In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 409-412, Dallas, TX, November 2014

Viewing Streaming Spatially-Referenced Data at Interactive Rates [*]

Shangfu Peng Hanan Samet Marco D. Adelfio

Center for Automation Research, Institute for Advanced Computer Studies
Department of Computer Science, University of Maryland
College Park, MD 20742 USA
{shangfu, hjs, marco}@cs.umd.edu

ABSTRACT

Given the increasing prevalence of streaming spatially-referenced datasets resulting from sensor networks usually consisting of text objects of varying length (termed labels) as well as streaming spatially oriented queries leads to closer scrutiny of mapping interfaces to present the data to users. These interfaces must cope with the fact that the labels associated with each location are constantly changing and that there are too many objects to display clearly within the interface. An algorithm meeting these challenges is presented. It differs from classical methods by avoiding expensive pre-computation steps, thereby allowing different labels to be associated with locations without needing to completely recompute the layout. In other words, we are addressing a write-many read-many setting instead of the conventional write-once read-many setting. Our experiments show consistent sub-second query times for query windows that contain as many as 11 million data objects, with only slight differences in the set of displayed labels when compared to an exhaustive baseline algorithm. This enables the algorithm to be used in a mapping application that involves both streaming data and streaming queries such as windowing realized by real-time, continuous zooming and panning operations.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms

Algorithms, Design

Keywords

Dynamic map labeling, label selection, streaming data

*This work was supported in part by the NSF under Grants IIS-10-18475, IIS-12-19023, and IIS-13-20791 and by NVIDIA.

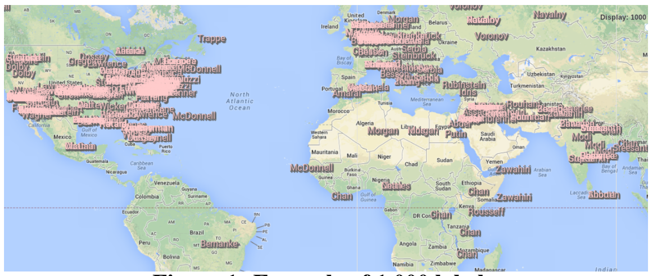
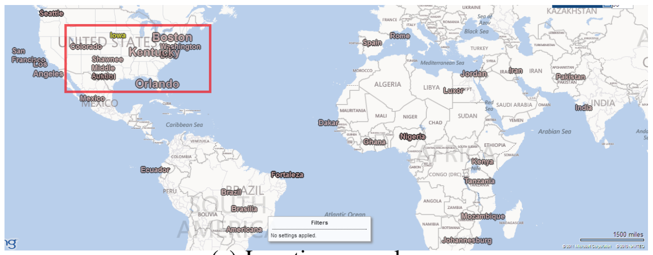


Figure 1: Example of 1,000 labels

1. INTRODUCTION

We have built many applications that use a map query interface (e.g., QUILT [25, 34] and the SAND Browser [7, 26]) to access and process data for which spatial and spatiotemporal indexes have been built in both conventional (e.g., [11, 12, 22, 24, 35]) and distributed (e.g., [36] environments). Unlike traditional paper maps (e.g., [3, 5, 6, 8, 38]), which only exist at a single scale for a point-in-time snapshot of a dataset, the interfaces for these applications must be dynamic in order to handle changes to the viewing parameters (i.e., usually with streaming pan and zoom actions), and, increasingly for many applications, changes to the underlying data, which are usually text objects of variable length (henceforth called *labels*). An example is a disease report browser which plots the names of diseases at locations where the diseases have been reported. In particular, the binding of the label to a location is dynamic as the data changes over time. A challenge arises when there are too many objects to display clearly within the map interface. Figure 1 gives an example scenario which consists of the task of displaying just $1,000$ text objects in a map. In these circumstances, a subset of objects must be selected from the original dataset to display so that users have a manageable amount of information (a task also known as "spatial sampling" [4] or "thinning"[33]). We use the term *layout problem* to describe the combined selection and display problem.

