



Evaluating the effect of compressing algorithms for trajectory similarity and classification problems

Antonios Makris¹ · Camila Leite da Silva² · Vania Bogorny² · Luis Otavio Alvares² · Jose Antonio Macedo³ · Konstantinos Tserpes¹

Received: 13 May 2020 / Revised: 2 February 2021 / Accepted: 18 February 2021 /
Published online: 7 May 2021

© The Author(s) 2021

Abstract

During the last few years the volumes of the data that synthesize trajectories have expanded to unparalleled quantities. This growth is challenging traditional trajectory analysis approaches and solutions are sought in other domains. In this work, we focus on data compression techniques with the intention to minimize the size of trajectory data, while, at the same time, minimizing the impact on the trajectory analysis methods. To this extent, we evaluate five lossy compression algorithms: Douglas-Peucker (DP), Time Ratio (TR), Speed Based (SP), Time Ratio Speed Based (TR.SP) and Speed Based Time Ratio (SP.TR). The comparison is performed using four distinct real world datasets against six different dynamically assigned thresholds. The effectiveness of the compression is evaluated using classification techniques and similarity measures. The results showed that there is a trade-off between the compression rate and the achieved quality. There is no “best algorithm” for every case and the choice of the proper compression algorithm is an application-dependent process.

Keywords Trajectory compression algorithms · Simplifying trajectory algorithms · Trajectory classification · Trajectory similarity · Error metrics

1 Introduction

Massive volumes of moving object trajectories are being generated by a range of devices, services and systems, such as GPS devices, RFID sensors, location-based services, satellites and wireless communication technologies. In essence, these data are spatio-temporal in nature, represented as chronologically ordered points consisting of the position (latitude, longitude) of a moving object, along with the time it was collected (timestamp). The more points a device collects, the more accurate a trajectory representation becomes. However, the increasing amount of data poses new challenges related to storing, transmitting, processing, and analyzing these data [1–3].

✉ Antonios Makris
amakris@hua.gr

Storing the sheer volumes of trajectory data can quickly overwhelm available data storage systems. For instance, one AIS¹-receiving station receives 6 AIS messages, or 1800 bytes, per second. Considering that a ship tracking company may operate thousands of such receivers, we end up with terabytes of data to deal with each month [4]. For example, MarineTraffic's on-line monitoring service collects 60Mb of AIS and weather data every second [5]. Additionally NASA's Earth Observing System produces 1 TB of data per day [6] while Hubble Telescope² generates 140 GB of raw spatial data every week. These facts demonstrate that the transmission of large trajectory data can be proven prohibitively expensive and problematic, as also shown in [7]. Furthermore, these data often contain large amounts of redundant information which in turn can overwhelm human analysis.

A typical approach towards these challenges is to reduce the size of trajectory data by employing compression techniques. The latter aims at substantial reductions in the amount of data while preserving the quality of information, and thus ensuring the accuracy of the trajectory. For example, Fig. 1 illustrates an example of trajectory compression, where the original trajectory is represented by a black line and consists of 9 points, while the compressed representation (dashed line) consists of 4 points. The information loss introduced from compression is, in most of the cases, inevitable. This information loss is known as *error*, and is measured through different metrics that calculate the applied error over a trajectory if a certain point is discarded.

Most works on trajectory compression neglect to evaluate the impact of compression techniques over their intended task. The objective of this work is to move one step forward by investigating and quantifying the relationship as well as the impact of trajectory compression techniques over trajectory classification and similarity analysis. We evaluate the impact of five top-down algorithms, namely Douglas-Peucker (*DP*), Time-Ratio (*TR*), Speed-Based (*SP*), Time Ratio-Speed Based (*TR_SP*) and Speed Based-Time Ratio (*SP_TR*) over similarity and classification. The comparison is based on the compression rate achieved and the execution time across four real-world datasets. Their performance characteristics were compared against six different dynamically defined thresholds. This practically means that in every trajectory of each dataset, a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of this trajectory. Thus, we have eliminated the need of arbitrary user-defined thresholds. The examined algorithms are lossy which means that they present information loss and therefore it is important to measure the quality of the compressed data. In order to evaluate the quality of the compression algorithms, we apply data mining techniques and similarity measures over both the original datasets and the compressed trajectory datasets in order to evaluate their accuracy.

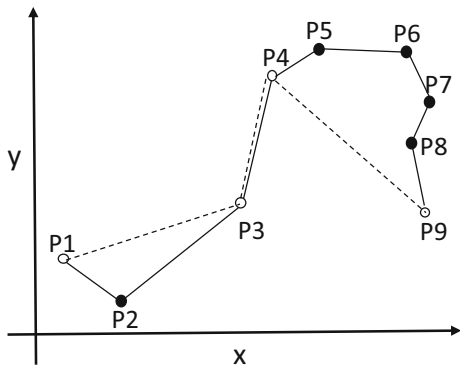
Various metrics such as spatial distance, synchronized Euclidean distance, speed, enclosed area and so on have been previously used in the literature in order to evaluate trajectory compression algorithms. To the best of our knowledge, our work is the first one that evaluates the effectiveness of the compression using classification methods and similarity measures.

The main contribution of this work is that it provides evidence using some proof concept scenarios that, in terms of data points representing a trajectory, only a small portion of carefully selected points are needed in order to conduct the analysis task without significant loss in its performance metrics (e.g. accuracy, precision, recall). We demonstrate that there is a trade-off between the compression rate and the achieved quality and that choosing a

¹AIS, <http://solasv.mcga.gov.uk/regulations/regulation19.htm>

²Hubble Telescope, <http://hubble.stsci.edu/>

Fig. 1 Trajectory compression example



proper compression algorithm is not an easy task: efficiency analysis demonstrated that there is no “best algorithm” and the selection is application-dependent.

The rest of the paper is structured as follows. Related work is discussed in Section 2. Section 3 presents the problem formulation and trajectory compression validation criteria. Section 4 describes the compression algorithms. Section 5 details the compression evaluation and presents the results. Section 6 outlines the conclusion and future work.

2 Related work

In this section we give an overview of trajectory compression methods, trajectory classification techniques and similarity measures.

2.1 Trajectory compression methods

Various compression algorithms exist in literature that attempt to balance the trade-off between the accuracy and storage size while the investigation of compression methods is an active area of research. These algorithms can be logically grouped according to criteria such as computation complexity and error metric employed. In general, data compression algorithms can be classified into two categories, namely lossy and lossless [8]. Lossy compression algorithms attempt to preserve the major characteristics of the original trajectory by removing the less significant data when compared to the original data. On the other hand, lossless compression techniques enable the exact reconstruction of the original data, which means that there is no information loss due to compression. The main advantage of lossy compression techniques is that they can reduce storage size while maintaining an acceptable degree of error [9]. In essence, compression techniques aim at substantial reductions in the amount of data without serious information loss. According to [10, 11] the traditional trajectory compression algorithms can be classified into the following four categories:

1. Top-Down. A trajectory is recursively splitted until a halting condition is met.
2. Bottom-Up. Starting from the finest possible representation of a trajectory, successive points are merged until a halting condition is met.
3. Sliding Window. Starting from one initial point to the end of the subsequent data points, data are compressed in a fixed window size.

4. **Opening Window.** Starting from one initial point to the end of the subsequent data points, a compression takes place on the data points inside a dynamic and flexible data segment size until a halting condition is met.

Compression algorithms can also be classified as online or offline depending on the moment the compression procedure is performed [12]. Specifically, offline algorithms compress a trajectory after it has been fully generated while online algorithms compress the trajectory during the point collection process [13]. As a trajectory is fully generated, offline algorithms are able to obtain more accurate results while on the other hand online algorithms remove redundant data from trajectories as they occur, avoiding the unnecessary transfer of data over the network, improving data storage and reducing the memory space.

