Multi-Level Filtering to Retrieve Similar Trajectories under the Fréchet Distance

Hong Wei[1], Riccardo Fellegara[2], Yin Wang[3], Leila De Floriani[2] and Hanan Samet[1]
[1]Department of Computer Science, University of Maryland, College Park, MD 20742
[2]Department of Geographical Sciences, University of Maryland, College Park, MD 20742
[3]Department of Computer Science, Tongji University, Shanghai, China
[1]{hyw, hjs}@cs.umd.edu [2]{deflo, felle}@umd.edu [3]yinw@tongji.edu.cn

ABSTRACT

Computing with trajectories has become an important and practical research topic. In many scenarios, the goal is to find similar trajectories. The Fréchet distance is a very promising metric for measuring trajectory similarity and yet limited in practical applications due to its expensive computing complexity. In this paper, we demonstrate an efficient approach to retrieve similar trajectories using the Fréchet distance. Essentially, the proposed method builds up a set of R-trees for indexing trajectories and thereby enables multi-level of positive and negative filtering to speed up the similarity queries. For answering 5,000 queries on a dataset of 20,000 trajectories, the experimental results show that the proposed method achieves significant speedups at certain filtering levels while maintaining very high precision and recall in retrieving similar trajectories.

CCS CONCEPTS

•**Theory of computation →Computational geometry;**

KEYWORDS

Fréchet Distance, Similar Trajectories

ACM Reference format:
Hong Wei, Riccardo Fellegara, Yin Wang, Leila De Floriani and Hanan Samet. 2018. Multi-Level Filtering to Retrieve Similar Trajectories under the Fréchet Distance In *Proceedings of 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, November 6–9, 2018 (SIGSPATIAL '18),* 4 pages.
DOI: 10.1145/3274895.3274978

1 INTRODUCTION

Recently, the rising popularity of GPS embedded devices has generated an enormous amount of trajectory data. Thus efficiently generating [1], querying [2–8] and matching [9–11] these trajectories to extract relevant information has become an important and practical research topic. For example, the rapid growth of ride sharing requires that similar-path queries between different users must be answered in real-time for effective and efficient carpools.

Many geometric definitions have been proposed and studied to measure trajectory similarity [12–15]. Among these, the Fréchet distance has been particularly interesting due to its parameterized trajectory representation. Although the Fréchet distance [14] has revealed extensive advantages in many applications, like map matching [16–18], its computation remains challenging. For example, considering two polygonal trajectories P and Q with p and q segments, respectively, the state-of-the-art solution for computing their Fréchet distance has a time complexity of $O(pq\log(pq))$ [19]. The major overhead lies in one of its subprocess, called *decision problem*, which takes $O(pq)$ to compute whether the two polygonal curves are within a given distance threshold. In this paper, we focus on the query problem defined in the ACM SIGSPATIAL GIS Cup 2017[1]. Given a set of polygonal trajectories $\mathcal{D}$, a query trajectory Q and a distance threshold ε, we retrieve all trajectories $\mathcal{P} \subseteq \mathcal{D}$ such that each returned trajectory $P \in \mathcal{P}$ satisfies $\delta_F(P, Q) \leq \varepsilon$, where $\delta_F(P, Q)$ denotes the Fréchet distance between P and Q.

This paper exploits four observations with respect to the calculation of the decision problem. First, if two polygonal curves have a Fréchet distance less than or equal to ε, the distance between their respective starting and ending vertices must also be less than or equal to ε. Second, for each vertex in a polygonal curve, there must exist, in its circular range of a radius ε, at least one line segment from another polygonal curve. Third, it has been proven that the Fréchet distance is within ε if the discrete Fréchet distance [20] of the two polygonal trajectories is not larger than ε. The discrete Fréchet distance is an approximation of the Fréchet distance by considering the positions only at the vertices of trajectories. Computing this discrete variant takes $O(pq)$ [20]. Fourth, the discrete Fréchet distance is greater than or equal to the Fréchet distance, but interpolating extra points on the trajectories as new vertices may result in a lower value of the discrete Fréchet distance.

