Line Simplification for Efficient Approximate Join Queries On Big Geospatial Data

Fatima Ahmed Alhammadi , Haya Almadhloum Alsuwaidi, Shooq Abdelrahman Alzarooni, Isam Mashhour Al Jawarneh

Department of Computer Science, University of Sharjah, P.O.Box. 27272 Sharjah, United Arab Emirates
U23102315@sharjah.ac.ae, U23102277@sharjah.ac.ae, U23102399@sharjah.ac.ae, ijawarneh@sharjah.ac.ae

***Abstract*— Line simplification algorithms are often used to render high-resolution geographic features at appropriate resolutions when applied to polygons. They are generalization techniques in which selective vertices are removed from a line feature to eliminate details whilst preserving the line's basic shape. In this paper, two different line simplification algorithms (Douglas-Peucker and Visvalingam-Whyatt) are used in conjunction with spatial join of geo-referenced mobility and air quality data, to reduce the size of the polygon files. A filter-and-refinement dimensionality reduction-based approach is then used to join the data. This framework allows for an optimized spatial join on an integrated schema through a state-of-the-art filter-and-refine based approach. The reduced files can then be used in geospatial related data science tasks such as DBSCAN, clustering, and regression at lower computational costs. Our experimental results show that incorporating a reduction approach such as line simplification before performing the spatial join can significantly reduce the computational cost and improve the performance with the number of vertices reduced by 94% after simplification and accuracy, MAPE is minimized with a low score of 0.048 and 0.049 for DP-Map shaper and VW-Map shaper respectively.**

Keywords—AQP, Douglas Peucker algorithm, line simplification, Visvalingam-Whyatt algorithm, spatial join

I. INTRODUCTION

The growing presence of the Internet of Things (IoT) in all aspects of our life has brought about an exponential increase in the data generated daily, a large majority of which is spatial (a.k.a. geo-referenced or spatially-tagged). Certainly, the processing of these large amounts of data is incredibly taxing, more so when data joins are carried out as one of the most fundamental skills in data analysis that is used to gather useful insights. In response, cultivating and efficiently analyzing this spatial data in a computational efficient manner has been a growing area of research. Geospatial join is a computationally costly workload that is more frequently being used and applied in dynamic smart city application scenarios, specifically those that require fusing data from multiple georeferenced heterogeneous data streams. For example, authors in [1] have described a very interesting scenario where there is a need to merge hyperlocal air quality and mobility data of a city in an attempt to unleash possible autocorrelations between vehicles density and the air quality on a street-by-street level. Both mobility and air quality data are georeferenced, meaning that they are tagged with locational coordinates. This scenario requires an advanced geospatial join processing technique that goes beyond the traditional woks of geospatial join methods. The problem is mostly attributed to the fact that geospatial point data need to be joined with shapefile polygons data representing the administrative city districts. Those shapefiles are normally huge in size, which negatively affects the overall join performance, thus reducing their size is becoming indispensable. Line simplification is one of the methods used to reduce the geometric points in a geographical space (specially applied to vector line and polygon shapes), to reduce their storage size, whilst maintaining their general topology of the shape. Two of the most prominent line simplification algorithms are Douglas-Peucker [2] and Visvalingam-Whyatt [3]. This paper aims to explore the results of the line simplification algorithms when used alongside spatial joins, across two different implementation techniques (Shapely library in Geo Python and Mapshaper.org [4]). The purpose of this combination is to minimize computational costs whilst preserving the important information, in a way that ensures, to a significant extent that it does not negatively impact any analysis thereafter. The rest of the paper is organized as follows: Section II covers the related state-of-the-art works in the field; Section III defines the methodology and overview of the models; Section IV details the results and their discussion; and Section V concludes with the findings of this paper and future work perspectives.

II. RELATED LITERATURE

Given the emergence of mixed workloads in smart cities inspired by the adoption of Internet of Things (IoT), there is a growing importance in finding the geographical regions to which data streams belong, leading to geospatial data. Its development revealed that processing and running geospatial data is computationally intensive. In response, line simplification algorithms were developed to reduce the size of the data whilst preserving the overall accuracy of the data – a compromise that is particularly important in big data handling where computer resources are expensive [5, 6].