Line generalization techniques have been widely used for data compression. Although these algorithms fail to comply with the low complexity requirements of the stream model. A trajectory sample technique must preserve certain properties such as changes in speed and direction with minimum process cost per point. The authors in [2] propose two sampling techniques called STTrace and Threshold-guided. In Threshold-guided sampling technique a new point is appended to the sample when a user defined threshold concerning velocity is exceeded for an incoming location. The algorithm estimates the safe area of the next point using the position, speed and orientation information. STTrace exploits spatiotemporal features that characterize movement and detects changes in speed and orientation. The algorithm uses a bottom-up strategy such that the synchronous Euclidean distance (SED) minimized in each step. Initially, the first recorded positions with each position's SED are inserted in the allocated memory. In the insertion of a new point, if the memory is exhausted, the sample is searched for the item with the lowest SED, which represents the least possible loss of information. If SED of the inserted item is larger than the minimum one, the current processed point is inserted into the sample at the expense of the point with the lowest SED. Finally, the SED attributes of the neighboring points of the removed one are recalculated. The main difference of these techniques is that concerning Threshold-guided sampling, the total amount of items in the sample keeps increasing without eliminating stored points, thus exceeding memory, while STTrace allows deletions from samples in order to store newly inserted points. These deletions are selected based on the significance of the point in trajectory preservation. The quality of these sampling techniques is measured by estimating the error committed using two metrics: the Mean Square Error of the Average Spatiotemporal Distance (APD) which based on average performance and the Maximum Absolute Error of the highest APD for the entire trajectory which capture deviations from the original trajectories. The aforementioned sampling algorithms are compared with uniform sampling, a well-known technique for managing massive trajectories. Uniform sampling downsamples a trajectory at fixed time intervals in order to achieve a desired compression ratio. The results show that Threshold and STTrace outperform Uniform Sampling in terms of preserved data quality. Furthermore the robustness of the proposed algorithms is evaluated by performing spatiotemporal queries (trajectory self-joins). Again, the proposed algorithms outperform Uniform Sampling. In general, STTrace outperforms Threshold for small compression rates but exhibits some overhead in maintaining the minimum synchronous distance and in-memory adjustment of the sampled points. This overhead can be reduced when STTrace is applied in offline mode. However STTrace presents high error rates and computational costs compared to other compression algorithms such as top down time ratio [8].

In [14] seven different compression algorithms are compared and two different error metrics are employed, SED and the median difference in speed. The algorithms are grouped based on processing mode (batch vs online) and the error metrics. In order to evaluate the

compression algorithms, two different datasets and three transportation modes are used that contain pedestrian, vehicle and multi-modal trajectories. The performance comparison is based on the execution times of compression algorithms and on the error committed. The results showed that the vehicle (bus) mode presents the highest degree of error for both SED and median speed difference followed by the pedestrian travel mode, while the multimodal dataset exhibits the least amount of error. The significant difference in the compression performance presented between the travel modes, suggests that selecting a compression algorithm is an application-specific process.

An online compression algorithm called SQUISH is presented in [9]. SQUISH is a similar approach to STTrace, differing on the criteria of adding a point in memory. The algorithm requires an input parameter which defines the size of a buffer that contains the number of points that will exist in the final data after compression. The method prioritizes the most important points of a trajectory stream and preserves the speed information at high accuracy. All incoming points are stored in a buffer until it is full, and any new incoming point requires the removal of another point that is already stored in the buffer. The removed point is that with the minimum estimated SED error, which will introduce the lowest amount of error in the compression. When the point with the lowest estimated SED error is removed, the algorithm estimates the upper bound SED error of the adjacent neighbors by adding the SED value of the deleted point. SQUISH is compared against three other algorithms, Uniform sampling, Douglas-Pecker and Dead Reckoning. The error metrics used to evaluate the algorithm include SED, speed and performance. SQUISH presents an improved performance with respect to SED in case of low compression rates and a superior performance concerning speed error. The main drawback of the algorithm is that the estimation of SED error of points is not based on accurate heuristics. The algorithm cannot compress trajectories while ensuring that the resulting error is within a user-specified bound. Furthermore, the local prioritization of points in the stream is unable to handle large compression rates and exhibits relatively large errors when the compression ratio is high. Thus, the same authors proposed the descendant algorithm called SQUISH-E [8], which controls the compression with respect to compression ratio and accuracy.

In the vast majority of the aforementioned trajectory compression algorithms, the performance is measured only on the reduction rate by the compression process. Authors in [15] present a lossy compression batch algorithm for GPS trajectories that considers both line simplification and quantization in the compression process. The quantization technique can improve the encoding procedure that takes into account speed and direction changes for selecting the approximated trajectory for compression. For the evaluation of the compression, the maximum SED is used and the results show an improved performance comparing to time ratio algorithm presented in [10].

A model that estimates the effect of trajectory compression in query results is presented in [16]. While most of the compression algorithms provides a formula to measure and minimize the error, the authors suggest that the mean error occurred in query results after the compression is more important concerning decisions about the suitability of the compressed data, compression rates, etc. Two types of error, false negatives and false positives, are introduced and a lemma that estimates the average number of false hits per query which depends on SED, of both error types. The model introduces a small overhead in the compression procedure while it is highly accurate regarding the number of false negatives and false positives.

Another compression method is presented in [17], which is based on time mapping between similar trajectories. This method exploits the similarities between sub-trajectories

and creates a time mapping by finding for each time value of a trajectory, the corresponding time value of a similar trajectory using linear interpolation. Then a compression algorithm is applied on the time mapping, which removes the points but keeps the compression error under a user-defined threshold. The strong correlation between time values of similar trajectories, allows high compression of time mapping. The proposed algorithm achieves higher compression ratios over existing solutions such as TrajStore and time ratio algorithm. TrajStore [18] is a dynamic storage system which is able to co-locate and compress spatially and temporally “close” adjacent trajectories. The compression algorithm combines two schemes: a lossless delta compression scheme which encodes spatiotemporal coordinates within a trajectory and a lossy compression scheme for clustering trajectories traveling on nearly identical paths. The results demonstrated that the compression ratio achieved is quite high when cluster group compression combined with delta compression of time-stamps and the proposed system can support large scale analytics over trajectories.

Authors in [19] present an evaluation of seven lossy compression algorithms. Several structural and performance characteristics were compared against five different user defined thresholds, aiming to identify the most important aspects concerning the selection of the appropriate compression algorithm. The results indicate that the application scenario must be examined before selecting the appropriate compression algorithm. However, considering the examined characteristics, Dead-Reckoning algorithm presented the most suitable performance.

A novel data-driven framework, called REST which is able to compress the spatio-temporal information of trajectory data is presented in [20]. The proposed framework extracts a small collection of subtrajectories, called reference trajectories, from raw trajectory data that form the compressed trajectory within a given spatio-temporal deviation threshold. These reference trajectories are selected according to three different techniques. In addition, greedy and dynamic algorithms achieve an optimal compression ratio and high levels of efficiency. The same authors present in [21] a hybrid index structure for efficient query processing over compressed trajectories without fully decompression of trajectory data. The reference trajectories are indexed directly with an in-memory RTree and building connections to the raw trajectories with inverted index. Thus, the algorithm is able to answer spatio-temporal range queries over trajectories in their compressed form.

Finally, The concept of semantics in trajectories is presented in [22]. The movement of an individual is described with elements of the transportation network annotated with significant elements instead of a large number of points. The semantic compression enables extreme compression rates by storing only the points that represent key events such as transport stops, but it is effective only for applications that have fixed requirements.

2.2 Mining techniques

There are many data mining methods in the literature for trajectory data analysis. In this paper we evaluate classification methods, and similarity measures, that are the basis for trajectory clustering. In the following we describe the main works for trajectory classification in Section 2.2.1 and for Similarity Measures in Section 2.2.2.

2.2.1 Trajectory classification techniques

Trajectory classification is the task that identifies the class label of the moving object based on its trajectories [23]. There is a number of techniques for trajectory classification and for many different purposes, such as identifying an animal specie, determining a hurri-

cane level, predicting the transportation mode of the moving object, etc. All works extract different trajectory features to use them as input for trajectory classifiers.

The main difference between the trajectory classification techniques is the type of trajectory features they extract for training the classification model [24]. Most works in trajectory classification extract features from the spatio-temporal properties of trajectories, as the speed, acceleration, direction change, etc.

