Cost-Effective Approximate Aggregation Queries on Geospatial Big Data

Isam Mashhour Al Jawarneh[1], Rebecca Montanari [2] , Antonio Corradi [2]

[1] *Department of Computer Science, University of Sharjah, P.O.Box. 27272 Sharjah, United Arab Emirates*

[2]*Dipartimento di Informatica – Scienza e Ingegneria, University of Bologna*
Viale Risorgimento 2, 40136 Bologna, Italy
ijawarneh@sharjah.ac.ae, rebecca.montanari@unibo.it, antonio.corradi@unibo.it

***Abstract*— Aggregation queries are essential in spatial data analytics, including Top-N, and geo-statistics such as 'mean' and 'count'. Those queries require grouping geospatial objects into pre-defined clusters that are typically administrative polygons representing study areas such as cities. Given a big georeferenced dataset on the order of millions, and a group of polygons representing a city, the aggregation query requires grouping objects by polygons and determining to which polygon each object belongs. This is a computationally expensive geospatial operation because polygons are typically represented by huge amounts of vertices. In this paper, we show the design and realization of a system that we term ApproxGeoAgg for the efficient approximation of costly geospatial aggregate queries that require group-by operations. We have performed extensive testing, and our results show that our system outperforms plain baselines by order-of-magnitude in terms of balancing running times with accuracy. Specifically, for Top-N aggregation queries we obtain tiny loss in accuracy that reaches 0.00038% depending on parameter configurations, with a corresponding gain in running time on par with 2.6%, which escalates to circa 12% as we decrease the number of polygon boundary vertices.**

Keywords*— *spatial approximate query processing, geospatial group-by, spatial aggregation, Douglas Peucker, line simplification

I. Introduction

Georeferenced locational data are nowadays available on the scale of terabytes and slated to be double those numbers in the next decade or so [1, 2]. Examples include data that is generated from GPS sensors in handheld devices, in addition to urban planning and transportation data from smart cities [3]. Scientists seek to analyze this data to explore the opportunities for better decision making in urban planning and transportation design decisions. However, analyzing such deluges of data is becoming increasingly more challenging and presenting analysts with tremendous challenges as the data increase in size and heterogeneity.

The notion of 3Vs of big data is very common for georeferenced data streams. Georeferenced data is very big, arriving in fast data streams and from heterogeneous data sources in various formats and data representations. Those 3Vs, and more, indigenously rationale the challenges that georeferenced data analysis is bringing to dynamic smart city application scenarios nowadays. Those challenges span any workflow of a pipeline that encompasses geospatial queries as an indispensable part of its design. This includes data representation, reduction, ingestion, storage, indexing, and processing.

In this paper, we focus on a specific type of geospatial data analytics. We specifically aim at optimizing aggregation queries over big geospatial data. Those queries such as Top-N ensemble queries and geo-statistics queries (such as 'mean' and 'count') that require group-by geospatial operations. Consider a georeferenced big data consisting of millions of spatial tuples tagged with locational labels (typically on the form of longitudes/latitudes), in addition to a big file containing polygons representing the study area (be that as it may, cities, countries, etc.,), where each polygon represents part of the study area (e.g., each polygon is a district in a metropolitan city). An aggregation query typically seeks to find the total number of spatial tuples (i.e., objects) that fall within the boundaries of each polygon in the file. Those polygons are typically non-overlapping and are represented by boundaries that consist of big number of connected vertices such as the one shown in Figure 1 for the city of Rome in Italy. Aggregating data by pre-defined polygons representing study areas are a kind of common sense in various domains such as social science and environmental science.

The canonical approach for performing aggregation queries on predefined polygons relies on an approach that is known as filter-and-refinement [4]. As its name implies, this approach comprises two main steps. In the first step, Minimum Bounding