Recently, authors in [5] proposed a polygon simplification method, named GeoRAP, built on geospatial approximate processing. Their framework is based on the Ramer-Douglas-Peucker line simplification algorithm to reduce the area coverage as well as a version of stratified spatial sampling to minimize the number of strata, and subsequently data points in each strata. This approach increases the throughput whilst minimizing response time, preserving the important information

979-8-3503-5475-1/24/$31.00 ©2024 IEEE

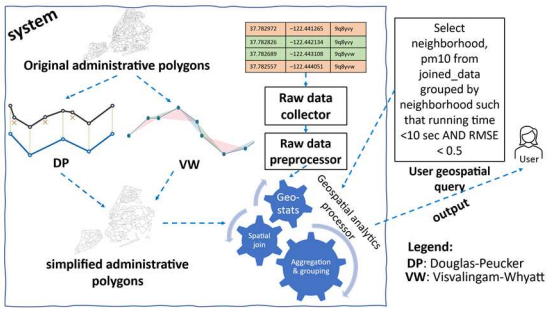

Fig. 1 methodology overview

which makes it more efficient for subsequent data science tasks such as complex geo-statistics and aggregation queries [7].

In another approach, authors in [8] introduced an adaptive spatial-aware approximate query processing solution (termed as SpatialSSJP) that focuses on stream-static geospatial joins and support Quality-of-Service goals and online aggregations. The paper proposes an efficient online sampling design to select a balanced and representative geospatial data from the stream using the Spark Structured Streaming framework [9], for a stream-static geospatial join operator downstream, even on large join workloads, with an improved performance that is capped at approximately 10-50% as compared with the baseline Apache Spark methods.

Similarly, researchers in [10] tackle data overloading through a spatial approximation query processing method, named ApproxGeoViz, where efficient region-based geo-maps from fast arriving big georeferenced data streams can be generated. The model was tested on real smart city data and evaluated on time-based and accuracy-based QoS constraints. Correspondingly, [5] presented a similar methodology, ApproxGeoAgg, designed for optimizing aggregation queries in spatial data analytics requiring grouping geospatial objects into predefined clusters of administrative polygons study areas. This study also involved cropping polygons to reduce their size and reduce the overall geospatial join cost in comparison to the full version.

In terms of employing spatial join for fusing heterogeneous georeferenced data streams, several works can be traced in the literature. Authors in [1] applied an filter-and-refinement approach for integrating geotagged air quality and mobility data with different spatial and temporal resolutions for smart city and urban analytics. In the same vein, authors in [11] have applied a method that is based on filter-and-refinement spatial join approach for joining meteorological and mobility data at scale with time-based and accuracy-based QoS guarantees.

As for the line generalization applied to spatial approximate data analytics, In the same vein, [12] have designed a modern spatial big data analytics framework which incorporates tools for trajectory data compression using DP line simplification algorithm intrinsically within its architectural design. Similarly authors [13] designed an Enhanced Douglas–Peucker (EDP) algorithm that employs a constellation of enhanced spatial–temporal constraints (ESTC) for simplifying and compressing trajectory data streams. Within the same consortium, [14] applied a modern DP algorithm based approach as a frontstage quick-and-dirty sieve to minimize the numbers of trajectory data points fed for clustering tasks downstream in the main system.

III. Methodology

In this section, as shown in figure 1, the methodology employed in this study is explained, including steps taken for data preprocessing, and the different algorithms and tools utilized to perform the line simplification.

An integral part of data science involves the combination and analysis of data from different sources. Spatial data, i.e. data that references a geographical location, is dense by nature, having information of both the topic such as urban information, or air quality readings, as well as the location. As the name suggests, spatial join is the combination of two sets of spatial data based on their geographic relationship, naturally this makes the data load for analysis more complex and computationally heavy. Despite this fact, this operation remains necessary as it improves the visualization of the data and enables deeper insights and more detailed spatial analysis.

To improve the computational efficiency of processing spatial data, line generalization algorithms exist as a set of techniques focused on reducing data points without compromising vital information depending on their relevance in a specific model, to determine the best method of representing the data. One of the subfields of generalization is line simplification, which decreases the data volume and complexity by reducing the number of vertices in the vector representation of geospatial data. The effect this reduction has on a systems complexity is particularly evident in large datasets.

This paper applies geospatial join on large hyperlocal air quality data to compare the effectiveness of two common line simplification algorithms for performing faster geospatial joins, those are Douglas-Peucker (DP hereafter for short) and Visvalingam-Whyatt (VW hereafter for short) algorithms, which are discussed in the next two subsections.