One of the first trajectory classification methods was TRACCLASS [23]. Three main works extract *local features* from trajectories, i.e., attributes from trajectory parts. Bolbol [25] segments the trajectories in a pre-defined number of subtrajectories and uses a sliding window to cover a certain number of subtrajectories. From the subtrajectories covered by the sliding window, it extract features as the average acceleration and average speed. Soleymani [26] segments the trajectories by using two types of grids. The first grid splits the trajectories based on their spatial location, and the technique extracts the time duration of the subtrajectories inside each grid cell. The second grid divides the trajectory by using a time window, and the technique calculates features as the standard deviation of speed and maximum turning angle from each subtrajectory inside a grid cell. Dabiri [27] calculates four features from sequential trajectory points (speed, acceleration, direction change and stop rate), and represents the trajectories as a vector of four dimensions, where each vector dimension is a the sequence of a given feature value.

The works of Zheng [28], Sharma [29] and Junior in [30] extract *global features*, i.e., features from the entire trajectory. Zheng [28] and Junior [30] differ from each other mainly on the statistics extracted from the trajectory points, and in the classification model trained for classifying trajectories. While [28] focuses on calculating different features as the maximum speed, the heading change rate, the stop rate for training a Decision Trees (DTs), Junior [30] extracts the maximum, minimum and average speed, direction change and traveled distance for training an Active Learning model. Sharma [29] uses the Nearest Neighbour Trajectory Classification (NNTC), where a trajectory is assigned to the same class as its neighbour, i.e., the closest trajectory.

Dodge in [31] calculates the features of speed, acceleration and direction change between every two consecutive trajectory points, and the trajectories are then represented as sequences of each feature, similar to an unidimensional time series. *Local features* are extracted from subtrajectories with the same characteristics (e.g. same speed, same acceleration, etc.) and global features are statistics of the entire trajectory as the minimum speed, maximum speed, average speed, minimum acceleration, etc. Xiao [32] and Etemad [33] extract several statistics from trajectory points, as the percentiles, interquartile range, skewness, coefficient of variation and kurtosis from the speed, acceleration, and heading change.

A very recent work and that has outperformed all previous works in trajectory classification is the method Movelets [34]. This approach is different from all previous works as it extracts relevant subtrajectories, where a relevant subtrajectory is the trajectory part that better discriminates the class. The relevant trajectories are called Movelets. The movelets are extracted by evaluating each possible subtrajectory in the dataset, where their quality is higher when they are closer to trajectories of a class and further from trajectories of other classes. Movelets is able to consider all trajectory dimensions as space, time and semantics.

2.2.2 Trajectory similarity measures

Similarity measures are the basis for another important data mining task: clustering. Therefore, it is important to understand the behaviour of similarity measures as the data are

summarized. Many similarity measures have been proposed for trajectories in the last years. They can be split in two main categories: measures that deal with spatial or spatio-temporal dimensions only and measures that can deal with any dimension, so supporting also trajectory data enriched with semantic dimensions other than space and time.

Furthermore, according to [35], trajectory similarity measures can be further classified into four categories (continuous sequence-only measures, discrete sequence-only measures, continuous spatiotemporal measures, and discrete spatiotemporal measures) based on whether the trajectory data are discrete or continuous and whether the measure considers temporal information.

The most well known similarity measures developed for raw trajectories are LCSS (Longest Common SubSequence) [36], EDR (Edit Distance on Real sequences) [37], CVTI [38], and UMS (Uncertain Movement Similarity) [39]. LCSS uses a user-defined matching threshold to determine if two trajectory points are similar. When the distance between two points is lower than this given threshold there is a match. The global similarity of two trajectories is computed by the number of matching point pairs divided by the number of points of the shortest trajectory.

EDR computes the distance from one trajectory to another based on the number of changes (not matched points) that are necessary to transform one trajectory in another. The problem of EDR is that two similar trajectories with different sizes may have a high distance score.

CVTI computes the similarity between two trajectories by splitting the space in cells. If two trajectories crossed the same cell during the same time interval, they will be considered more similar because there is a common time interval.

UMS is a more recent trajectory similarity measure, which does not use a threshold for matching trajectory points. It builds ellipses between two trajectory points, so the ellipses size varies as the distance between two trajectory points changes, building large ellipses for distant points and small ellipses for closer points. The similarity between two trajectories is measured by the proportion of intersection of their ellipses. This measure has outperformed all previous works for trajectory similarity, and is so far the best measure for raw trajectory similarity.

Only a few measures consider both the spatio-temporal dimensions and textual trajectory attributes. The main works are MSM (Multidimensional Similarity Measuring) [40], SMSM (Stops and Moves Similarity Measure [41], and MUITAS (Multiple Aspect Trajectory Similarity) [42]. As semantic features are normally not added to each trajectory point, but to a subset of points, and the trajectories are in some sense summarized (e.g. check-in trajectories), they do not need to be compressed.

As raw trajectories are those with a larger number of points than semantic trajectories that are summarized, they need to be compressed in order to allow data analytics for real data. Therefore, in this work we will evaluate the impact of trajectory compression on the best similarity measures for raw trajectories.

3 Problem formulation

Compression algorithms aim to minimize the amount of data and hence storage requirements. Most works on trajectory compression do only analyse the compression itself, without evaluating the compression impact over trajectory data mining or analysis tasks. Nevertheless, compression introduces information loss, commonly known as error. The

error is measured through metrics that calculate the applied error over a trajectory if a certain point is discarded. A compression algorithm is considered suitable when it reaches high compression rates with a low amount of applied error. Trajectory mining techniques can benefit from compression approaches but current works focuses on the optimization of the compression rate. In this work the objective function includes, apart from the compression rate, the data mining performance metrics (minimization of the error). The objective of this work is to investigate and quantifying the relationship as well as the impact of trajectory compression techniques over trajectory classification and similarity analysis.

3.1 Validation criteria

In order to measure the quality of the compression algorithms, the results of each algorithm must be validated. When evaluating the effectiveness of each algorithm, both spatial and temporal accuracy must be taken into account as the temporal component t is also stored along with the spatial location. Applications using stored GPS data often require the preservation of additional components such as speed, heading and acceleration. The main validation solutions presented in the literature, include performance and accuracy metrics. Performance metrics are used for comparing the efficiency while accuracy metrics are used to measure information loss and therefore the accuracy of compared compression algorithms.

3.1.1 Performance metrics

For performance we consider the following metrics: a) compression ratio and b) compression time. Compression ratio refers to the ratio of the points remaining after compression (m) compared to the original points (n), being defined as follows:

$$\text{compression ratio} = \left(1 - \frac{m}{n}\right) \cdot 100\% \quad (1)$$

Compression time refers to the total time required for the compression execution, and it is an important index to measure the efficiency of each trajectory compression algorithm.

3.1.2 Accuracy metrics

To measure the accuracy, several error metrics such as spatial distance (SD), synchronized Euclidean distance (SED), heading and speed are employed to measure the difference between a trajectory T and its compressed representation T' . Compression algorithms attempt to minimize one or more of the above error metrics.

The spatial error of T' with respect to a point P_i in T is defined as the perpendicular distance between this point and its estimation P_i' . If T' contains P_i then P_i' coincides with P_i , otherwise P_i' is defined as the closest point to P_i along the line segment composed by the precursor and successor point of P_i in T' .

The main drawback of the spatial error is that it does not incorporate temporal data. Synchronized Euclidean Distance (SED) overcomes this limitation. SED is defined as the distance between P_i and its estimation P_i' , which is obtained by linear interpolation, owning the same time with P_i . Figure 2 illustrates SD and SED error metrics. Additionally, both speed and heading errors are determined in a way similar to SED, except that they capture the difference in speed and heading between the actual and estimated movements.

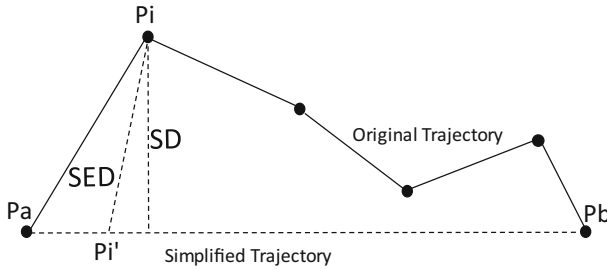


Fig. 2 Spatial distance and SED error metrics

Various research works evaluate the compression algorithms according to different error metrics. For example in [10] the error is provided by a formula which measures the mean error of the compressed trajectory representation in terms of distance from the original trajectory. In [9] the average and median SED as well as the speed error are employed in reference to compression ratio while in [14] the same metrics are used with the extra addition of heading error. Furthermore [8] utilizes the average SED, spatial and speed error according to the compression ratio.

