure.

Shape Generation: We generate shapes by splitting boundary segments on a polygon by repeatedly moving their midpoints. We begin with a boundary defined by four points arranged as a square. We create each new edge by selecting a random segment to divide with a new point. We then move that point a random direction and distance (bounded by a defined amplitude). We verify that the new segments do not intersect or coincide with any existing segment, because both would lead to an invalid polygon. If the move fails this test, we try again. This process is repeated until all of the desired points have been inserted. Four sample shapes are shown in Figure 9. We chose to generate square shapes so that they align well the Z-order curve used in comparison.

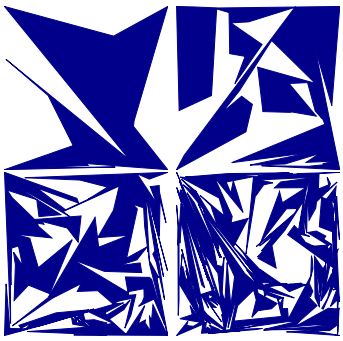

Fig. 9. Shapes with 15, 50, 200 and 500 edges.

A metric based on the distance between points in the interior of the shape characterizes the "complexity" of the generated shapes. This metric is defined as the ratio of the Euclidean distance to shortest interior path between the points. The metric captures the distinction between Euclidean distance and water-distance, a distinction that motivated the design of RINGS. We compute

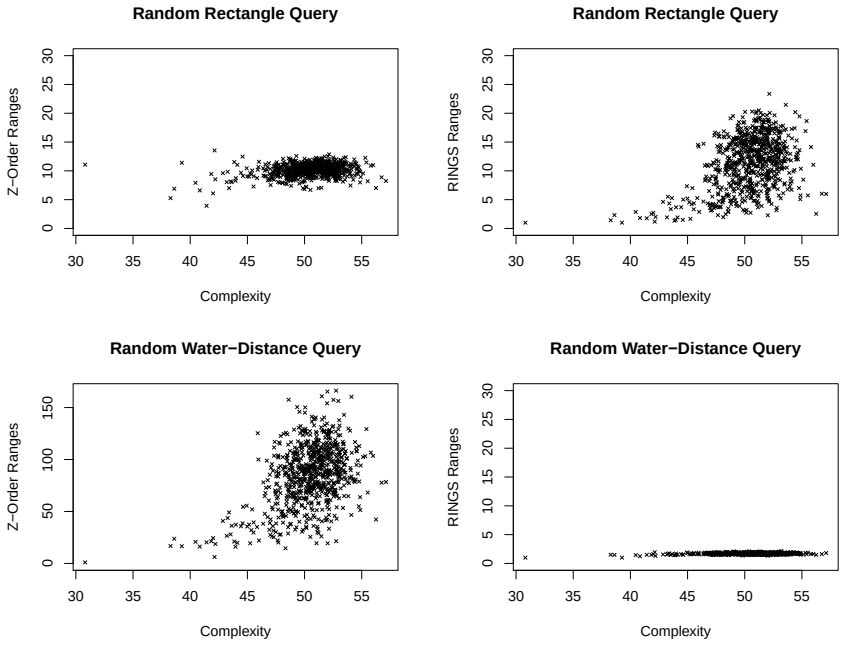

Fig. 10. The number of index regions necessary to resolve a query as a function of shape complexity.

this as an average over 2000 points randomly selected from within the interior of the shape. The RINGS medial axis gives an upper bound on point-to-point interior distance, which we use in the calculation.

Experimental Results: For both distance queries, we examine the number of non-contiguous ranges of data that the queries access. The number of ranges roughly corresponds with the number of disk seeks needed to read the data. The data set consists of 10,000 points randomly placed within the interior of the shape. For each experiment, we perform 5,000 randomly generated queries. Each range query is centered at a randomly chosen point interior to the shape. The rectangular range queries were restricted to be between 10% and 80% of the shape's bounding box. The range used for water-distance queries was a random value between 5% and 50% of the shape's perimeter.