A. Visvalingam-Whyatt Algorithm

Visvalingam-Whyatt (VW), introduced in 1993 [3], works by removing the least significant points in a given line, and treating the remainder of the line as a new one. It does that by considering the triangular features and recursively eliminating the smallest triangles as they are assumed to have the least amount of contribution. By removing the triangles with the smallest area, the important geometric characteristics are preserved. The VW algorithm is relatively easy to compute, and its straightforward framework makes it an efficient algorithm. For visualization purposes, this method is often preferred as more significant features on the map tend to be preserved, forming large triangles.

B. Douglas-Peucker Algorithm

Another commonly used line simplification technique is the Douglas-Peucker (DP) algorithm [2]. Preceding the VW algorithm, DP was introduced in 1973 and works by eliminating points on a polygonal chain whilst preserving the original shape of the polygon. As a result, this algorithm is particularly useful to applications where simplification is needed without significantly altering the appearance of the polygon and

maintaining its visual integrity whilst reducing the computational power required. The algorithm takes a tolerance parameter that adjusts the degree of simplification, making it easy to scale the outcomes to varying levels of detail. Several vector line generalization algorithms are employed in the literature, which are categorized into five groups in [15]. For example, Reumann–Witkam routine [16], the Douglas–Peucker (DP) algorithm [2], and an algorithm by Visvalingam and Whyatt (VW) [3]. Global algorithms (e.g., DP) considers the entire line while reducing the line, as opposed to other categories which work with segments of the entire line. DP and VW are preferred as they are more accurate for preserving the original line's shape as corroborated in [15]. Having said that, we employ the DP and VW algorithms as integral parts of our system system in this paper.

C. Data Importing and Preprocessing

For testing the different line simplification algorithms, two publicly available georeferenced mobility datasets were chosen. The first dataset contains information about New York City Polygons, including the geographical boundaries for the neighborhoods across the city, where each neighborhood is represented by a polygon, a form of vector geospatial representation (in the form of GeoJSON file extension). The dataset has 310 enteries each consisting additional information such as the neighborhood name, borough, borough code and a unique identifier that links to more detailed resources about the neighborhood. The second dataset used is a unique geotagged air quality dataset collected using low-cost air-quality sensors, consisting of 170K records, each entry includes a timestamp, geographic coordinates, temperature, humidity and particulate matter levels.

In this phase, a series of pre-processing steps were performed to make sure that the geographical data was cleaned, consistent, and suitable for further analysis and computations. The data that was gathered in CVS and GeoJSON formats including multiple attributes such as geographical boundaries (polygons and point geometries, longitude, and latitude) and environmental data (such as temperature, humidity, and pm25).

The first step was to clean the data by removing erroneous coordinates (coordinates where the longitude and latitude are equal to (0,0) respectively), and missing data were dealt with by filling them. To standardize the analysis, all geographical data was converted into a uniform coordinate reference system (CRS), namely, EPSG 4326. This uniform CRS was used to ensure that the accuracy of the geographical measures such as distance and area are maintained and accurate across the dataset.

To better understand the data, feature extraction and visualization tasks were performed, out of necessity for our line simplification analysis. Feature extraction includes finding the area in square meters and computing the number of vertices of the data. Evaluating these features played a role in assessing the efficiency of our data simplification algorithms thereafter in this study. In Figure 2, After pre-processing the data, spatial join was performed by joining the air quality data with the neighborhood data and a heatmap was generated as depicted to visualize the varying levels of air quality. The bright areas in the map indicate areas with better air quality, given that they have a lower concentration of particulate matters (pm10 and pm2.5)

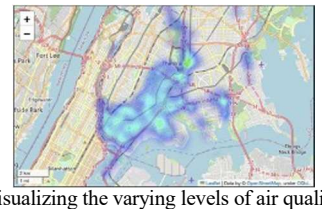

Fig. 2 heatmap visualizing the varying levels of air quality, NYC USA.

pollutants. On the other hand, the darker areas indicate higher concentrations of the pollutants, these areas may be closer to major highways or areas with high vehicle emissions.

D. Simplification Algorithms

In this section, the different algorithms and frameworks used are outlined. Two algorithms (DP and VW) were evaluated on more than one framework.