Our primary contribution is developing a method with a good time complexity bound for sampling and placing labels to represent large spatially-referenced datasets on a map. Our approach has two phases: filtering and intersection testing. In the filtering phase, we adapt the multiresolution select distinct (MRSD) method of Nutanong et al. [19, 20] for the centroids of the labels. It has been used in a system that made use of a map query interface for news photos [31] of a fixed size [9, 29]. The intersection testing phase handles dynamic objects of varying size by ensuring that the variable-width objects do not overlap. The effect of the streaming nature of the data is accommodated by constantly changing the data samples that were displayed and repeating the labeling action.


(a) Location name layer

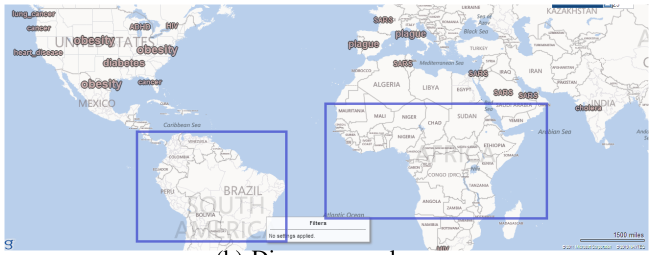
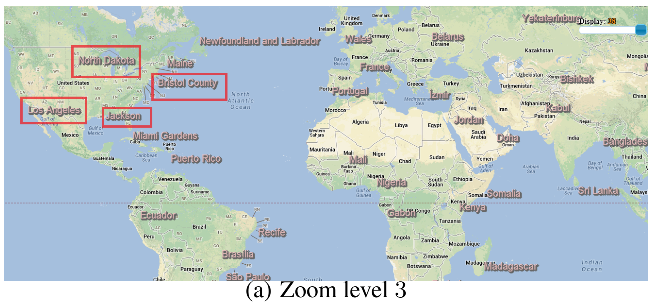

(b) Disease name layer

Figure 2: The greedy algorithm version in NewsStand


(a) Zoom level 3

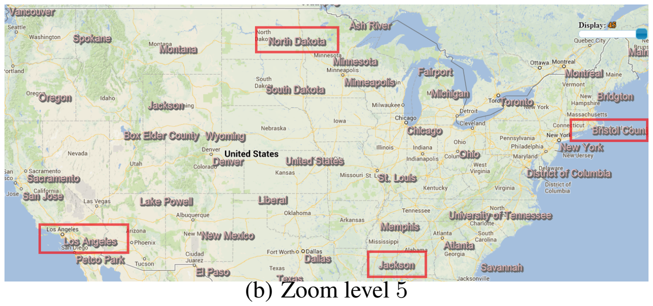

(b) Zoom level 5

Figure 3: The MIF algorithm version in NewsStand

Using our approach, each labeling action took no more than 0.3 seconds regardless of the zoom level. The panning was simulated by moving the query windows between consecutive queries.

The rest of this paper is organized as follows. Section 2 formalizes our problem and constraints, which are handled by the method that we present in Section 3. Section 4 provides an evaluation of our method's applicability to both synthetic and real-world datasets. Section 5 contains concluding remarks about our method and presents avenues for future work.

2. PROBLEM DEFINITION

A *sampled labeling* of a geographic dataset $\mathcal{D}$ is a subset of elements $S \subseteq \mathcal{D}$ that can be displayed within a query window $\mathcal{Q}$ without any overlap between elements of S. Each element of $\mathcal{D}$ has attributes <*lat*, *long*, σ, *text_str*>, where σ is a relative importance score and higher values indicate that an object is more likely to appear. Each text labels center is located at its actual point location, as shown in Figure 4, where the black circle is the location <*lat*, *long*> in the map.

Input: dataset $\mathcal{D}$, query window $\mathcal{Q}$.

Output: Subset $S \subseteq \mathcal{D}$ of data objects, such that all objects in S lie within $\mathcal{Q}$ in a way that satisfies the following criteria.

Maximize Importance Score. Each object has an importance score attribute σ. A higher score means that an object is more important and more desirable to display, so the first goal is:

$$\max_{S} \sum_{i \in S} \sigma_i. \qquad (1)$$

From previous work [5, 38], we know that an exact solution to the labeling problem that maximizes this type of global objective function is an NP-hard problem. So we aim to find an approximate optimal solution in a reasonable amount of time.