For each query, we build multiple levels of negative and positive filtering to reduce the number of candidate trajectories. Our contribution is summarized as follows:

- We generate an efficient two-level R-tree for indexing the start/end points and segments of trajectories. As a result, this data structure yields a significant reduction in the number of candidates to consider for answering a query.
- We exploit a positive filter by using the discrete Fréchet distance as an upper bound of the Fréchet distance. This filter further reduces the number of uncertain candidate trajectories to only 2% in our experimental evaluation.
- We innovatively interpolate points on a query trajectory and its candidate trajectories, which reduces the number of the uncertain candidate trajectories.

[1]http://sigspatial2017.sigspatial.org/giscup2017/home

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGSPATIAL '18, Seattle, WA, USA
© 2018 Copyright held by the owner/author(s). 978-1-4503-5889-7/18/11…$15.00
DOI: 10.1145/3274895.3274978

2 RELATED WORK

A lot of studies have been developed to solve the Fréchet distance computation problem due to its academic and industrial potential [16, 18–21]. However, directly computing this distance metric between polygonal trajectories is very time-consuming [19, 22]. Therefore, some work focused on solving variants of the problem [21, 23–25]. For example, F. Cook Iv and Wenk [26] describe a polynomial-time algorithm to compute the geodesic Fréchet distance between two polygonal curves in a simple polygon. Chambers et al. [25] describe a polynomial-time algorithm to compute the homotopic Fréchet distance between polygonal curves in the Euclidean plane with obstacles. The original definition of the Fréchet distance continuously sweeps every point on the given curves including the interior points on the edges. If we relax this requirement by only examining the vertices positions of polygonal curves, we have the so-called discrete Fréchet distance. Eiter and Mannila [20] have presented a polynomial-time solution using dynamic programming for computing the discrete Fréchet distance

The most related work are those presented at ACM SIGSPATIAL GIS Cup 2017, which is a competition on efficiently retrieving similar trajectories from a large set of data. Baldus and Bringmann [27] propose a three-phase algorithm to solve the competition problem. They first build a quadtree like data structure to enumerate the candidate trajectories whose starting and ending points are close to the query trajectory. Then, they apply several heuristics to find the true positive and negative candidates by only examining the distances between vertices from two trajectories. Finally, they run an exact Fréchet distance decision problem to finalize the results. Similar to [27], Buchin et al. [28] also start with identifying trajectories whose starting and ending points are within ε to the ones of the query trajectory but using a spatial hash instead, which is essentially grid division. Next, they compute several levels of simplification of trajectories. In each level, they calculate the equal-time distance to find positive candidates, or further apply the decision procedure to finalize the results if necessary. Dütsch and Vahrenhold [29]'s method also exploits the strategy of filter-and-refinement, but it uses minimum bounding boxes to first exclude negative candidates. Then, it examines the distance between starting points and ending points between pair of trajectories and, finally, uses the decision procedure to get the exact results.

Differently from the above methods, we: (i) generate an efficient two-level hierarchical R-tree for indexing the start/end points and segments of trajectories; (ii) exploit a positive filter based on the discrete Fréchet distance; and (iii) interpolate points on a query trajectory and its candidate trajectories.

3 PRELIMINARIES

3.1 The Fréchet Distance

Let P be a polygonal curve in $\mathbb{R}^2$, a continuous and piecewise linear mapping from the interval $[0, p]$ into $\mathbb{R}^2$, where p is the number of segments formed its vertex sequence $\langle u_0, u_1 \cdots u_p \rangle$. This curve can be seen as parameterized by a point in the interval $[0, p]$. Similarly, let Q be another polygonal curve with the vertex sequence $\langle v_0, v_1 \cdots v_q \rangle$ and the mapping range $[0, q]$. Formally, the Fréchet distance between two trajectories $P : [0, p] \rightarrow \mathbb{R}^2,\ Q : [0, q] \rightarrow \mathbb{R}^2$ is defined as

$$\delta_F(P, Q) := \inf_{\substack{\alpha:[0,1]\rightarrow[0,p] \\ \beta:[0,1]\rightarrow[0,q]}} \max_{t\in[0,1]} \|P(\alpha(t)) - Q(\beta(t))\|$$

where α and β are continuous and non-decreasing time-warping functions with $\alpha(0) = \beta(0) = 0$, $\alpha(1) = p$, and $\beta(1) = q$, which may be seen as re-parameterizations of the curves P and Q. Therefore, calculating the Fréchet distance consists of finding an infimum over all possible re-parameterizations α and β. A popular intuitive definition of the Fréchet distance between two curves is the minimum length of a leash required to connect a dog and its walker, constrained on two separate paths, as they walk without backtracking along respective curves from one endpoint to another.