The DP algorithm was implemented via two different frameworks: the first algorithm is implemented using the Shapley library provided by GeoPython, for precise geometric manipulation with a tolerance of 0.001, and the second algorithm is implemented using Mapshaper.org, a web-based simplifier using a tolerance of 0.1%, for more efficient results and less overlapping between the neighborhoods, allowing for a comprehensive comparison of the data in terms of data reduction and geometry preservation.

Douglas-Peucker with Shapely (hereafter DP-Shapely for short). This approach implemented the DP algorithm which uses a distance measure to test each single point by using the GeoPython Shapely library to simplify the geographical data. As mentioned in section II, the DP algorithm mainly reduces the number of vertices in each polygon while preserving the topology. The implementation of the function which iterates through each feature of the GeoJSON data, takes the polygon geometry as an input and applies the 'simplify()' method provided by the Shapely library, which takes a specified tolerance level as input; this determines how much of the geometry will be simplified. In general, they are indirectly proportional such that when the tolerance value increases, the number of vertices decrease. In order to maintain the original shape and minimize overlapping between the vertices, the tolerance was set to 0.001. As a result, a new dataset in the format of GeoJSON was created where each polygon contains a lower number of vertices.

Douglas-Peucker with Mapshaper.org [4] (hereafter DP-Mapshaper for short): A second tool was introduced to apply the DP algorithm via Mapshaper, which was designed for efficient map simplification. Mapshaper is a generalization web service developed to help mapmakers simplify and smooth their vector line work using a suite of visual-editing tools. This approach is unique in that Mapshaper can handle large data more efficiently than Shapley while minimizing overlapping as much as possible, making it appropriate for extensive geographical data. The original GeoJSON file is fed into the Mapshaper and executes the 'simplify()' command with a specified tolerance, which was set to 0.1% corresponding to a tolerance of 0.001.

a. DP-Shapely b. DP-Mapshaper

Fig. 3 Polygon diagram of NYC neighborhoods applying DP algorithm

The command regulates the complexity of the polygon geometry in a similar manner to Shapely while simultaneously optimizing for processing and supporting various output formats, which makes it easier for subsequent analysis. The last step is exporting the simplified data back into GeoJSON format for further processing. To compare the results of the DP simplifier, the VW simplifier using the Mapshaper tool was examined. Similar to the previous method, this algorithm was implemented to analyze and provide insights about the data.

Visvalingam-Whyatt with Mapshaper.org (VW-Mapshaper hereafter for short). This simplification technique was added for comparison and analysis purposes. The VW method applies an area measurement by using the Mapshaper web service. This method is different from DP as VW works by removing the points that result in the smallest area change first, making it more preserved in terms of data visualization. It works similarly to the DP algorithm with the Mapshaper.org tool but by replacing the 'simplify()' command with 'method=Visvalingam'.

E. Statistical Analysis

To evaluate the performance of the data simplification algorithm against the original data, several statistical metrics were examined after applying the data processing methods including aggregation, normalization, and merging the original and simplified data based on their neighborhoods. The metrics examined were: Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Spearman Correlation, and Jensen-Shannon divergence (hereafter JSD for short).

a) Root Mean Square Error (RMSE): This measure is used to calculate the average magnitude of the error between the original and simplified data. It shows clearly how much the simplification deviates from the original in terms of spatial accuracy. The formula is represented in (1) Where O_i and P_i are the original values and simplified values, respectively, and n is the number of observations.