Avoid Overlaps. Figure 2 shows a sample screen from the existing version of *newsstand.umiacs.umd.edu*, a system for displaying news articles geographically [16, 27, 28, 30, 37] (see also the related TwitterStand system [10, 13, 32]). Its heart consists of a system for understanding textual specifications of location (e.g., [1, 2, 14, 15, 17, 18, 21]). The full NewsStand database contains more than eleven million entries over the past 12 months. The layout of this version uses a greedy algorithm that selects the most important 100 entries ordered by importance score. It does not perform exact intersection checking. So in the red rectangle of Figure 2(a), the label "Kentucky" overlaps the label "Washington D.C.", which is not desirable. The challenge of avoiding overlaps is the large amount of data entries in our database — specifically, there are 22 possible zoom level settings in the Google Maps API, ranging from 0 to 21. Furthermore, the data entries in the database may be *dynamic*, which means that labels may be added, deleted, or changed at will.

Maximize Spatial Fullness. The subset S should cover most of the area in the viewing window. Comparing Figures 2 and 3, Figure 3 is more uniform and informative since the labels are present in many areas of the map, while in Figure 2 text labels are concentrated in the United States, and there are no labels displayed in the blue rectangles of Figure 2(b). Figure 3 shows a proposed redesign of NewsStand, which implements our method. In summary, maximizing spatial fullness requires the algorithm to consider the spatial relative importance score, not only the global importance score. This means that some important entries may be hidden by other more important entries, and some minor entries may be visible since there are no other entries around it.

Zooming and Panning Consistency. This criteria is a core requirement mentioned in the work of Das Sarma et al. [33] from Google. If a geographic window W contains a visible text label R, and another window W', obtained by zooming in, also contains R, then R should also be sampled in W', which means that R should be visible in W'. Figure 3(b) is obtained from Figure 3(a) by two zoom-in actions. Figure 3(b) shows the United States map. To achieve zooming consistency, all the visible entries in Figure 3(a) and within the United States, which are the red markers "Los Angeles", "North Dakota", "Jackson", and "Bristol Country", should also be visible in Figure 3(b).

Perform Efficiently over Dynamic Data. As mentioned earlier,

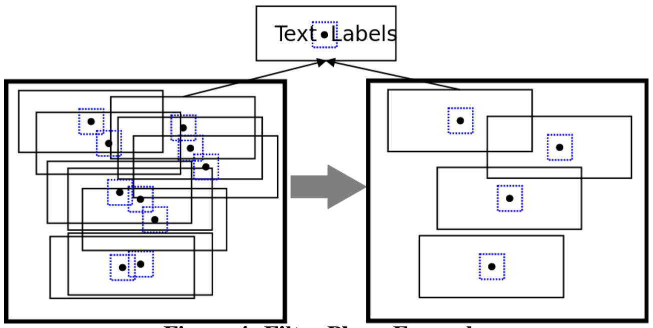


Figure 4: Filter Phase Example

there are two aspects of "dynamic" data that must be addressed. The first aspect involves the data entries, which can be added or deleted from the dataset, e.g., news update in NewsStand system. A second "dynamic" aspect involves the text labels. In many cases, there are multiple different text labels that can represent the same data entry. NewsStand is a good example — users can choose between a Keyword layer, Location layer, People layer, and Disease layer. For each layer, a different label can represent the same news story. Figure 2(b) displays the Disease layer while other figures show the Location layer. The difficulty is that the length is varying between the different text labels associated with the same location. In this dynamic case, traditional algorithms do not suffice since they focus on a static environment.

3. METHOD

Our method named Morton Index Filter (MIF) algorithm involves a filtering phase using multi-resolution select distinct (MRSD) query to retrieve objects with high distinctiveness scores within the query window, followed by a intersection testing phase to ensure that displayed objects do not overlap. From Figure 1, we know that among millions of entries, most labels overlap each other and should not be displayed. The intuition of our algorithm is to first find all the labels that are the most important in a certain small area around their associated locations. Figure 4 gives a good example. The small blue dashed squares are the same size for all text labels. After the filtering phase, each existing text label is the most important one within its blue dashed square.

3.1 Filtering Phase for Text Labels

Our method employs the MRSD query of Nutanong et al. [19] to perform a first-pass sampling of the collection of data objects within a query window. The MRSD method decomposes the data space into regions, and each region votes for the label within it that has the highest importance value. Using only a single decomposition will mean that a pair of nearby objects separated by a region boundary might both be voted for, even though they are likely to overlap. So we instead use 9 overlaid decompositions, obtained by offsetting the square regions of a grid by $\frac{1}{3}$ and $\frac{2}{3}$ of the grid width in horizontal and vertical directions.