3.2 Computing the Fréchet Distance

A novel polynomial time algorithm to determine the Fréchet distance between two polygonal curves was first proposed by Alt and Godau [19]. The core part of this algorithm is a decision problem and a set of critical values for ε.

3.2.1 Decision Problem. For a given distance threshold ε, the decision problem decides whether $\delta_F(P, Q) \leq \varepsilon$. In order to solve this, the algorithms builds the *free space diagram* of P and Q with respect to ε.

The *free space* of two polygonal curves $P : [0, p] \rightarrow \mathbb{R}^2,\ Q : [0, q] \rightarrow \mathbb{R}^2$ is defined as:

$$F_\varepsilon(P, Q) := \{(s, t) \in [0, p] \times [0, q] \mid \|P(s) - Q(t)\| \leq \varepsilon\} \quad (1)$$

Region $[0, p] \times [0, q]$ is therefore partitioned into free space and non-free space. The free space of two line segments is a full or partial ellipse, and the free space diagram of two polygonal curve of p and q segments is a $p \times q$ segment-segment free space diagram [19]. It has been shown that $\delta_F(P, Q) \leq \varepsilon$ if and only if there exists a path within $F_\varepsilon(P, Q)$ from the lower left corner $(0, 0)$ to the upper right corner (p, q), which is monotone in both coordinates [19]. This path induces functions α and β in the definition of $\delta_F(P, Q)$. Constructing the free space diagram and determining the existence of a monotone path for two polygonal curves takes $O(pq)$.

3.2.2 Critical Values. After solving the decision problem, a binary search is performed on a set of candidate ε values which the decision procedure examines whether they are larger or smaller than the Fréchet distance. The set of candidate ε values to perform search is called *critical values*, which represent the cases where the existence of a monotone path occurs. There are three cases of such critical values, namely:

(1) The distances between the two starting points and the two ending points of P and Q, respectively.
(2) The distances from a vertex in one curve to line segments of the other.
(3) The common distance of two vertices of one curve to the intersection point of their bisector with some line segment of the other.

Together, there are $O(p^2q + q^2p)$ critical values in total.

3.3 The Discrete Fréchet Distance

The discrete Fréchet distance [20], also known as the coupling distance, is an upper bound of the continuous Fréchet distance. Differently from the Fréchet distance, that considers the interior points on each segment of a polygonal curve, the discrete Fréchet distance addresses only the vertex positions at the two polygonal curves. Eiter and Mannila [20] give a dynamic programming algorithm that calculates the discrete Fréchet distance with $O(pq)$ time complexity. More importantly, they have shown that:

$$\delta_F(P,Q) \leq \delta_{dF}(P,Q) \leq \delta_F(P,Q) + max\{D(P), D(Q)\} \quad (2)$$

where $\delta_{dF}(P,Q)$ is the discrete Fréchet distance between P and Q, and $D(P)$ and $D(Q)$ refer to the maximal lengths of the line segments in P and Q, respectively. This is an important theorem as it guarantees the Fréchet distance is within ε if the $\varepsilon \geq \delta_{dF}(P,Q)$.

4 METHOD

Given a query trajectory $Q = \langle u_0, u_1 \cdots u_q \rangle$ and a candidate trajectory in the database $P \in \mathcal{D}$ with vertices $\langle v_0, v_1 \cdots v_p \rangle$, in order to determine whether $\delta_F(P,Q) \leq \varepsilon$, our method uses multiple levels of positive filtering and negative filtering as follows.

- Level 0: P is a negative candidate if $\varepsilon \leq max(\|u_0, v_0\|, \|u_q, v_p\|)$, which implies $\delta_F(P,Q) > \varepsilon$. It corresponds to the first type of critical values described in Section 3.2.2.