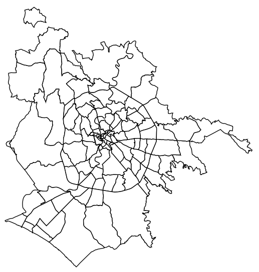


Fig. 1 Polygons file heuristic, city of Rome, Italy

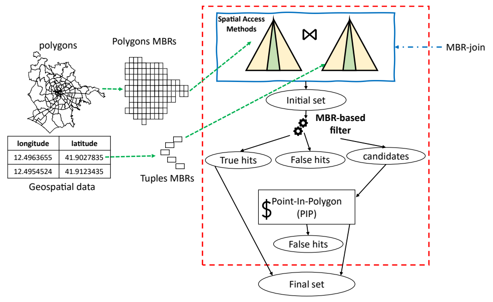


Fig. 2 Filter-Refinement spatial join approach heuristic overview

Rectangles (MBRs) are generated for each polygon in the boundaries file, in addition to each spatial tuple in the locational data. Thereafter, a cheap equijoin-like type of join is performed the MBRs of both datasets (polygons projected with spatial tuples). Spatial tuples that have their MBRs fall completely inside one of the MBRs of one of the polygons from the polygons file are maintained, while those that have their MBRs intersect one of the MBRs overlaying the study area are subjected to further check (those known as candidate set), others that neither have their MBRs fall within MBRs of polygons nor have their MBRs intersect are discarded. Tuples in the candidate set are then subjected to exact geometries costly test that is known as Point-In-Polygon (PIP for short). This type of test entails a very high computationally expensive cost as it requires checking geometrically for each tuple whether it belongs to the candidate polygon are not [5]. Figure 2 shows a schematic depicting the mechanism by which filter-and-refinement approach works. This approach is widely employed in modern geospatial data management systems, including those that operate in distributed computing such as GeoSpark [6] and GeoMesa [7]. However, its application entails high computational costs in cases where the polygon files are excessively large. This is so because PIP is a kind of containment geospatial queries that incurs high cost in their plain versions, as they rely basically on traditional algorithms such as ray-casting algorithm. Having said that, this hurdle stands as a clear obstacle on the way of exploring interactively big geospatial data, especially for queries that require aggregation of millions of georeferenced data over thousands of polygons representing the geographic study areas.

Existing approaches rely on applying the stock version of filter-and-refinement approach, thus confronting the challenge of having to join data using PIP test on complex polygons representations that stem from the fact that each polygon could be represented by large number of vertices.

To address this problem, in this paper, we propose and show the design and prototyping of a novel system that we term ApproxGeoAgg (for Approximate Geospatial Aggregation, hereafter AGA for short), which is a spatial query processing framework in support of geospatial aggregation queries in standard-compliant geospatial data management systems. ApproxGeoAgg works efficiently on aggregating high volume geospatial data over complex polygons representing geographic study areas. Most importantly, our system hosts a simplifier model that operates in the front-stage to reduce the number of vertices on the boundaries representing the polygons that cover the geographic study area. We show how this has a direct impact on reducing end-end running times when it comes to running highly-costly spatial aggregation queries over millions of georeferenced data tuples, while keeping accuracy levels in check at statistically plausible figures.

We have performed extensive performance testing using real-world big georeferenced datasets projected with complex polygons having high number of border vertices. Our evaluation results confirm that our system significantly outperforms a plain baseline (which is common in the literature) by statistically plausible figures. We first review the related literature. In what follows we formally define the problem of spatial aggregation over polygons. Thereafter, we show the design and implementation insights of our system. We follow that with a discussion of the results we obtained. We conclude the paper with concluding remarks recapitulating the system functionalities and performance, and we provide a few recommendations for future research.

II. Related Literature

Geospatial aggregations on complex spatial polygons representing study areas (e.g., cities, countries etc.,) is a well-defined problem in the related literature [8]. This type of operation encapsulates spatial join intrinsically as an indispensable part of its operation. Those approaches rely on a two-step approach known as filter-and-refinement for spatial join in the aggregation pipeline [4]. Filtering is traditionally performed by covering the study area by rectangles that are known as Minimum Bounding Rectangles (MBRs) or Minimum Bounding Boxes (MBBs) that completely cover the area, thereafter, finding the MBRs of every tuple in the georeferenced dataset, and projecting the two resulting intermediate sets based on what is known as MBR-join, which resembles equijoin processing in conventional database management systems. The refinement stage is typically performed by a costly geometrical operation known as Point-In-Polygon (PIP) using algorithms such as ray-casting algorithm. MBRs covering each polygon are typically generated using approaches that are based on Z-order curves such as geohash encoding.