The regular decomposition used for the MRSD method is applied to points projected using the standard Mercator projection with a square truncation to match the extent of online mapping applications, using the unit square ($[0, 1] \times [0, 1]$) as the coordinate system. The decomposition is a regular Morton decomposition [23] where each square block can be referenced by its Morton code. So the MRSD method can be implemented within an RDBMS using standard B-Tree indexes. Adding a data object to the collection involves computing the Morton code for its containing block at the deepest level, for each of the 9 offsets. When the system requires a sample of objects within a query window, it first determines the depth of decomposition to use when determining distinctiveness, and then identifies the objects with the highest distinctiveness scores. For a given query window, the MRSD method expects objects with a size of $(\epsilon \times \epsilon)$ to be displayed (i.e., each object will have a square representation on the map). Indeed, the result of the query is a row representing the most important data object for every block in the window. By counting the number of times each object appears in this result, we get the distinctiveness score of each object, a number between 1 and 9. By construction, objects with a size of $(\epsilon \times \epsilon)$ and with a distinctiveness score of 9 will not overlap any other objects. Consequently, we can treat MRSD as a filter.

3.2 Intersection Testing Phase

The filtering phase yields all objects of maximum size $(\epsilon \times \epsilon)$ that do not overlap. As we are dealing with labels of arbitrary size, the actual labels may be bigger than the $(\epsilon \times \epsilon)$ objects and we need an intersection testing phase to check for overlaps since the size of the labels can vary. Here we introduce three methods to implement this phase.

In the text output layers of NewsStand, larger font point sizes mean using much space and thus the number of results returned by the filter step is smaller. When this number is on the order of hundreds (i.e., small), then it is efficient to use an $O(nm)$ method to check for overlaps, where n is the number of entries that are returned by the filtering step, and m is the maximum number of entries that can be displayed in a window, which we call the *window capacity*. However, if the font point size become smaller or the size of filter square decreases (the blue dashed square in Figure 4, then the size of the filter result would increase to thousands or more, which causes the $O(nm)$ algorithm to be less practical.

One straightforward approach to speed up the intersection phase, is to limit the overlap check to a small subset of neighboring objects (e.g., 5) so that a negative is reported if no overlap is detected.

However, since we are dealing with labels that are rectangle objects rather than points, we may miss some overlaps. We avoid this by sorting the objects with a spatial index of which many choices are available and we choose the PR-CIF quadtree [23]. We construct it by inserting the objects in decreasing order of the importance score and only making the insertion if the new object does not overlap any of the existing objects in the tree.

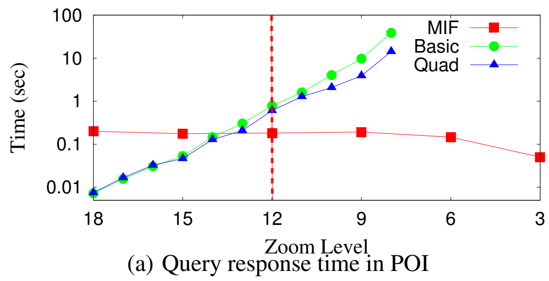
4. EVALUATION

We compare three text label layout methods:

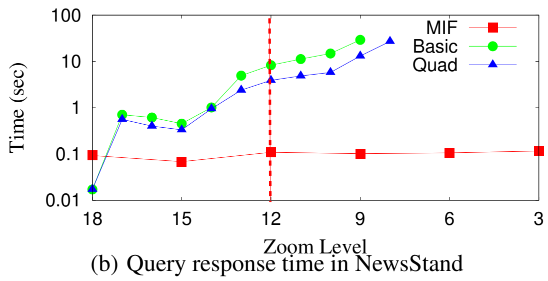
- Our Morton Index Filter method (MIF).
- The $O(nm)$ brute-force method (Basic), where n is the number of labels in the window being queried, and m is the maximum number of non-overlapping labels in the window.
- The PR-CIF quadtree based method (Quad).