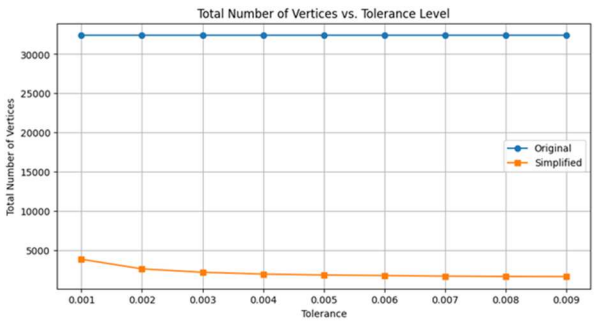

Fig. 4 Number of Vertices vs. Tolerance, NYC polygons

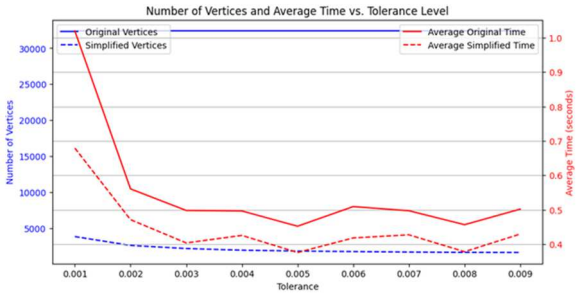

Fig. 5 Number of Vertices vs. Tolerance vs. Average Time

$$RMSE = \sqrt{(\frac{1}{n}) \sum_{i=1}^{n} (O_i - P_i)^2} \quad (1)$$

b) Mean Absolute Percentage Error (MAPE): MAPE is used to express the error as a percentage of the original data, which gives us an insight about the errors in terms of relative size. The formula is shown in (2).

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{O_i - P_i}{O_i} \right| \quad (2)$$

c) Spearman Correlation: The spearman correlation is a statistical test to measure the strength of a monotonic relationship. A monotonic function is a function that either never increases or decreases as its independent variable increases between paired data. In this case, the comparison is computed between the original neighborhood data and the simplified neighborhood data. It is denoted by r_s and follows the constraint below, the closer r_s is to 1 the stronger the monotonic relationship [10], such that $-1 \leq r_s \leq 1$.

d) Jensen-Shannon (JS) Divergence: JS is a statistical method used to measure the similarity between two probability distributions. It is defined in (3) where *H(.)* represents the Shannon entropy of a distribution. It is important that this metric is minimised, since a lower score indicates that the important information between the two sets of data, that is, original and simplified is preserved, whilst reducing the data size.

$$D_{JS}(P, Q) = H(M) - \frac{1}{2}(H(P) + H(Q)) \quad (3)$$

IV. Results and Discussion

For comparison purposes, the polygons representing New York City neighborhoods in the USA were generated, and the same diagram was generated for all implementations of the simplification algorithms, to analyze their effectiveness.

Applying the DP simplifier with shapely and a tolerance of 0.001 produced the diagram in Figure 3.a, where some overlapping in the boundaries of the neighborhoods is evident.

To resolve the overlapping issue encountered while using the DP simplifier with GeoPython Shapely, the Mapshaper framework with a tolerance of 0.1% was explored. This is due to Mapshaper's ability to preserve the topology and boundaries whilst minimizing overlapping between the neighborhoods. As

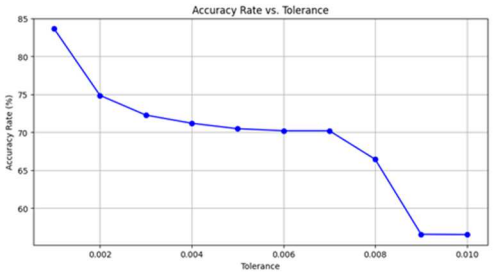

Fig. 6 Spatial Join Accuracy Rate vs. Tolerance

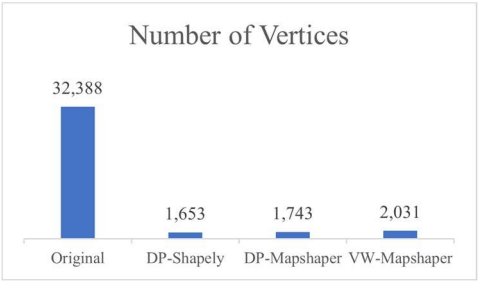

Fig. 7 Number of Vertices across the algorithms

a result, the model was able to ensure less overlapping in the simplified data and the retention of boundaries between adjacent polygons. The resulting diagram can be seen in Figure 3.b.

Lastly, the VW simplifier was applied using Mapshaper, which shows more details with 0.1% tolerance whilst also preserving the topology and minimizing overlapping. This is due to the retention of boundaries between adjacent polygons feature, which allows the map to appear to be more detailed.This tolerance level determines how much the simplified lines can deviate from the original lines. The overlapping between the boundaries of the neighborhoods is a result of the tolerance applied. For a deeper understanding of the results, the number of vertices per neighborhood as well as their accuracies were computed.