Recently, authors in [9] have designed an adapted method that is based on retrofitting filter-refinement approach in the following way. For each polygon, instead of using the ray-casting algorithm to determine whether each tuple is contained within a polygon or not, they cover the inner area of each polygon with a minimum rectilinear polygon (which is then subdivided into several connected rectangles), which acts as a quick-and-approximate sieve that filters out irrelevant data that do not have their MBRs contained within the rectilinear polygon. They apply the costly ray-casting algorithm to the remaining boundary points. They have engineered their system atop SpatialHadoop [10], which is a scalable geospatial data management framework, thus they utilize the spatial support provided over-the-counter by the codebase of SpatialHadoop,

such as spatial indexing. This way they guarantee that their method operates in a distributed fashion and gains significant speedups as the costly geospatial PIP operations are executed in parallel.
In the same vein, authors in [11] have designed a system they term as MPI-Vector-IO system which incorporates spatial data processing components that utilize the filter-and-refine approach on spatial data stored on parallel distributed systems. Perhaps most importantly is the Apache Sedona (previously GeoSpark [6]) which utilizes filter-and-refinement approach for spatial queries including aggregations and operates in a distributed fashion where data is spatially-partitioned and indexed with two-level indexing (local and global).
Within the same consortium, authors of [12] have designed GeoFlink to support the operation of range and KNN geospatial queries atop Flink [13, 14] on parallel distributed computing environments to speed up the processing.
Filter-and-refine approach is also common in spatial data management systems that are based on NoSQL distributed storage frameworks such as MongoDB. For example, in a previous work that appears in [2], we have designed a spatial data storage management and processing system that operates over MongoDB, where we utilize filter-refinement approach for scalable spatial-join queries over huge amounts of disk-resident georeferenced mobility data.
Also, adapted, and retrofitted versions of filter-and-refinement approach for scalable geospatial aggregation queries are employed for joining heterogeneous big multi-domain georeferenced datasets that are varying in spatial and temporal resolutions. For example, in our recent works that appear in [15, 16], we have developed methods that are based on a retrofitted, repurposed and adapted version of filter-and-refinement approach for efficiently integrating geospatial mobility data with contextual enrichment data such as air pollution and meteorological data, aiming at supporting efficient running of geospatial aggregate queries on the generated unified views.
Nonetheless, the image that emerges from the recent literature is that systems rely on the stock versions of filter-and-refinement approach for geospatial aggregate-type queries. However, given the fact that real-life polygons could have complex representations of borders and boundary lines with possibly thousands of vertices, requiring a great deal of running time just to deal with geospatial data that reside on borders of those polygons in real geometries. However, according to Tobler's First law of geography [17], everything is related to everything, however nearby things are more related than those that reside far apart.
Our approach presented in this paper differs from those in the recent literature on the basis that it performs spatial joins on reduced versions of the complex polygons, thus speeding up subsequent operations that incorporate aggregation queries over huge amounts of georeferenced data streams, increasing therefore the throughput while keeping accuracy in check.

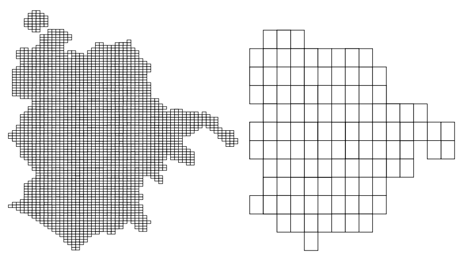
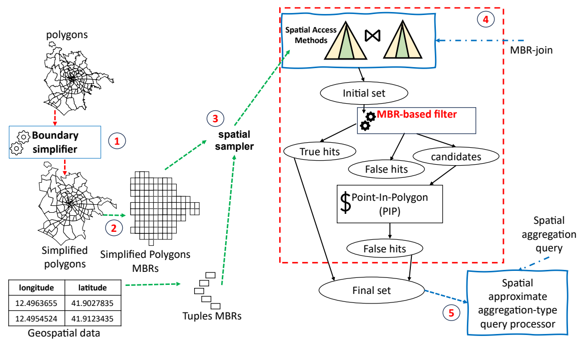