As there is a trade-off between the achieved compression rate and the acceptable error, we also evaluate the error in terms of quality obtained from classification techniques and similarity measures. To evaluate these techniques we used the Accuracy (ACC), as it is commonly used as classification metric. The ACC indicates how well a classification algorithm can discriminate the classes of the trajectories in the test set, and it is calculated with Eq. 2, described below:

$$ACC = \frac{|Correctly\ Labeled\ Examples|}{|Test\ Set|} \quad (2)$$

For the similarity analysis, we use the F-Score, which measures whether the most similar trajectories of a given trajectory are from the same class, and it is given by Eq. 3:

$$F - Score = 2 \cdot \frac{P \cdot R}{P + R} \quad (3)$$

where the F-Score consists on the harmonic mean of Precision (P) and Recall (R), where P is the average precision in each class, and R is the average recall in each class.

4 Top-down compression algorithms

As mentioned in Section 2, compression algorithms can be grouped into four categories: top-down, bottom-up, sliding window and opening window. In this work five top-down algorithms are evaluated:

1. Douglas-Peucker (DP)
2. Time Ratio (TR)
3. Speed Based (SP)
4. Time Ratio with Speed Based combined (TR_SP)
5. Speed Based with Time Ratio combined (SP_TR)

Table 1 depicts the basic characteristics of the algorithms compared in this study.

Table 1 Basic characteristics of trajectory compression algorithms

Algorithm	Mode	Error metric
<i>DP</i>	Batch	Spatial distance
<i>TR</i>	Batch	SED
<i>SP</i>	Batch	Speed difference
<i>TR_SP</i>	Batch	SED + Speed difference
<i>SP_TR</i>	Batch	Speed difference + SED distance

4.1 Douglas Peucker (DP) algorithm

One of the most famous top-down algorithm is Douglas-Peucker [43], which was originally developed for line simplification. The process of this algorithm is illustrated in Fig. 3. The first step of the algorithm is to select the first point as the anchor point and the last one as the float point (Fig. 3a). The starting curve is a set of points and a threshold (distance dimension) $\epsilon > 0$. For all the intermediate data points, the perpendicular distance (\perp) of the farthest point (P_1) and the line segment between the start and the end points ($LS a \rightarrow f$) is determined. If the point is closer than ϵ to the $LS a \rightarrow f$, all the points between a and f are discarded. Otherwise the line is cut at this data point that causes the maximum distance and this point (P_1) is included in the resulting set. P_1 becomes the new float point for the first segment, and the anchor point for the second segment (Fig. 3b). The same procedure

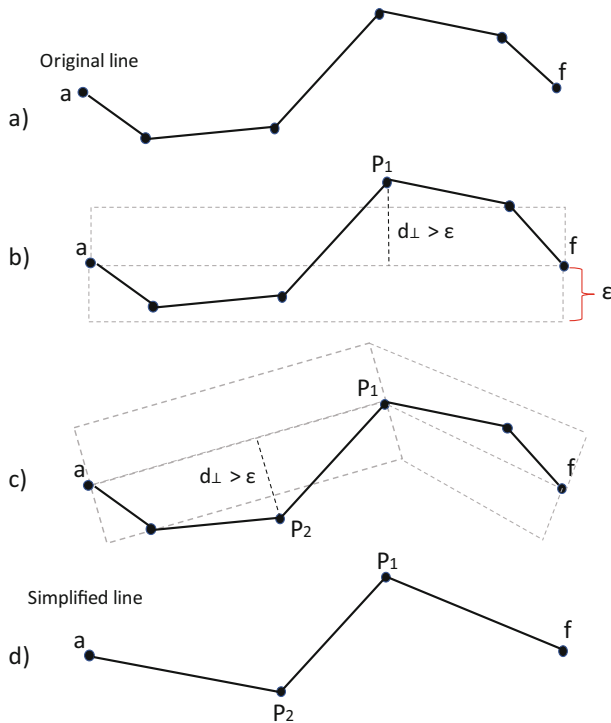


Fig. 3 Douglas Peucker algorithm

is recursively repeated for both segments until all points are handled (Fig. 3c). When the recursion is completed, a new simplified line is generated of only those points that have been marked as kept (Fig. 3d).

The main drawback of *DP* algorithm is that it does not consider the temporal aspect of the trajectory. The spatial error used which finds the closest point on the compressed representation of a trajectory to each point on the original trajectory and a trajectory is treated as a line in two-dimensional space. But a trajectory has an important extra dimension, time.

4.2 Time ratio (TR) algorithm

The Synchronous Euclidean Distance (SED) error, which is employed by the time ratio algorithm, finds the distances between pairs of estimated temporally synchronized positions, one on the original and one on the new trajectory as illustrated in Fig. 4. The temporally synchronized point P_i' of P_i is located on the new trajectory $Tr_{P_a-P_b}$ and the coordinates (x'_i, y'_i) of P_i' are calculated using linear interpolation as shown in Eq. 4. Db and Di time intervals indicate travel time from P_a to P_b and from P_a to P_i respectively. The SED error for two points is measured using the Eq. 5. The (x_i, y_i) and (x'_i, y'_i) represent the coordinates of a moving object at time t_i and synchronized time t'_i in the uncompressed and compressed traces respectively. In the final step if the distance between P_i' and P_i is greater than a threshold, the particular point is included in the resulting set otherwise discarded. The main advantage of this algorithm, is that the temporal factor is included, thus providing a more accurate compression.

$$\begin{aligned}
 Db &= t_b - t_a \\
 Di &= t_i - t_a \\
 x'_i &= x_a + \frac{Di}{Db}(x_b - x_a) \\
 y'_i &= y_a + \frac{Di}{Db}(y_b - y_a)
 \end{aligned}
 \tag{4}$$

$$SED(P_i, P'_i) = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}
 \tag{5}$$

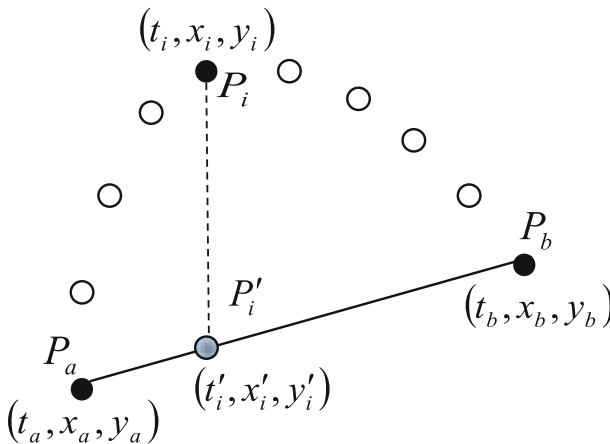


Fig. 4 Original data points (open circles) and P_i 's synchronized point position P_i' in the new trajectory

4.3 Speed based (SP) algorithm

As Meratnia and Rolf stated in [10], several algorithmic improvements can be achieved from hidden spatiotemporal information extracted from the dataset. The speed based algorithm exploits the speeds from subsequent segments of a trajectory. Neither of the dataset used in this work contain the speed information. The speed was derived from the positions (coordinates) and timestamps using the formula of velocity $v = Dx/Dt$. If the absolute value of speed $|v_{i+1} - v_i|$ of two subsequent segments is greater than a threshold, the point in the middle is retained otherwise discarded. The algorithm is illustrated in Fig. 5.

4.4 Combined algorithms: TR_SP & SP_TR

By combining the time ratio with the speed based algorithm, a new class of algorithms can be derived that exploit the concept of both algorithms simultaneously as proposed in [10]. For retain or discard a point, two thresholds are used, a distance (SED) threshold and a speed threshold. As a result, two algorithms can be extracted depending on the decision of which threshold would be exploited first. The decision extracts TR_SP or SP_TR in case of distance followed by speed threshold or the opposite. In case of TR_SP, if the distance or speed is greater than a distance and speed threshold respectively, the point is included in the data set otherwise discarded. On the other hand, in case of SP_TR if the speed or distance is greater than a speed and distance threshold respectively, the point is included in the data set otherwise discarded.