The rectangular query results show that RINGS provides roughly equivalent performance to the Z-order curve with higher variance (Figure 10). The number of disjoint data regions in the Z-order curve index vary relatively little because the Z-order curve indexes the entire space not the domain. RINGS exhibits similar performance with some increase in the disjoint data regions as the shape becomes more complex. As the shape increases in complexity, each rectangular query region intersects the boundary of the shape more times. But, even for shapes much more complex than the Chesapeake Bay, RINGS compares favorably.

For the water-distances queries for which it was designed, RINGS accesses very few disjoint regions of data. In contrast, the Z-order curve has to access many disjoint regions of data and performance worsens as shape complexity increases. Because the domain defines the query region, the query region has a complex boundary that intersects the Z-order curve many times. For RINGS, the number of disjoint regions increases very little as the shape grows in complexity. Disjoint regions arise only when the query region spans multiple internal structures, i.e. when the tree defined by the MAT forks and the query does not cover both branches in their entirety. Comparing these results with the complexity of the Chesapeake Bay reveals that the Chesapeake lies at the low end of the range for which the non-convex complexity results in marked performance improvements and more complex systems would realize even bigger benefits.

8 Conclusions

We have presented RINGS, an organization system for spatial databases based on the automatic characterization of non-convex domains. The key behind our approach is the use of the medial-axis transform for indexing the data and supporting efficient traversal of spatial structures. Our system was designed for environmental data sets and, more specifically, for the Chesapeake Bay Environmental Observatory's (CBEO) heterogeneous collection of observations and models. In empirical evaluation, we have found that RINGS is competitive with traditional indexing methods for general range queries and improves performance substantially on queries that take into account the geometry of the domain. These queries arise naturally in the CBEO, because the users study structures, such as rivers and estuaries, that are contiguous within the interior of the Bay.

Acknowledgments

We thank Jennifer Bosch, Damian Brady, Jeremy Testa and the other members of CBEO team for their expertise on estuarine science and Chesapeake Bay data. We also thank Damian Coventry for his pseudo-random polygon code. This work was supported by the National Science Foundation award ATM-0618986.

References

1. Szalay, A., Gray, J., Fekete, G., Kunszt, P., Kukol, P., Thakar, A.: Indexing the sphere with the hierarchical triangular mesh. Technical Report MSR-TR-2005-123, Micrsoft Research (2005)
2. Perlman, E., Burns, R., Li, Y., Meneveau, C.: Data exploration of turbulence simulations using a database cluster. In: SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, ACM (2007)
3. Joseph, A. D. (editor): Urban computing and mobile devices. IEEE Pervasive Computing **6**(3) (2007) 52–57
4. Reddy, S., Burke, J., Estrin, D., Hansen, M., Srivastava, M.B.: A framework for data quality and feedback in participatory sensing. In: SenSys. (2007)