a. Geohash 6 b. geohash 5

Fig. 3 Geohash cover, city of Rome, Italy

III. Prelimenaries and Foundations

A. Geohash Encoding

To work with voluminous locational data, spatial data management systems typically encompass two main steps. First, a representation is overlayed on the study area, for example using a square grid. Thereafter, a spatial data access structure is imposed on the representation to speed up spatial operations. Perhaps most importantly, an ordering data structure such as Z-order curves is imposed on the grid representation, which is a structure that projects the two-dimensional cells in equivalent single-dimension representation. A variant of z-order curves is the geohash [1] encoding. It is basically a string encoding such that geographically-nearby objects have the same geohash value. It can be heuristically visualized as adjacent non-overlapping squares (or rectangles) completely covering the study area such as those shown in Figure 3. It has a configurable parameter that is normally known as geohash precision, where the higher the value the smaller the rectangle size (each rectangle has a distinct geohash value). Geohash encoding is a widely-adopted as a quick-and-approximate filter for proximity searches [18]. Figure 3.a shows geohash covering generated for the city of Rome in Italy at a precision that equals 6, whereas figure 3.b shows the geohash covering for the same city with precision 5.



longitude	latitude
12.4963655	41.9027835
12.4954524	41.9123435

Fig. 4 ApproxGeoAgg architecture

[1] http://geohash.org/

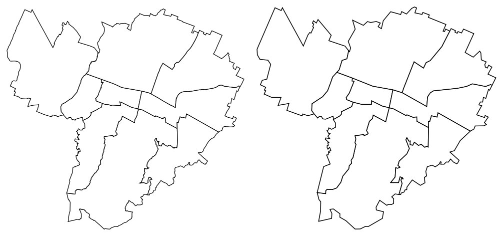


Fig. 5 Boundary simplifier function applied to polygons representing Bologna city, Italy.

B. Problem Formulation

We first formally define the problem of group-by based type of aggregate queries in spatial data management.

Group-by based aggregate queries. Given a geospatial dataset on the form $S = \{(x_1, y_1, v_1), (x_2, y_2, v_2), \dots \dots, (x_n, y_n, v_n)\}$, where x/y are longitude/latitude coordinates, n is the number of the tuples in the dataset. In addition, another georeferenced data representing the study area (e.g., city, country) as shapefile or a GeoJson file, where each administrative regions (e.g., neighborhoods, districts, etc.,) are designated as polygons and are represented by coordinates $area = \{P_1, P_2, \dots \dots, P_m\}$, where m is the number of polygons covering the area. Then, Group-by based kind of spatial aggregation between points and polygons results in a set $R = \{GS_1, GS_2, \dots GS_j \dots, GS_m\}$, where GS_j is the geo-statistic in polygon j. This geo-statistic could be 'count', 'mean', etc. This type of queries is extremely expensive when applied to big georeferenced datasets with millions of tuples as it requires applying spatial join as part of the pipeline. In the next section, we show an efficient system that we have designed specifically for efficient processing of such kinds of queries.

IV. APPROXGEOAGGR: EFFICIENT AGGREGATION QUERIES OF BIG GEOREFERENCED DATA OVER COMPLEX POLYGONS

In this section, we show the design and realization of our system ApproxGeoAgg (short for Approximate Geospatial Aggregation) for the efficient implementation of costly aggregation queries of big georeferenced data over complex polygons representing geographical study areas (such as cities).

A. ApproxGeoAgg Architecture

The schematic context diagram of Figure 4 shows a high-level overview of our system ApproxGeoAgg (for Approximate Geospatial Aggregation system). An integral component of our system is the *boundary simplifier*, which operates as it follows. Our simplifier component is an adapted version of the Douglas-Peucker (DP) algorithm and based on the idea of percentage of boundary points to keep ('percentage of BPK' hereafter for short). Figure 5 shows example application of the boundary simplifier function to the polygons representing the city of Bologna in Italy. We first profile the georeferenced big dataset to capture a figure showing the running time for a bunch of points based on permutations of 'percentage of BPK', based on that we decide the percentage and we pass it as a parameter to the plain DP algorithm to simplify the polygons. We then pass the simplified version of polygons to a *geohash cover generator* component which is responsible for generating geohash cover of the simplified version. We pass this list to the plain filter stage of *filter-and-refine* and project the georeferenced data with this version as shown in Figure 5. This results in two sets, points that have their MBRs fall completely within one of the MBRs of the cover, and points that are candidates which have their MBRs intersect with one of the MBRs of the polygon. For the candidates, we pass them to the refinement component which employs the stock version of the refiner to check, using the PIP, the points that geometrically fall within one of the polygons, those are retained, others are discarded. Based on the query running time budget, we decide the sampling fraction and sample the data accordingly. Then, we pass this to the *aggregator* component that is, based on the type of the aggregation query (and hence the group-by) , groups the objects and computes an approximate result with rigorous error bounds and serves them to the user.