We used two datasets, one of points of interest (POI) from *cloudmade.com* and one of news keywords from *newsstand.umiacs.umd.edu*, which contain $11,469,485$ and $9,964,607$ labels, respectively, from all over the world. The methods were evaluated using a realistic set of 5475 query windows from *newsstand.umiacs.umd.edu* query logs. Figure 5 plots the query time for zoom levels from 18 to 3, although most queries were made at zoom levels $9-4$ and thus the response time in this range is the most important. Deeper zoom levels are needed when the font point size is smaller. The red broken line in Figure 5 reinforces this point in that the part of the curve to its right is more important than the part to its left.

The query response time of MIF includes three parts: converting a window query into a number of SQL statements, retrieving data using Morton code indexes, and discarding overlaps. The second



(a) Query response time in POI



(b) Query response time in NewsStand

Figure 5: Efficiency evaluation

step is the filtering phase, which consumes the most time. The third step, which checks intersection, can be implemented using either the $O(nm)$ brute-force method or a spatial index such as the PR-CIF quadtree, and its execution time depends on the size of the filter square (i.e., the query window and the font point size). For the Basic and Quad methods, we first use the PostGIS extension to select all of the labels in the current window query. Since it retrieves a large amount of data entries, the query response time can be large for query windows that contain many labels, and likewise for the step to detect overlaps. Note that Figure 5 does not show any data for the Basic and Quad methods for zoom levels 8 to 3 as it takes too long to process queries for this range. Its clear that the efficiency of MIF is stable, while the other methods (Basic and Quad) exhibit response times that are roughly proportional to the number of entries in the current window.

5. CONCLUDING REMARKS

We have presented the Morton Index Filter (MIF) algorithm for efficiently sampling variable-sized data objects, such as text labels, for display within a map query window. The algorithm satisfies many important criteria for interactive map labeling. The core of our algorithm is a set of indexes on the Morton code representation of each object's location, which allows for extremely fast identification of distinctive objects. One particularly novel aspect of our algorithm is how it handles variable-width labels. Our evaluation shows that these objectives are achieved with consistent sub-second response time at all zoom levels, even on datasets with tens of millions of points in the query window.