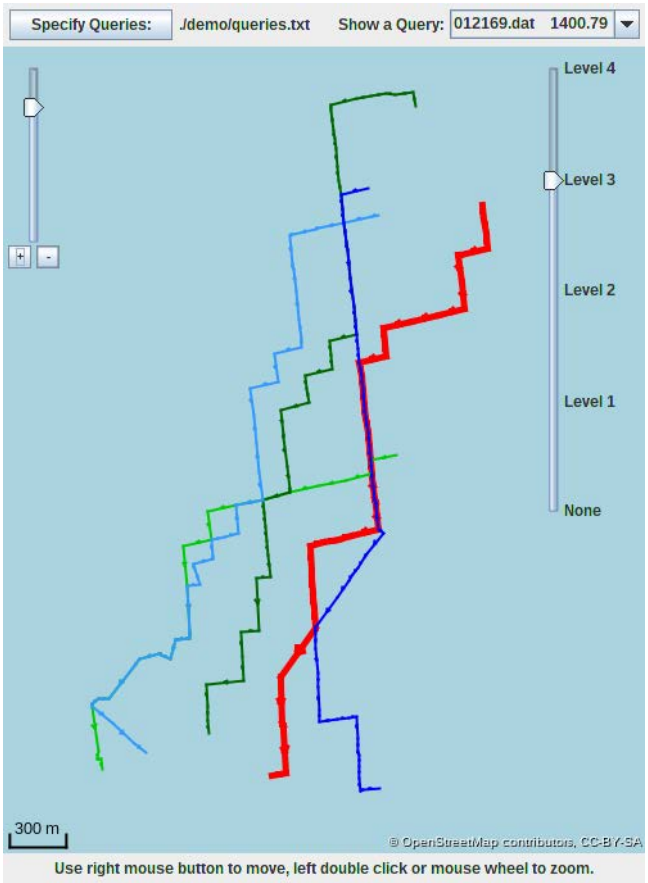

Figure 1: A screenshot of multi-level filtering.

- Level 1: P is a negative candidate if either of the following two conditions are not satisfied: i) for each vertex u in Q, there exists at least one line segment v_iv_j in P such that the distance from u to v_iv_j is within ε; ii) for each vertex v in P, there exist at least one line segment u_iu_j in Q such that the distance from v to u_iu_j is within ε. This is a negative filtering and corresponds to the second type of critical values described in Section 3.2.2.
- Level 2: From Equation 2, we can deduce that $\delta_F(P,Q) \leq \varepsilon$ if $\delta_{df}(P,Q) \leq \varepsilon$, in which case P is a positive candidate; or $\delta_F(P,Q) > \varepsilon$ if $\delta_{dF}(P,Q) - max\{D(P), D(Q)\} \geq \varepsilon$, in which case P is a negative candidate. We implement the dynamic programming solution in [20] to calculate the discrete Fréchet distance. This is a composite level which contains both positive and negative filtering.
- Level 3: After interpolating extra points along the line segments of Q and P at every distance of ε, we apply the filtering in Level 2 again on Q' and P', which denote the two new trajectories with extra points, respectively. The rationale is that the discrete Fréchet distance only addresses the positions of vertices on the trajectories, excluding the internal points. Therefore, interpolating extra points on the line segments of trajectories reduces the length of the longest segments, which are the upper bounds of the discrete Fréchet distance and therefore result in smaller discrete Fréchet distance.
- Level 4: We use the exact decision procedure in Section 3.2.1 to conclude whether $\delta_F(P,Q) \leq \varepsilon$.

Note that in order to support the filtering in Level 0 and Level 1, we construct a two-level R-tree [30] to index the trajectories in the database. The R-tree in the first level indexes the starting and ending points of trajectories in $\mathcal{D}$ so that, given the query trajectory Q, it only returns a subset trajectories of $\mathcal{D}$ whose starting (and ending) point is within ε distance to P's starting point. This is implemented by enlarging P's starting (and ending) point to a square with side length of ε when performing the search in the R-tree. Instead of indexing on vertices, the R-tree in the second level now indexes on the line segments of candidate trajectories and only needs to build for the trajectories who passes Level 0. Indexing the line segments of a trajectory in R-tree is implemented by inserting a minimum bounding box for each of its segments.