4.5 Defining the thresholds

In general, top-down algorithms split the data series based on an approximation error. If the latter is above a user-defined threshold, the algorithm recursively continues to split the resulting subseries until the approximation error in subsegments is below the threshold [44]. Setting the threshold of a compression algorithm means always making a trade-off between the compression rate and quality.

One of the challenges of this work is to define the thresholds to be employed by the examined algorithms, and specifically the epsilon (ϵ) threshold in DP, SED threshold in TR, speed threshold in SP and the combined synchronized distance/speed threshold in TR_SP and SP_TR. In general, the compression algorithms: (i) group the points and create a trajectory of each object based on an identifier. This practically means that the number of

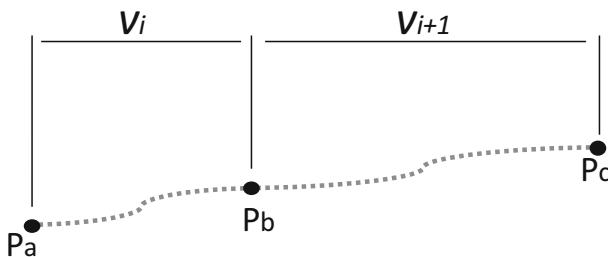


Fig. 5 Speed based algorithm, v_{i+1} and v_i are related to the speeds of the subsequent segments of a trajectory STr_{Pb-Pc} and STr_{Pa-Pb} respectively

trajectories in a dataset is as large as the number of objects; (ii) compress the whole trajectory of each identifier; (iii) write the points remaining after compression to a file grouped by identifiers.

As we group the points for each identifier (object) in the dataset, we extract the trajectories (one trajectory for each object). In order to determine the threshold that each algorithm will use, we propose a dynamic process: for each individual trajectory a different threshold is automatically defined based on an average. Specifically, we calculate the average epsilon (DP), average SED (TR), average speed (SP), average SED/speed (TR_SP and SP_TR) for each trajectory before any of the respective algorithms is applied. Then we use this average value (AVG) as a reference point to define a common rule for the threshold calculation.

This practically means that in every trajectory of each dataset a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of this trajectory. Thus, we have eliminated the need of arbitrary user-defined thresholds.

In order to “see the big picture” and observe the behaviour of the compression algorithms, a set of threshold rules is defined based on the AVG as follows:

- $AVG/8$
- $AVG/4$
- $AVG/2$
- **AVG (average)**
- $AVG + AVG/2$
- $2AVG$

5 Evaluation

In this section the algorithms are evaluated using four real datasets and six different threshold values. Section 5.1 describes the datasets, Section 5.2 presents the mining techniques (classification and similarity) and Section 5.3 presents the experimental setup. Finally, Section 5.4 describes the results and the evaluation regarding the compression algorithms and the mining techniques employed.

5.1 Datasets

According to [12] the derivation of trajectories can be classified into four major categories regarding to mobility: people, vehicles, animals and natural phenomena. There are a few real trajectory datasets that are publicly available and address the above categories. The datasets used for evaluating the compression trajectory algorithms were selected according to some criteria:

1. the data are categorized in classes, so they can be used to evaluate both classification problems and similarity measures.
2. they contain trajectories of different objects (people, vehicles, hurricanes, and animals).
3. they present variations in their sampling rates, ranging from highly sampled to very low sampling rate.

In the following we present a detailed description of the trajectory datasets. Many of these datasets have already being used in [24] and [34] for comparing several classification methods.

5.1.1 Microsoft GeoLife dataset

This dataset contains dense pedestrian and car trajectories. Specifically, this GPS trajectory dataset was collected in (Microsoft Research Asia) GeoLife project by 182 users in a period of over three years (from April 2007 to August 2012). This dataset recorded a broad range of users' outdoor movements, including not only life routines like go home and go to work but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking and cycling. This trajectory dataset can be used in many research fields, such as mobility pattern mining, user activity recognition, location-based social networks, location privacy, and location recommendation. This dataset has a high point sampling rate, and some issues as outliers and duplicated data needed to be fixed before applying data mining techniques. The pre-processing in this dataset consists of: (i) removing duplicated records, (ii) splitting the trajectories where two consecutive points had more than 300 seconds of difference between them, as it represents a large gap in such dense dataset, (iii) removing trajectories with less than 100 points, as it represents a very small portion of time due to the density, (iv) excluding the transportation modes with too less trajectory examples, which are the classes of airplane, boat, running and motorcycle, (v) removing trajectories with unreal average velocity given the transportation mode, as for instance, trajectories labeled with walk with average velocity of more than 10m/s, and, finally, from the remaining trajectories, (vi) a proportion of 20% of the trajectories of each transportation mode was selected, as some techniques were unable to perform in a reasonable amount time. This dataset involves several transportation modes as cars, bike, bus, and foot and is highly sampled [28, 45, 46].

5.1.2 Hurricane dataset

This dataset³ is provided by the National Hurricane Service (NHS) and is related to Atlantic Hurricanes Database collected between 1950 and 2008. It contains sparse points with hurricane trajectories, where the hurricanes are classified in strength level. It has a low sampling rate and the class is the degree of the hurricane.

5.1.3 Animals dataset

A dataset⁴ with sparse points which contains the trajectory information of three animal species from Starkey project. In this dataset, major habitat variables derived for radiotelemetry studies of elk, mule deer and cattle at the Starkey Experimental Forest and Range, 28 miles southwest of La Grande, Oregon. It has a low sampling rate and the class is the type of animal.

5.1.4 UFC (Federal University of Ceará) dataset

This anonymous dataset contains movements data in the area of Fortaleza, Brazil. Also includes movements data (smaller fraction) from Parnaíba, São Paulo, Natal, Maceió, Rio De Janeiro, Manaus, Belo Horizonte, Campos dos Goytacazes, Macaé and Oberstadt (Germany). It is a dense trajectory dataset with highly sampled trajectories.

³Hurricanes dataset, <https://www.nhc.noaa.gov/data/>

⁴Animals dataset, <https://www.fs.fed.us/pnw/starkey/data/tables/>

Table 2 provides information about the datasets used for evaluating the trajectory compression algorithms.

5.2 Mining techniques

We used the compressed trajectory datasets in tasks of data mining in order to evaluate the capability of extracting meaning from the trajectories even after their compression. The data mining techniques that were used can be divided in classification techniques (Section 5.2.1) and similarity measure techniques (Section 5.2.2).

5.2.1 Trajectory classification techniques

The classification techniques aim to identify the classes of the trajectories by extracting features that are capable of distinguishing the classes. In these experiments we used four methods for extracting trajectory features, which were selected because of their best classification accuracy. Table 3 summarizes the classification methods and the trajectory dimensions they support and the features they extract from trajectories and that are input for the classification algorithms.

5.2.2 Evaluated similarity measures

We evaluate the compression results on the measures EDR, which overcame some limitations of LCSS, and UMS. Table 4 summarizes the similarity measures showing the trajectory dimensions they support and how they measure the similarity between two trajectories.

The EDR technique identifies how many changes are needed on a trajectory for turning it into another trajectory, where the more changes, the more dissimilar the trajectories are. The UMS, otherwise, generates ellipses between consecutive points of each trajectory, and the similarity between two trajectories is measured by the area shared by their ellipses.

5.3 Experimental setup

In the classification tasks, after the extraction of the features, the Random Forest (RF) is trained and used to compare the techniques, once it is fast and commonly used by the techniques in the literature. The works of [34] and [47] have compared several classifiers including SVM (Support Vector Machine), MLP (Multi Layer Perceptron) and random forest, and they have shown a similar accuracy. We have selected RF because it performs faster

Table 2 Datasets overview

Dataset	Trajectories	Points	Sampling rate	Type
GeoLife	1,763	953,966	high	dense
Hurricane	1003	26,783	low	sparse
Animals	253	287,136	low	sparse
UFC	411	888,810	high	dense

Table 3 Summary of the classification techniques, with the trajectory dimensions they use, and the features they extract from trajectories

Technique	Trajectory Dimensions	Features Extracted
Dodge [31], Zheng [28] and Xiao [32]	Space and time	Numerical features, e.g. speed, acceleration, deviation rate, etc.
Movelets [34]	Space and Time	Movelets extracted from space dimension + global features as indicated in [34]

than SVM and MLP. Random Forest was implemented in Python by using the “sklearn” package, and it was constructed with 100 decision trees with default structure.