6. REFERENCES

[1] M. D. Adelfio and H. Samet. GeoWhiz: Using common categories for toponym resolution. In P. Widmayer, editors, *GIS*, pages 542–545, Orlando, FL, 2013.
[2] M. D. Adelfio and H. Samet. Structured toponym resolution using combined hierarchical place categories. In *GIR*, pages 49–56, Orlando, FL, Nov. 2013.
[3] P. K. Agarwal, M. J. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998.
[4] K. Been, E. Daiches, and C.-K. Yap. Dynamic map labeling. *IEEE TVCG*, 12 (5):773–780, Sep–Oct, 2006.
[5] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM TOGS*, 14(3):203–232, 1995.
[6] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. E. Moret, and B. Zhu. Map labeling and its generalizations. In *SODA*, pages 148–157, New Orleans, LA, 1998.
[7] C. Esperança and H. Samet. Experience with SAND/Tcl: a scripting tool for spatial databases. *Journal of Vis. Lang. and Comput.*, 13(2):229–255, 2002.
[8] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *SOCG*, pages 281–288, North Conway, NH, USA, 1991.
[9] B. C. Fruin, H. Samet, and J. Sankaranarayanan. Tweetphoto: photos from news tweets. In *GIS*, pages 582–585, Redondo Beach, CA, 2012.
[10] N. Gramsky and H. Samet. Seeder finder - identifying additional needles in the Twitter haystack. In *LBSN*, pages 44–53, Orlando, FL, 2013.
[11] G. R. Hjaltason and H. Samet. Speeding up construction of PMR quadtree-based spatial indexes. *VLDBJ*, 11(2):109–137, 2002.
[12] E. G. Hoel and H. Samet. Benchmarking spatial join operations with spatial output. In *VLDB*, pages 606–618, Zurich, Switzerland, 1995.
[13] A. Jackoway, H. Samet, and J. Sankaranarayanan. Identification of live news events using Twitter. In *LBSN*, pages 25–32, Chicago, 2011.
[14] M. D. Lieberman and H. Samet. Multifaceted toponym recognition for streaming news. In *SIGIR*, pages 843–852, Beijing, China, 2011.
[15] M. D. Lieberman and H. Samet. Adaptive context features for toponym resolution in streaming news. In *SIGIR*, pages 731–740, Portland, OR, 2012.
[16] M. D. Lieberman and H. Samet. Supporting rapid processing and interactive map-based exploration of streaming news. In *GIS*, pages 179–188, Redondo Beach, CA, 2012.
[17] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *ICDE*, pages 201–212, Long Beach, CA, 2010.
[18] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR*, article 6, Zurich, Switzerland, 2010.
[19] S. Nutanong, M. D. Adelfio, and H. Samet. Multiresolution select-distinct queries on large geographic point sets. In *GIS*, pages 159–168, Redondo Beach, CA, 2012.
[20] S. Nutanong, M. D. Adelfio, and H. Samet. An efficient layout method for a large collection of geographic data entries. In *EDBT*, pages 717–720, Genoa, Italy, 2013.
[21] G. Quercini, H. Samet, J. Sankaranarayanan, and M. D. Lieberman. Determining the spatial reader scopes of news sources using local lexicons. In *GIS*, pages 43–52, San Jose, CA, 2010.
[22] H. Samet. A quadtree medial axis transform. *CACM*, 26(9):680–693, Sep. 1983. Also see CORRIGENDUM, *CACM*, 27(2):151, 1984.
[23] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006. (Translated to Chinese ISBN 978-7-302-22784-7).
[24] H. Samet and M. Tamminen. Bintrees, CSG trees, and time. *Computer Graphics*, 19(3):121–130, 1985.
[25] H. Samet, A. Rosenfeld, C. A. Shaffer, and R. E. Webber. A geographic information system using quadtrees. *Pattern Recognition*, 17(6):647–656, 1984.
[26] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin. Use of the SAND spatial browser for digital government applications. *CACM*, 46(1):63–66, 2003.
[27] H. Samet, M. D. Adelfio, B. C. Fruin, M. D. Lieberman, and B. E. Teitler. Porting a web-based mapping application to a smartphone app. In *GIS*, pages 525–528, Chicago, 2011.
[28] H. Samet, B. E. Teitler, M. D. Adelfio, and M. D. Lieberman. Adapting a map query interface for a gesturing touch screen interface. In *WWW (Companion Volume)*, pages 257–260, Hyderabad, India, 2011.
[29] H. Samet, M. D. Adelfio, B. C. Fruin, M. D. Lieberman, and J. Sankaranarayanan. PhotoStand: a map query interface for a database of news photos. *PVLDB*, 6(12):1350–1353, 2013.
[30] H. Samet, J. Sankaranarayanan, M. D. Lieberman, M. D. Adelfio, B. C. Fruin, J. M. Lotkowski, D. Panozzo, J. Sperling, and B. E. Teitler. Reading news with maps by exploiting spatial synonyms. *Communications of the ACM*, 57(10), Oct. 2014.
[31] J. Sankaranarayanan and H. Samet. Images in news. In *ICPR*, pages 3240–3243, Istanbul, Turkey, 2010.
[32] J. Sankaranarayanan, H. Samet, B. Teitler, M. D. Lieberman, and J. Sperling. TwitterStand: News in tweets. In *GIS*, pages 42–51, Seattle, WA, 2009.
[33] A. D. Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Y. Halevy. Efficient spatial sampling of large geographical tables. In *SIGMOD*, pages 193–204, Scottsdale, AZ, 2012.
[34] C. A. Shaffer, H. Samet, and R. C. Nelson. QUILT: a geographic information system based on quadtrees. *IJGIS*, 4(2):103–131, 1990.
[35] R. Sivan and H. Samet. Algorithms for constructing quadtree surface maps. In *SDH*, volume 1, pages 361–370, Charleston, SC, 1992.
[36] E. Tanin, A. Harwood, and H. Samet. A distributed quadtree index for peer-to-peer settings. In *ICDE*, pages 254–255, Tokyo, Japan, 2005.
[37] B. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS*, pages 144–153, Irvine, CA, 2008.
[38] F. Wagner and A. Wolff. An efficient and effective approximation algorithm for the map labeling problem. In *ESA*, vol. 979 of Springer-Verlag Lecture Notes in Computer Science, pages 420–433, Corfu, Greece, 1995.