B. Supported Queries and Qunatifying Uncertainty

Our system currently supports two kinds of aggregation geospatial queries. Specifically, Top-N, and geo-statistics, both requiring geospatial group-by operations.

For error estimations, specifically accuracy-based, we employ Pearson Correlation Coefficient (PCC) for Top-N, while we rely on Mean Absolute Percentage Error (MAPE) for geo-statistic group-by queries (specifically 'mean' queries). MAPE is a measure of prediction accuracy, and we calculate it as in (1).

$$MAPE = {}^{100\%}/_{n} \sum_{i=1}^{n} \left| {}^{AC_i - P_i}/_{AC_i} \right|. \quad (1)$$

Where AC_i is the actual average value calculated using the baseline method, whereas P_i is the average value calculated by our ApproxGeoAgg sampling method. n is the number repetitions of the test, thus accounting for stochastic behavior of the computation. PCC measures the linear correlation between two data distributions. It is computed as a ratio between covariance of two variables and product of their standard deviations (SDs). It is a normalized measurement of covariance. The output is a value between −1 and 1. It is computed as in (2).

$$\rho_{\text{AGA,baseline}} = \text{cov}(\text{AGA, baseline}) \,/\, (\sigma_{\text{AGA}} \,.\, \sigma_{\text{baseline}}). \quad (2)$$

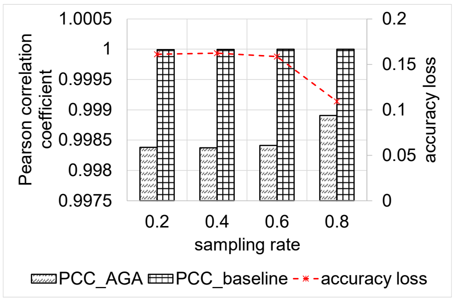


Fig. 6 Pearson Correlation Coefficient (PCC) ApproxGeoAgg (AGA) against plain baseline, geohash size 6, percentage of BPK 5%, NYC data

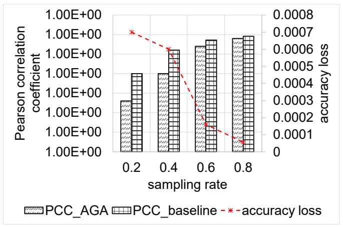


Fig. 7 Pearson Correlation Coefficient (PCC) ApproxGeoAgg (AGA) against plain baseline, geohash size 6, percentage of BPK 90%, NYC data

We calculate the end-to-end running time as the summation of spatial join time (the adapted filter-and-refine) and the group-by time. It is worth mentioning that the filter-refine time is on the order of seconds, whereas the group-by time is measured in milliseconds.

V. Perfomrance Evaluation

In this section, we summarize test settings, the datasets, in addition to the baseline methods and evaluation metrics.

A. Experimental setup

Datasets. To test the performance of the system functionalities, we depend on a mobility georeferenced dataset. Specifically, New York City taxicab trip datasets (https://www1.nyc.gov/site/tlc/about/tlc-trip-recorddata.page, accessed on: 10 January 2023), consisting of around 1,400,000 tuples, representing data taxi rides for the first month of 2016. We selected the green taxi trip records, which included fields such as GPS locations and trips distances.

Deployment. We deploy our experiments on a virtual machine on Microsoft Azure with the following resources. 4 E8 v3 machines with 8 cores processors and 64 GB of RAM. We implemented the system in Python utilizing geo libraries such as Geopandas.

Configurations. We depend on varying sampling rate, geohash precision and number of BPK (boundary points to keep). We specifically vary BPK between stringent 5% and permissive 90%. We also vary sampling rate between 20% and 80%, with 20% step size.