Due to the large number of datasets, we performed the classification experiment in two stages. On the first stage, the datasets were split in hold-out manner, in a proportion of 70% for training and 30% for testing, and always respecting the class balancing. After extracting the features and training the RFs, it was possible for us to analyse and elect which compression algorithm was capable of producing the datasets with most accurate classification results. The second stage was a 5-folds cross-validation using the previous best accurate dataset. We did not perform cross-validation in every dataset due to the large number of datasets, classification techniques and different compression methods and thresholds to be evaluated. For example, the effort of completing a 5-folds cross-validation experiment is equivalent of extracting the features from 4 different dataset, multiplied by 5 possibilities, which are the different folds, using 4 different classification techniques, under 4 compression methods, with 6 different thresholds, a total of 2,400 different tests ($4 \cdot 5 \cdot 4 \cdot 4 \cdot 6 = 2,400$).

The threshold values used by each technique are those reported in the original paper. To extract the local features of [31] the sinuosity threshold is necessary to identify similar subtrajectories, and this threshold is given by the mean between the minimum and maximum sinuosity value. To extract the Zheng features that require thresholds (e.g. the heading change rate) we used the method suggested in [28] for finding the thresholds, which consists on evaluating each feature individually for finding the best set of thresholds for each dataset. In the Movelets technique the distance measure is euclidean.

The underlying computing infrastructure that has been used for the execution of the compression algorithms has been a commodity machine with the following configuration: Ubuntu 18.04 LTS 64-bit; Intel Core i7-8550U CPU @ 1.80GHz × 8; and 16 GiB RAM.

Table 4 Summary of the similarity measure techniques, with the trajectory dimensions they use, and how they measure the similarity between two trajectories

Technique	Trajectory Dimensions	Similarity
EDR	Space	How much a trajectory needs to change to become another trajectory
UMS	Space	The area between the ellipses of two trajectories.

5.4 Results and discussion

5.4.1 Compression evaluation

In order to evaluate the trajectory compression algorithms, we measured the compression rate obtained based on the six thresholds defined in Section 4.5. The experiments were performed over four datasets.

According to Fig. 6, *DP* and *TR* achieve high compression rates for all thresholds in the UFC dataset with the lowest value being 78.87% concerning *AVG/8* and the highest being 96.34% for *2AVG* in the *DP* algorithm. *TR* exhibits a similar behaviour. Compression is slightly lower in the *TR* algorithm compared to the *DP* but nevertheless it remains high. As can be seen, for *DP* and *TR* a change in the threshold value does not dramatically impact the compression rate. Algorithms that consider time information in order to decide whether or not to discard a point, tend to preserve more points in a trajectory. As mentioned in Section 4.1, the *DP* algorithm treats a trajectory as a line in a two-dimensional space and does not consider the temporal aspect. Thus, as it is shown in the results, the compression obtained by *DP* possess the highest rates with minor deviations, almost in all datasets and thresholds. *SP*, *TR_SP* and *SP_TR* present similar compression rates when the same thresholds were applied. For the *AVG* threshold and below, these algorithms present bigger fluctuations regarding compression compared to *DP* and *TR*. It can be observed that the latter exhibits a relatively small but constant downward trend. For example the compression rate obtained for *DP* and *SP_TR* in *2AVG* is 96.34%, 87.87% while in *AVG/8* is 78.87%, 35.23% respectively. The difference between the highest threshold to the lowest is 17.47% in *DP* while in *SP_TR* is 50.64%, almost 3 times larger. Generally, the lower the threshold value, the lower the achieved compression rate as more points are included in the resulting set.

As shown in Fig. 7, almost the same behaviour as above is observed in the Animals dataset. *DP* and *TR* present high compression rates with small variations while the remaining algorithms present similar compression rates when the same thresholds were applied. Only in *2AVG* and *AVG+AVG/2* thresholds, the compression differs by a more significant proportion between *SP*, *TR_SP* and *SP_TR*. Compared to the compression results obtained from UFC, the experiments show a higher sensitivity towards thresholds for compression where the reduction is more noticeable and furthermore the compression are lower regarding all algorithms and thresholds. As an example, the compression obtained in Animals for

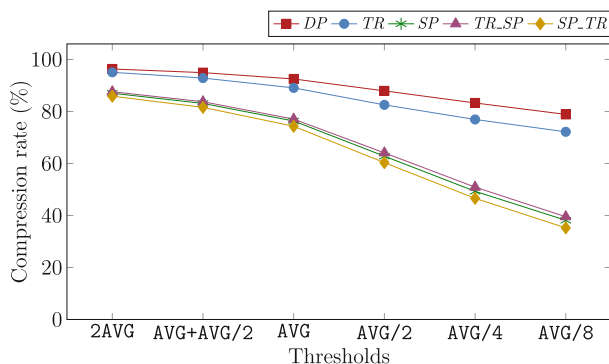


Fig. 6 Compression rate for different thresholds, UFC dataset

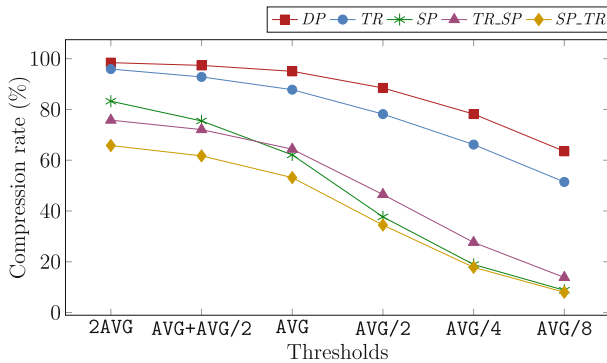


Fig. 7 Compression rate for different thresholds, Animals dataset

SP.TR algorithm in case of *AVG/8* is 7.97% while in *UFC* is 35.23% when the same algorithm and threshold are applied. Largest variations are presented for *AVG/2* and below in *SP*, *TR.SP* and *SP.TR* where the compression rates are almost 4x smaller than in *UFC*.

Regarding the Hurricane dataset the results are illustrated in Fig. 8. In case of the threshold *2AVG* the compression rate is almost the same for all algorithms. The *SP* algorithm presents the highest compression rate for thresholds *2AVG*, *AVG+AVG/2* and *AVG* while *DP* and *TR* outperform the rest of the algorithms for the remaining thresholds. The compression rates are maintained at a very high level for thresholds *AVG* and above with the lowest value to be 81.47% in *SP.TR*. As the results show, a change in threshold value hardly impacts the compression rate for the aforementioned thresholds. Furthermore, the difference in compression rates between algorithms, is more noticeable in case of the smallest two thresholds, *AVG/4* and *AVG/8*.

Finally, Fig. 9 presents the compression rates achieved in the GeoLife dataset. For the *2AVG* threshold, the compression presents quite small variations among algorithms. The compression for *DP* and *TR* is almost constant and maintained at a sufficiently high level. Furthermore, the compression is also very high for the remaining algorithms for threshold *AVG* and above, with the lowest value to be 93.82% in *SP*. In comparison with Hurricane dataset, GeoLife presents higher compression especially for threshold values below *AVG/2* and achieves the highest compression among all datasets.