We implement the above multi-level filtering in C++. To better understand what types of trajectories got filtered out or kept in each level of filtering, we wrap the implementation with a Java GUI application, as illustrated in Figure 1. The bold red polygonal curve represents the query trajectory with its name and Fréchet distance threshold specified in the drop-down menu proceeded by a label "Show a Query". The other trajectories represent the output up to the current filtering level, which is set to "Level 3" in the slider located at the right side of the screenshot. The colors of trajectories distinguish overlapped trajectories in the drawing. In addition, the slider in the top-left corner is to zoom in/out the map. Above it is a file selector that chooses the file containing all the queries.

Implementation Remark: To enable fast read/write of trajectories from/to disks, we use memory map to read/write every n files in batch, where n is configured according to the hardware memory size. After that, a pool of threads are allocated to utilize the hardware concurrency throughout the multi-level of filtering.

5 EVALUATION

The trajectory dataset in GISCUP 2017 is used for the evaluation, which contains 20,198 trajectories in total. We randomly select 5,000 trajectories to form a query set: $\langle Q^i, \varepsilon^i \rangle$, $i = 1 \ldots 5,000$, where Q^i is query trajectory and the ε^i is its Fréchet distance threshold. Each ε^i is, thus, chosen from a set of random numbers uniformly distributed in the range of [123.40, 2099.86]. The experiment is performed on a computer with an Intel Xeon E5 CPU and a 64GB RAM.

Table 1: Evaluation Results on 5000 queries

Step	Output (pair of trajectories)			Time (ms)
	True Positive	Undecided	Total	
Load Data		100990000	100990000	91.33
Build R-tree of Starting and Ending Points				74.18
Level 0		10171	10171	206.99
Build R-tree of Line Segments				125.02
Level 1		9151	9151	683.67
Level 2	9031	120	9151	17.12
Level 3	9031	4	9035	23.14
Level 4	9035	0	9035	3700.09

Table 1 reports the results of processing 5000 queries. Assuming that each query trajectory and each of its positive candidates form a "trajectory pair", there are 9035 of such correct pairs in total. The column "True Positive" specifies how many of these trajectory pairs are found by a specific Level. The column "Undecided" shows how many candidate trajectory pairs remain to check whether they satisfy the query. The column "Time" highlights the computation time by each procedure or filtering level, and is reported as the mean time of running 5 iterations of the program.

The results show that the filtering at Level 0 is able to reduce significantly the number of candidate trajectory pairs to check. However, it is unable to give positive determination on the queries and thereby left its output undecided. The filtering in Level 1 improves slightly than Level 0 but takes much more time. The significant change happens at the filtering of Level 2 when it starts using discrete Fréchet distance to help determine which trajectories are true positives. The results show that it manages to verify most of the undecided trajectory pairs in Level 1 and only takes 17 ms. By interpolating extra points, the filtering in Level 3 further reduces the number of undecided trajectory pairs from 120 to 4, even though no positive candidate is found. The worst part happens at Level 4, where we only run the exact decision procedure for 4 pairs of trajectories and the consuming times significantly increases, almost four-times. It indicates that, without appropriate filtering, directly applying the decision problem to answer the queries will takes a considerably long time.

6 CONCLUSIONS

Our proposed idea of multi-level filtering to retrieve similar trajectories can achieve significant improvement on running time and meanwhile gets a very close output to the correct results, even without running the exact decision problem. In particular, we find that applying discrete Fréchet distance and using the given distance threshold for interpolation are two critical techniques for reducing the number of uncertain trajectories. There is, however, quite a few other filtering criteria that can be explored and incorporated to add more levels of refinement such as equal-time distance of trajectories [28], minimum and maximum coordinate values checking [31] and minimum bounding box expansion [29]. We leave the implementation of these criteria for the future work. Other future work includes the incorporation into a spatial browser [32–34].

7 ACKNOWLEDGEMENT

This work was supported in part by the NSF under Grants IIS-13-20791 and IIS-1816889.