Figure 4 depicts a graph that compares the number of vertices in the polygons with the data before and after simplification. The blue line represents the original geometries while the orange line represents the simplified geometries. It is evident from the graph that the original polygon of the neighborhoods has number of vertices much higher than the simplified in which for the simplified the number of vertices decrease with increasing tolerance, indicating a loss in detail as the number of vertices decrease. As for the original polygon, it remains constant with no significant changes as the tolerance increases; this indicates that the number of vertices in the original geometry is preserved across all tolerance levels, as expected given that no simplification was applied to the original data.

Additionally, figure 5 shows the computational time required for performing the join operation, specifically a spatial join and the number of vertices produced after line simplification were analyzed in relation to the tolerance levels. The straight lines represent the original data before line simplification and the dotted line represents the data after line simplification. Correspondingly, the red lines represent the average time of the spatial join prior to and following the line simplification while the blue lines represent the vertices prior to and following line simplification using DP-Shapely.

The graph reveals that the tolerance is indirectly proportional to the average time of the spatial join, as well as the number of vertices after simplification; that is, as the tolerance increases, the average time and number of vertices decrease. This result was expected given that increasing the tolerance decreases the number of vertices, thereby reducing the time needed to perform the join operation. It is also evident that the average time taken to perform the spatial join on the original and simplified data exhibit similar behaviors as visible from the parallel trend observed in the figure, although the average time of the original starts very high and has a drastic decrease, this behavior could be attributed to various factors, although it is not possible to determine the specific cause. In addition to the simplified models naturally having a lower number of vertices. Generally, it appears that the number of vertices of the original data remain constant even with increasing tolerance, which is expected given that the original data is not subjected to any changes.

Further, to determine the optimal tolerance score for the spatial data join, figure 6 is used to show the accuracy rate of the join against the tolerance. The graph compares the accuracy of spatial joins between the two sets of geographical data, where the neighborhood and air quality data is evaluated to see how simplification affects the joins.

The tolerance values are ranging from 0.001 and 0.010 with an increment 0.001. The tolerance value determines how the vertices are simplified in the original data. As evident from the graph,the accuracy decreases with increasing tolerance, indicating that there is a tradeoff between reducing the complexity of the data via simplification and maintaining the accuracy. It can also be seen from the graph that there is a significant drop in accuracy at around 0.007 tolerance, this could be an indication that important geographical information necessary for the joins are being removed. This graph can also be used to show the optimal tolerance which seems to be the lowest at approximately 0.001. This tolerance implies that less simplification preserves the most important geographical information, given that with increasing tolerance, less vertices are retained thus, resulting in simpler polygons with less accurate information. Figure 7 compares the number of vertices between the different simplifiers, as evident from Table I, the original – referring to data without any line simplification, has the highest number of vertices. Comparatively, the data with line simplification algorithms have much lower number of vertices. DP-Shapely was able to remove the highest number of vertices while retaining only 1,653 vertices, followed by 1,743 and 2,031 retentions for DP Mapshaper and VW Mapshaper respectively. While these results may indicate that DP-Shapely is the most effective in simplifying the polygons, the graphical representations, Figures 3, 4 and 5 revealed that shapely causes overlap in the data. As such, Mapshaper proves to be the better method for accuracy, as the increase in number of vertices is very minor, but the accuracy is assumed to improve greatly as it avoids overlapping and preserves the geometric characteristics and topology. Table I compares the geographical area of the data and the total number of vertices before and after simplification. The tolerances used are the following: DP-

Shapely simplifier with a tolerance of 0.001, DP-Map shaper simplifier with 1% tolerance, and VW-Map shaper simplifier with 1% tolerance were used.

The readings show that the results were similar between the two algorithms, making them both effective. MAPE is minimized with a low score of 0.048 and 0.049 for DP-Mapshaper and VW-Mapshaper, respectively, suggesting a very accurate prediction and indicating the effectiveness of the model. The algorithms also computed promising Spearman Correlation values of 0.88 and 0.92, respectively, VW performing slightly better, statistically indicating that the original and simplified data are highly correlated and comparable, further reaffirming that the geospatial data is preserved despite reducing the number of vertices by approximately 94%. Further, the Jenson-Shannon Divergence results were relatively low at 0.33 and 0.35, reiterating the similarity between the original and simplified data – that is, the data is sufficiently well-preserved whilst decreasing the computational cost. The correlation and divergence metrics are especially useful for aggregation queries, which are optimized in this analysis. As the results are relatively comparable, the question as to which algorithm to use depends on the application. Generally, DP is better suited for general purpose and adaptive simplification, where the level of detail can be dynamically set, whereas VW is more often used for visual and area-focused applications.