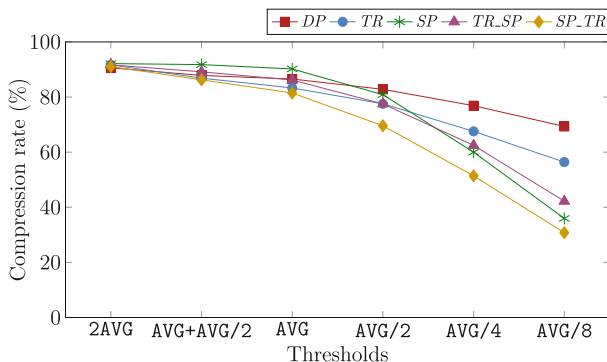


Fig. 8 Compression rate for different thresholds, Hurricane dataset

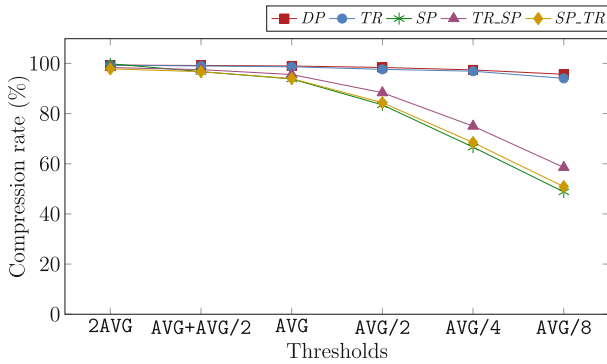


Fig. 9 Compression rate for different thresholds, GeoLife dataset

Besides the compression rate, it is important to assess the execution time of each algorithm as constitutes an important index to measure efficiency. Figure 10 presents the execution times of all algorithms for the different datasets. DP and TR present the lowest execution among all algorithms. In fact, DP exhibits the lowest execution for all threshold values in all datasets. One advantage of DP compared to TR is that the former reaches higher compression rates in less execution time than the latter, but on the other hand TR presents higher accuracy. SP along with DP and TR had similar behaviour, increasing linearly the execution time when the threshold decreases as more points are included in the resulting set. SP spent considerable more time than DP and TR especially for threshold values equal or lower to the AVG threshold. However, SP does present better accuracy in most of the cases.

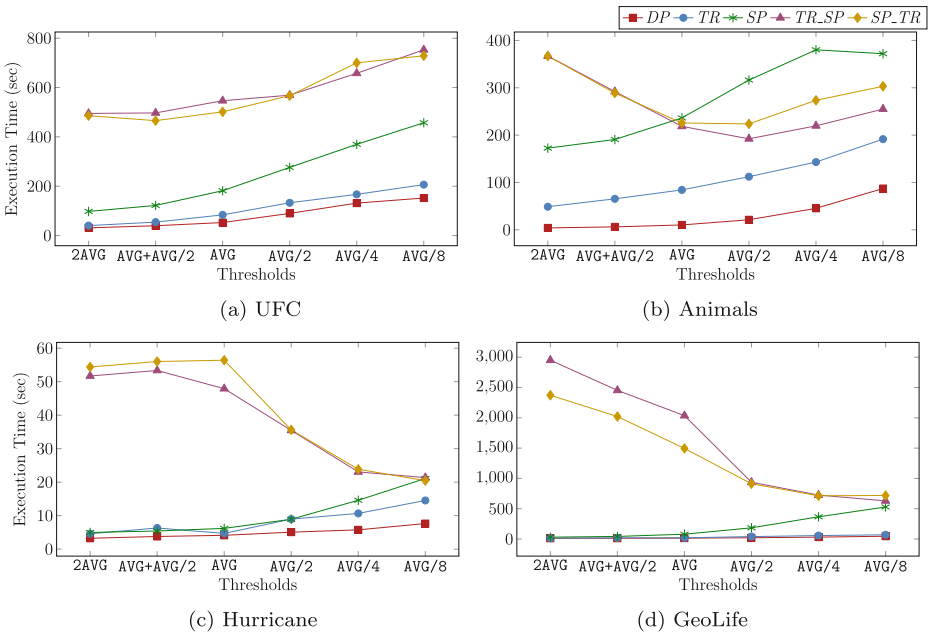


Fig. 10 Execution times

In the UFC dataset the execution of *TR.SP* and *SP.TR* also follows this norm (increased execution time as threshold decreases), but the opposite behaviour is observed in the other datasets. The execution time of the aforementioned algorithms presents a downward trend as lower values of the threshold are selected. As expected, *TR.SP* and *SP.TR* have the worst performance due to their exponential complexity but present higher accuracy, at least in most cases. On the other hand, the compression is lower especially for threshold values equal to *AVG/4* and below. However, *TR.SP* achieved a bit higher compression rate than for every analyzed threshold. Finally, the execution time in GeoLife dataset is quite high compared to the other datasets. The reason is that GeoLife is a dense dataset which contains a large number of trajectories as shown in Table 2.

5.4.2 Classification evaluation

We performed the classification task by using the Random Forest classification algorithm, and the obtained results using the hold-out approach are presented in Table 5. The higher values in each line are presented in bold, while the second highest is underlined. Once we have an accuracy value for each threshold (six in total) for each of the compression techniques (five in total), four classification methods and four datasets, which lead to a total of 480 accuracy results (5x6x4x4), it was necessary to summarize the results in order to enable the presentation. Thus we presented, for each classification technique, and for each dataset, the accuracy result by using the original dataset in the first line, and the maximum and minimum accuracy achieved by any of the compression techniques in the second and third lines, respectively. This is the best way for presenting how is the accuracy without any compression, the best result after compression, and also how “bad” in terms of quality the accuracy can be after compression.

The first main observation is that the Movelets technique could outperform the others in every dataset when using the original dataset and by selecting the minimum accuracy among

Table 5 Summary of the classification results, with the ACC obtained from the original dataset without compression, the maximum and minimum ACC obtained from the compressed datasets

Dataset	Metric	Techniques			
		Zheng	Dodge	Xiao	Movelets
Animals	original	<u>87.17%</u>	82.05%	85.89%	91.02%
	max. ACC	91.02% (<i>SP.TR</i>)	<u>85.89%</u> (<i>TR.SP</i>)	91.02% (<i>SP.TR</i>)	91.02% (<i>SP.TR</i>)
	min. ACC	71.79% (<i>DP</i>)	65.38% (<i>TR</i>)	<u>73.07%</u> (<i>DP</i>)	74.35% (<i>DP</i>)
UFC	original	19.81%	18.22%	<u>26.02%</u>	89.9%
	max. ACC	19.65% (<i>SP.TR</i>)	17.39% (<i>SP.TR</i>)	<u>24.43%</u> (<i>SP.TR</i>)	89.64% (<i>SP.TR</i>)
	min. ACC	5.57% (<i>DP</i>)	3.45% (<i>DP</i>)	<u>6.24%</u> (<i>DP</i>)	83.13% (<i>TR</i>)
Hurricane	original	58.22%	<u>60.52%</u>	60.19%	61.84%
	max. ACC	<u>60.85%</u> (<i>DP</i>)	59.86% (<i>SP</i>)	<u>60.85%</u> (<i>TR</i>)	62.82% (<i>TR.SP</i>)
	min. ACC	54.60% (<i>SP</i>)	53.28% (<i>TR</i>)	<u>55.59%</u> (<i>DP</i>)	56.56% (<i>DP</i>)
GeoLife	original	82.33%	81.01%	<u>83.77%</u>	83.98%
	max. ACC	<u>80.26%</u> (<i>SP.TR</i>)	79.69% (<i>SP</i>)	82.14% (<i>SP.TR</i>)	<u>80.26%</u> (<i>SP.TR</i>)
	min. ACC	<u>69.73%</u> (<i>SP.TR</i>)	56.01% (<i>SP</i>)	67.10% (<i>SP</i>)	72.74% (<i>TR.SP</i>)

The classification algorithm used is the Random Forest

the compressed trajectories. It only loses when selecting the maximum accuracy among the compressed datasets in the GeoLife dataset, as it performed 80.26% by using the *SP-TR* against the Xiao technique that reached 82.14% also using the *SP-TR*.

In the UFC dataset, only the Movelets could perform a high accuracy value, which is 89.9% in the original dataset, 89.64% in the maximum value among the compressed datasets and 83.13% in the minimum value. The other techniques did not surpass 26.6% of accuracy, which is the result obtained by Xiao using the original dataset.

In the compressed datasets, the higher accuracy values were found by using the *SP-TR* and *TR-SP* compression algorithms, while the *DP* and *TR* present the lower results in accuracy. Only the Hurricane dataset does not present this behavior, where the classification techniques Zheng and Xiao had better accuracy results using the *DP* and *TR*, respectively. The best accuracy in Hurricane is observed in case of *TR-SP* for Movelets (62.82%) while the worst in case of *DP* for Zheng (60.85%). *DP* and *TR* present the lowest accuracy except in case of Zheng where the lower accuracy is obtained by *SP* (54.60%). In some cases the compressed datasets help on improving the classification results, which is the case for the Zheng, Dodge and Xiao techniques in the Animals datasets, where their results in the maximum accuracy value among the compressed datasets is higher than using the original dataset.