5. Ball, W.P., et al.: A prototype system for multi-disciplinary shared cyberinfrastructure—Chesapeake Bay Environmental Observatory (CBEO). Journal of Hydrological Engineering **13**(10) (October 2008) 960–970
6. Murphy, R.R., Curriero, F.C., Ball, W.P.: Comparison of spatial interpolation methods for water quality evaluation in the Chesapeake Bay. Journal of Environmental Engineering **136**(2) (2010) 160–171
7. Kamel, I., Faloutsos, C.: On packing r-trees. In: CIKM '93: Proceedings of the second international conference on Information and knowledge management, ACM (1993) 490–499
8. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc., San Francisco (2006)
9. Blum, H.: A transformation for extracting new descriptors of shape. In: Models for the Perception of Speech and Visual Form. MIT Press, Cambridge, MA (1967) 362–380
10. Ge, Y., Stels, D., Wang, J., Vining, D.: Computing centerline of a colon: A robust and efficient method based on 3d skeletons. Journal of Computer Assisted Tomography **23(5)** (1999) 786–794
11. Joshi, S., Pizer, S., Fletcher, P.T., Yushkevich, P., Thall, A., Marron, J.: Multiscale deformable model segmentation and statistical shape analysis using medial descriptions. IEEE Transactions on Medical Imaging **21(5)** (2002) 538–550
12. TRIM Watershed Atlas. `http://www.barrodale.com/watershed/twapage.htm`
13. Amenta, N., Choi, S., Kolluri, R.: The power crust. In: Sixth ACM Symposium on Solid Modeling and Applications. (2001) 249–260
14. Bruck, J., Gao, J., Jiang, A.: MAP: medial axis based geometric routing in sensor networks. In: MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking, New York, ACM 88–102
15. Maragos, P., Schafer, R.: Morphological skeleton representation and coding of binary images. IEEE Transactions on Acoustics, Speech and Signal Processing **34** (October 1986) 1228–1244
16. Lam, L., Lee, S., Suen, C.: Thinning methodologies: A comprehensive survey. Transactions on Pattern Analysis and Machine Intelligence **14**(9) (September 1992) 869–885
17. Joan-Arinyo, R., Pérez-Vidat, L., Gargallo-Monllau, E.: An adaptive algorithm to compute the medial axis transform of 2-d polygonal domains. In: CAD Systems Development: Tools and Methods, London, UK, Springer-Verlag (1997) 283–298
18. Brandt, J.W.: Convergence and continuity criteria for discrete approximation of the continuous planar skeletons. Image Understanding **59** (1994) 116–124
19. Aichholzer, O., Aurenhammer, F., Alberts, D., Gärtner, B.: A novel type of skeleton for polygons. Journal of Universal Computer Science **1**(12) (1995) 752–761
20. Gold, C.: Crust and anti-crust: A one-step boundary and skeleton extraction algorithm. In: Annual Symposium on Computational Geometry. (1999) 189–196
21. Zou, J.J.: A fast skeletonization method. In: DICTA. (2003) 283–288
22. Moon, B., Jagadish, H.V., Faloutsos, C., Saltz, J.H.: Analysis of the clustering properties of the Hilbert space-filling curve. IEEE Transactions on Knowledge and Data Engineering **13**(1) (2001)
23. Klinger, A. In: Patterns and Search Statistics. Academic Press (1971) 423
24. Hunter, G.M.: Efficient computation and data structures for graphics. PhD thesis, Dept. of Electrical Engineering and Computer Science, Princeton University (1981)
25. Kunszt, P.Z., Szalay, A.S., Thakar, A.R.: The hierarchical triangular mesh. In: ESO Astrophysics Symposia: Mining the Sky. (2001) 631–637

26. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data. (1984) 47–57
27. Papadomanolakis, S., Ailamaki, A., Lopez, J.C., Tu, T., O'Hallaron, D.R., Heber, G.: Efficient query processing on unstructured tetrahedral meshes. In: SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. (2006) 551–562
28. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R+-tree: A dynamic index for multi-dimensional objects. In: VLDB. (1987)
29. McAllister, M., Snoeyink, J.: Medial axis generalization of river networks. CaGIS **27**(2) (2000) 129–138
30. Gold, C., Thibault, D., Liu, Z.: Map generalization by skeleton retraction. In: ICA Workshop on Map Generalization. (1999)
31. Chesapeake Bay Environmental Observatory (CBEO), `http://cbeo.communitymodeling.org/`.
32. Testa, J.M., Kemp, W.M., Boynton, W.R., Hagy III, J.D.: Long-term changes in water quality and productivity in the Patuxent River estuary: 1985 to 2003. Estuaries and Coasts **31**(6) (December 2008) 1021–1037
33. Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. Systematic Zoology **18**(3) (1969) 259–278
34. Amenta, N., Bern, M., Eppstein, D.: The crust and the β-skeleton: combinatorial curve reconstruction. Graphical Models and Image Processing **60** (1998) 125–135
35. Rathbun, S.L.: Spatial modelling in irregularly shaped regions: kriging estuaries. Environmetrics **9**(2) (1998) 109–129
36. Guibas, L.J., Hershberger, J.: Optimal shortest path queries in a simple polygon. In: SCG '87: Proceedings of the third annual symposium on Computational geometry, New York, ACM (1987) 50–63
37. CGAL, Computational Geometry Algorithms Library, `http://www.cgal.org/`.
38. Baumgart, B.G.: Winged edge polyhedron representation. Technical Report CS-TR-72-320, Stanford University (1972)
39. Rathakrishnan, B., Kleinerman, C., Richards, B., Venkatesh, R., Rao, V., Kunen, I.: Using CLR integration in SQL Server 2005. Technical report, Microsoft (2005)