REFERENCES

[1] J. Sankaranarayanan, H. Samet, and H. Alborzi. Path Oracles for Spatial Networks. PVLDB '09.
[2] S. Nutanong and H. Samet. Memory-efficient algorithms for spatial network queries. ICDE '13.
[3] S. Peng and H. Samet. CDO: Extremely High-throughput Road Distance Computations on City Road Networks. SIGSPATIAL '16.
[4] J. Sankaranarayanan, H. Alborzi, and H. Samet. Efficient Query Processing on Spatial Networks. SIGSPATIAL '05.
[5] J. Sankaranarayanan and H. Samet. Distance Oracles for Spatial Networks. ICDE '09.
[6] J. Sankaranarayanan and H. Samet. Query Processing Using Distance Oracles for Spatial Networks. TKDE '10.
[7] J. Sankaranarayanan and H. Samet. Roads Belong in Databases. *IEEE Data Eng. Bull.*, 33:4–11, 2010.
[8] S. Peng, H. Wei, H. Li, and H. Samet. Simplification and Refinement for Speedy Spatio-temporal Hot Spot Detection Using Spark. SIGSPATIAL '16.
[9] E. H. Jacox and H. Samet. Metric Space Similarity Joins. TODS '08.
[10] S. Nutanong, E. H. Jacox, and H. Samet. An Incremental Hausdorff Distance Calculation Algorithm. PVLDB '11.
[11] J. Sankaranarayanan, H. Alborzi, and H. Samet. Distance Join Queries on Spatial Networks. SIGSPATIAL '06.
[12] P. C. Besse, B. Guillouet, J. M. Loubes, and F. Royer. Review and Perspective for Distance-Based Clustering of Vehicle Trajectories. *IEEE TITS '16.*
[13] F. Hausdorff. *Grundzuge der Mengenlehre.* 1914.
[14] M. Fréchet. *Sur quelques points de calcul fonctionnel.* 1914.
[15] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. AAAIWS '94.
[16] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching Planar Maps. *J. Algorithms*, 49 (2):262–283, 2003.
[17] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-matching Vehicle Tracking Data. PVLDB '05.
[18] H. Wei, Y. Wang, G. Forman, and Y. Zhu. Map Matching: Comparison of Approaches Using Sparse and Noisy Data. SIGSPATIAL'13.
[19] H. Alt and M. Godau. Computing The Fréchet Distance Between Two Polygonal Curves. *Int. J. Comput. Geom. Appl.*, 05(1-2):75–91, 1995.
[20] T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical report, Technische Universitat Wie, 1994.
[21] K. Buchin, M. Buchin, and Y. Wang. Exact Algorithms for Partial Curve Matching via the Fréchet Distance. SODA '09.
[22] K. Bringmann. Why Walking the Dog Takes Time: Fréchet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. FOCS '14.
[23] K. Bringmann and W. Mulzer. Approximability of the Discrete Fréchet Distance. SoCG '15.
[24] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet Distance for Realistic Curves in Near Linear Time. SoCG '10.
[25] E. W. Chambers, E. Colin De Verdière, J. Erickson, S. Lazard, F. Lazarus, and S. Thite. Homotopic Fréchet Distance Between Curves or, Walking Your Dog in the Woods in Polynomial Time. *SoCG '08.*
[26] A. F. Cook Iv and C. Wenk. Geodesic Fréchet Distance With Polygonal Obstacles. Technical report, 2008.
[27] J. Baldus and K. Bringmann. A Fast Implementation of Near Neighbors Queries for Fréchet Distance (GIS Cup). SIGSPATIAL'17.
[28] K. Buchin, Y. Diez, T. van Diggelen, and W. Meulemans. Efficient Trajectory Queries Under the Fréchet Distance (GIS Cup). SIGSPATIAL'17.
[29] F. Dütsch and J. Vahrenhold. A Filter-and-Refinement-Algorithm for Range Queries Based on the Fréchet Distance (GIS Cup). SIGSPATIAL'17.
[30] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. SIGMOD '84.
[31] J. Baldus and K. Bringmann. A Fast Implementation of Near Neighbors Queries for Fréchet Distance (GIS Cup). SIGSPATIAL'17.
[32] F. Brabec and H. Samet. Client-Based Spatial Browsing on the World Wide Web. *IEEE Internet Computing*, 11(1):52–59, 2007.
[33] C. Esperança and H. Samet. Experience with SAND-Tcl: A Scripting Tool for Spatial Databases. DG.O '00.
[34] 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. *Commun. ACM*, 46(1):61–64, 2003.