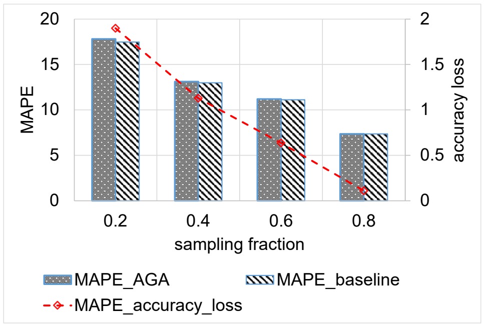


Fig. 8 MAPE for ApproxGeoAgg (AGA) against plain baseline, geohash size 6, percentage of BPK 5%, NYC taxi pickup data

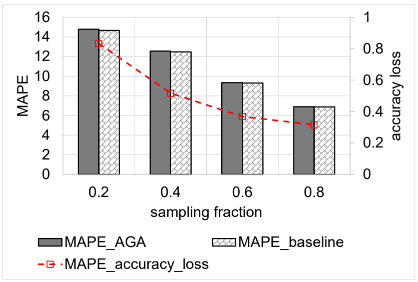


Fig. 9 MAPE for ApproxGeoAgg (AGA) against plain baseline, geohash size 6, percentage of BPK 90%, NYC taxi pickup data

B. Performance Testing Results Discussion

In this section we show the results we obtained by testing the performance of our system ApproxGeoAgg using various settings and varying several configuration parameters.

We specifically depend on varying the geohash precision, between 5 and 6, and the sampling rate for both stratified-like and simple random sampling.

Figure 6 shows the accuracy results for NYC data, AGA against baseline using the PCC for measuring the performance on Top-N queries, with geohash precision 6 and 'percentage of BPK' 5%, we obtain roughly a loss in accuracy that equals to 0.0147 %, on average, ranging from roughly 0.16 at sampling fraction 0.20 to 0.01 at sampling fraction 0.80. Accuracy improves as we increase the 'percentage of BPK' to a generous 90%, where we obtain roughly 0.00038% loss in accuracy, extremely tiny and statistically insignificant, ranging from 0.0007% to 0.000056% as shown in Figure 7. This shows that the performance of the system improves as we increase the sampling fraction. Also, the loss in accuracy decreases as we increase the sampling fraction.

For geo-statistics aggregate queries. We vary the geohash precision, the sampling fraction and the 'percentage of BPK'. For a geohash precision 6 and 'percentage of BPK' that are equals to 5%, we obtain MAPE values that are shown in Figure 8 for AGA against the plain baseline. The loss in accuracy equals roughly to 0.94%, on average, ranging from circa 1.8% at sampling fraction 0.2 to 0.1% at a high sampling fraction of 0.8. By increasing the 'percentage of BPK' to permissive 90%, we obtain higher accuracy as the accuracy loss is on par with 0.43%, on average, ranging from 0.83% to 0.034% at sampling fraction 0.8 as shown in Figure 9. The image that emerges from those figures is that we obtain higher accuracy by either increasing the sampling fraction with same 'percentage of BPK' or increasing 'percentage of

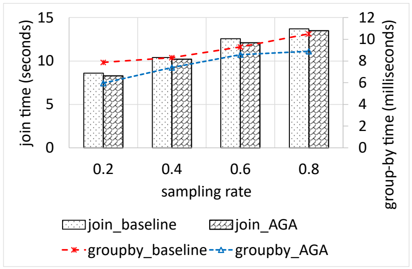


Fig. 10 Spatial join and group-by running times for ApproxGeoAgg (AGA) against plain baseline, geohash size 6, percentage of BPK 90%, NYC taxi pickup data

BPK' themselves. The performance change is linear and stable with various configurations of the parameters as the accuracy loss decreases linearly by changing those parameters, signifying the stability and predictability of the system in real deployment settings. For 'percentage of BPK' that equals 90% on geohash precision 6, we obtain a gain in running time for the aggregation queries that equals to roughly 2.6%, on average as shown in Figure 10. We obtain higher gain for the same geohash precision with aggressive 'percentage of BPK' that equals to 5%, where we obtain a running time gain that equals to 12%, on average.