TABLE I. COMPARISON OF RESULTS

Metric	**Algorithm**			
	Original	***DP-Shapely***	***DP-Map shaper***	***VW-Map shaper***
Tolerance		***0.001***	***1%***	***1%***
Area (m^2)	2536228.4	2311793.02	2792454.7	**2260661.6**
No. of Vertices	32,388	**1,653**	1,743	2,031
RMSE	-	**-**	**62.69%**	65.00%
MAPE	-	**-**	**0.04758**	0.04853
Spearman Correlation	-	**-**	0.88370	**0.92074**
JSD	-	**-**	0.33109	0.35564

V. CONCLUSIONS AND FUTURE WORKS

This paper compares the performance of different line simplification algorithms namely Douglas-Peucker and Visvalingam-Whyatt. Two different tools – Shapely and Map shaper were used to compare DP while only Map shaper was used for the VW. It was evident from the results that both algorithms performed in a similar manner via the Map shaper interface, computing promising evaluation metrics. The results of these models prove to be essential as the need for spatial joins increases, seeing as it is computationally expensive in its full form. This approach allows the data size to be significantly reduced whilst preserving the geometric characteristics of the data as well as its visual topology. In doing so, the efficiency of the programs can be greatly improved, cutting down on computational costs. This is particularly important for data science aggregation tasks, including comparisons, DBSCAN (Density Based Spatial Clustering of Applications with Noise), clustering, and regression; allowing analysts to make more informed, strategic decisions.

REFERENCES

[1] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization,* vol. 10, no. 2, pp. 112-122, 1973.

[2] M. Visvalingam and J. D. Whyatt, "Line generalization by repeated elimination of points," in *Landmarks in Mapping*: Routledge, 2017, pp. 144-155.

[3] M. Harrower and M. Bloch, "MapShaper. org: A map generalization web service," *IEEE Computer Graphics and Applications,* vol. 26, no. 4, pp. 22-27, 2006.

[4] I. M. A. Jawarneh, L. Foschini, and P. Bellavista, "Polygon Simplification for the Efficient Approximate Analytics of Georeferenced Big Data," *Sensors,* vol. 23, no. 19, p. 8178, 2023.

[5] I. M. Al Jawarneh, R. Montanari, and A. Corradi, "Cost-Effective Approximate Aggregation Queries on Geospatial Big Data," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023: IEEE, pp. 1313-1318.

[6] A. Hassan and J. Vijayaraghavan, *Geospatial Data Science Quick Start Guide: Effective techniques for performing smarter geospatial analysis using location intelligence*. Packt Publishing Ltd, 2019.

[7] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "SpatialSSJP: QoS-Aware Adaptive Approximate Stream-Static Spatial Join Processor," *IEEE Transactions on Parallel and Distributed Systems,* 2023.

[8] M. Armbrust *et al.*, "Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark," presented at the Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA, 2018.

[9] I. M. Al Jawarneh, L. Foschini, and A. Corradi, "Efficient Generation of Approximate Region-based Geo-maps from Big Geotagged Data," in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2023: IEEE, pp. 93-98.

[10] 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.

[11] 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.

[12] S. Gao *et al.*, "Automatic urban road network extraction from massive GPS trajectories of taxis," in *Handbook of Big Geospatial Data*: Springer, 2021, pp. 261-283.

[13] H. Qian and Y. Lu, "Simplifying GPS Trajectory Data with Enhanced Spatial-Temporal Constraints," *ISPRS International Journal of Geo-Information,* vol. 6, no. 11, p. 329, 2017.

[14] L. Zheng, Q. Feng, W. Liu, and X. Zhao, "Discovering trip hot routes using large scale taxi trajectory data," in *Advanced Data Mining and Applications: 12th International Conference, ADMA 2016, Gold Coast, QLD, Australia, December 12-15, 2016, Proceedings 12*, 2016: Springer, pp. 534-546.

[15] W. Shi and C. Cheung, "Performance evaluation of line simplification algorithms for vector generalization," *The Cartographic Journal,* vol. 43, no. 1, pp. 27-44, 2006.

[16] K. Reumann and A. Witkam, "Optimizing Curve Segmentation in Computer Graphics. International Computing Symposium," ed: Amsterdam, North Holland, 1974.