VI. Conclusions and Future Works

In this paper, we present the design and realization of our new system ApproxGeoAgg that we have devoted for smart city scenarios which require running geospatial aggregate queries over tremendous amounts of georeferenced data streams by projecting them with highly complex polygon data representing geographic study areas of interest. Our system is composed of various main components, including an adapted version of the filter-refinement approach for geospatial join processing, in addition to a front-stage filter that is based on the Douglas-Peucker algorithm for efficiently reducing the number of vertices in the boundaries of the polygons, thus providing a significant speedup and overall reduction in the running times. This is significant in smart city dynamic scenarios that require the interactive processing and analytics of big georeferenced data streams.It is worth investigating and testing other methods for simplifying the boundaries of polygons and reducing the number of vertices of the boundary, and we consider this as a potential future research direction. Our system has a potential scalability beyond the datasets tested in this paper. Current limitations include the absence of a sensing model and frontline controller that can quantify the tradeoff between accuracy and latency and serve the quantification result as a parameter value to the simplifier and sampler modules in the front-stage, similar to a recent work in [19]. Future works include testing with various parameters configurations of the line simplification algorithms, applied to bigger amount of data, also in distributed computing environments, in addition to other methods of simplification. We also aim to quantify the trade-offs between running time and accuracy. We also aim at expanding on potential applications in other domains such as low-cost air quality sensors data to showcase the versatility of our system for various smart city dynamic scenarios.

Acknowledgment

This research was supported in part by the OntoTrans project and has received partial funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No 862136.

References

[1] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Locality-Preserving Spatial Partitioning for Geo Big Data Analytics in Main Memory Frameworks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 2020: IEEE, pp. 1-6.

[2] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Efficient QoS-Aware Spatial Join Processing for Scalable NoSQL Storage Frameworks," *IEEE Transactions on Network and Service Management,* 2020.

[3] I. M. Aljawarneh, P. Bellavista, C. R. De Rolt, and L. Foschini, "Dynamic Identification of Participatory Mobile Health Communities," in *Cloud Infrastructures, Services, and IoT Systems for Smart Cities*: Springer, 2017, pp. 208-217.

[4] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger, "Multi-step processing of spatial joins," *Acm Sigmod Record,* vol. 23, no. 2, pp. 197-208, 1994.

[5] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Big Spatial Data Management for the Internet of Things: A Survey," *Journal of Network and Systems Management,* vol. 28, no. 4, pp. 990-1035, 2020.

[6] J. Yu, J. Wu, and M. Sarwat, "Geospark: A cluster computing framework for processing large-scale spatial data," in *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015, pp. 1-4.

[7] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest, "Geomesa: a distributed architecture for spatio-temporal fusion," in *Geospatial informatics, fusion, and motion video analytics V*, 2015, vol. 9473: SPIE, pp. 128-140.

[8] I. M. Al Jawarneh, P. Bellavista, L. Foschini, and R. Montanari, "Spatial-Aware Approximate Big Data Stream Processing," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019: IEEE, pp. 1-6.

[9] W.-T. Chu and H.-P. Tsai, "Rectilinear Range Query Processing on SpatialHadoop Platform," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021: IEEE, pp. 1-6.

[10] A. Eldawy and M. F. Mokbel, "Spatialhadoop: A mapreduce framework for spatial data," in *2015 IEEE 31st international conference on Data Engineering*, 2015: IEEE, pp. 1352-1363.

[11] S. Puri, A. Paudel, and S. K. Prasad, "MPI-Vector-IO: Parallel I/O and partitioning for geospatial vector data," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp.

[12] S. A. Shaikh, K. Mariam, H. Kitagawa, and K.-S. Kim, "GeoFlink: A distributed and scalable framework for the real-time processing of spatial streams," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3149-3156.

[13] V. Kalavri and F. Hueske, "Stream Processing with Apache Flink," ed: O'Reilly Media, Inc, 2019.

[14] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* vol. 36, no. 4, 2015.

[15] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Efficiently Integrating Mobility and Environment Data for Climate Change Analytics," in *2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2021: IEEE, pp. 1-5.

[16] I. M. Al Jawarneh, L. Foschini, and P. Bellavista, "Efficient Integration of Heterogeneous Mobility-Pollution Big Data for Joint Analytics at Scale with QoS Guarantees," *Future Internet,* vol. 15, no. 8, p. 263, 2023.

[17] W. R. Tobler, "A computer movie simulating urban growth in the Detroit region," *Economic geography,* vol. 46, no. sup1, pp. 234-240, 1970.

[18] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Spatially Representative Online Big Data Sampling for Smart Cities," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020: IEEE, pp. 1-6.

[19] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "QoS-Aware Approximate Query Processing for Smart Cities Spatial Data Streams," *Sensors,* vol. 21, no. 12, 2021, doi: 10.3390/s21124160.