Once we had to suppress most of the accuracy results, and in order to present another point of view over the obtained accuracy values, we also calculated the graphs of Avg-Max-Min, which are presented in Fig. 11. This graph presents the higher, the lowest and the average value obtained by each technique in the compressed datasets. It also presents

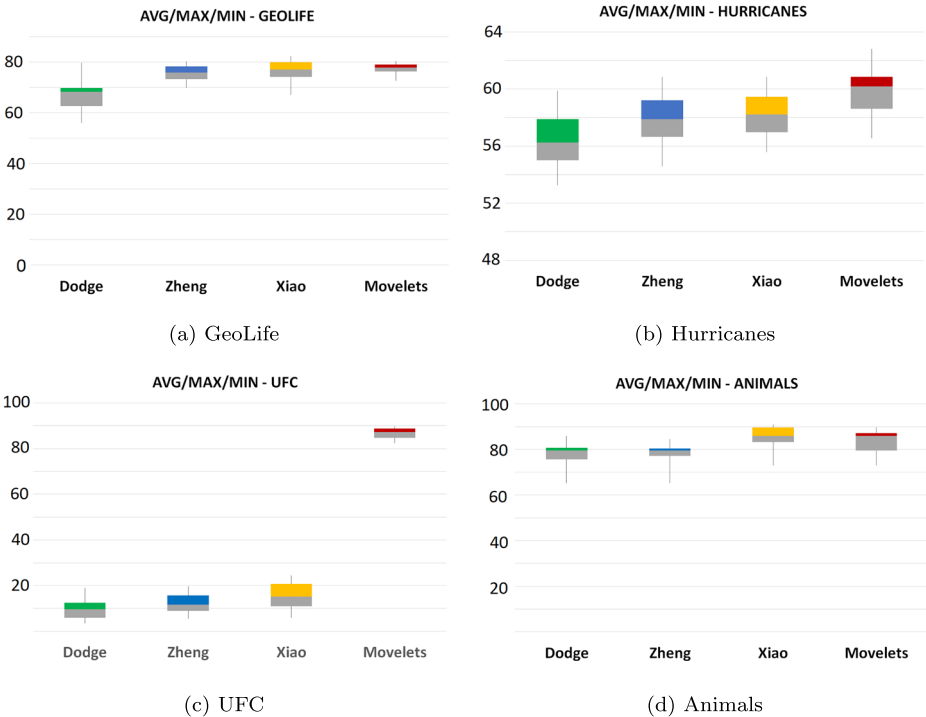


Fig. 11 AVG-MAX-MIN graphs that summarizes the accuracy values reached by each technique

the range of values in the first quadrant after and before the average value, where the first quadrant higher than the average is colored in different colors for each technique, and the first quadrant lower than the average in gray. Each quadrant is consisted by 25% of the accuracy values, or one standard deviation. It is useful for identifying which techniques had a wide variety of results when changing the compression technique and thresholds, and which techniques were more stable, presenting a less range of accuracy values.

We can observe that Movelets outperformed the other methods in the UFC dataset, where the lowest accuracy value is higher than maximum accuracy value for any technique. In the Hurricanes dataset, the average accuracy value for the Movelets is higher than all the results of the other methods in one quadrant higher than the average value. In the Animals and GeoLife datasets the average accuracy value for the Movelets is slightly higher than for the other techniques. Another observation is that the accuracy results for the Movelets technique is distributed in smaller ranges than other techniques in both GeoLife, Hurricanes and UFC datasets, indicating that even with the different threshold and compression algorithms, this technique had accuracy values that changed less than the other techniques.

Given the classification results presented in Table 5, we can identify that the compression algorithm *SP_TR* produced the most accurate classification results. It presented the higher accuracy values in 62.5% of the totality of the compared datasets, followed by the *SP* and *TR_SP* with 12.5 each, and *TR* and *DP* with 6.25 each. Given this information, we performed the 5-folds cross validation in the datasets resulting from the *SP_TR* compression algorithm, and presented the accuracy results in Table 6. In Table 5, we also present the accuracy result obtained from the Original dataset, that is, the non-compressed, from Table 6, for comparison only.

The main observation we can extract from Table 6 is that the Movelets technique is, in general, the best or second best technique in accuracy value for all datasets. As movelets has shown to be the best classification method in the literature because it is based on subtrajectory extraction, we notice that, in general, the compression algorithm has not penalized its classification results. In the UFC dataset, the results followed the previously concluded in Table 5, where the Movelets outperforms the other methods proposed by Zheng, Dodge and Xiao in large difference.

In Table 6 we can observe clearer the impacts of the compression threshold in the classification results. In general, for every classification method, the threshold **2AVG** presented the lowest classification results, while the **AVG/8** presented the highest accuracy values.

5.4.3 Similarity measure evaluation

In this section we present the results obtained by performing the similarity measure algorithms of EDR and UMS over the original and compressed datasets. In order to evaluate the results, we used the F-Score metric (P@R), which are presented in Figs. 12 and 13. Given a trajectory, this metric orders the other trajectories in ascending order of distance, and count one hit for each trajectory that is from the same class as the selected trajectory. The result obtained from the original dataset is presented in black, while the results obtained from each compression algorithm is presented in other colors. These results can be interpreted as the capacity of the similarity measures to recover similar trajectories from a given trajectory, which are expected to be those of the same class. We presented the maximum and minimum F-Score value from each compression algorithm. In this graph, the objective is to increase the area under the curve.

For both Animals and UFC dataset, the format of the curve for the EDR are similar to every technique. In the Animals, it is possible to observe that independent of the

Table 6 Summary of the classification results by using the 5-folds cross validation, with the ACC obtained from extracting the classification features from the *SP_TR* compression algorithm

Dataset	Threshold	Techniques			
		Zheng	Dodge	Xiao	Movelets
Animals	AVG	88.90%	82.98%	91.64%	92.14%
	2AVG	90.48%	82.19%	89.29%	89.36%
	AVG+AVG/2	90.89%	83.76%	90.08%	92.53%
	AVG/2	87.72%	79.80%	92.06%	91.74%
	AVG/4	87.33%	78.21%	91.67%	89.77%
	AVG/8	90.50%	78.61%	90.90%	90.17%
	Original (Table 5)	87.17%	82.05%	85.89%	91.02%
UFC	AVG	10.95%	09.71%	14.20%	81.06%
	2AVG	10.23%	07.40%	12.54%	81.18%
	AVG+AVG/2	10.73%	08.47%	11.89%	81.53%
	AVG/2	12.35%	10.93%	17.55%	82.22%
	AVG/4	13.12%	12.62%	18.63%	82.25%
	AVG/8	13.49%	12.82%	21.29%	82.57%
	Original (Table 5)	19.81%	18.22%	26.02%	89.9%
Hurricane	AVG	58.81%	55.42%	58.12%	58.93%
	2AVG	55.04%	54.63%	56.63%	59.22%
	AVG+AVG/2	57.33%	54.13%	56.13%	60.52%
	AVG/2	57.02%	57.22%	59.42%	56.91%
	AVG/4	59.21%	57.03%	59.12%	59.12%
	AVG/8	58.72%	58.22%	58.73%	59.32%
	Original (Table 5)	58.22%	60.52%	60.52%	60.19%
GeoLife	AVG	74.47%	65.73%	75.26%	79.00%
	2AVG	72.03%	58.36%	71.62%	75.77%
	AVG+AVG/2	73.79%	61.93%	73.16%	77.92%
	AVG/2	76.62%	71.41%	78.95%	79.28%
	AVG/4	78.09%	75.49%	80.31%	81.16%
	AVG/8	79.35%	77.93%	81.11%	81.22%
	Original (Table 5)	82.33%	81.01%	83.77%	83.98%

The classification algorithm used is the Random Forest

compression method, the results are all in the same range of values, in exception to the minimum values of F-Score acquired by *DP* and *TR*. In the UFC dataset, otherwise, the compression method *SP* is responsible for the lower values of F-Score. The UMS similarity technique had the characteristic of having higher values of F-Score for the compressed trajectories in the UFC dataset, where the line of the original dataset result is under most of the other lines, in exception of the minimum value acquired by the *DP* compression technique. In the Animals dataset instead the line of the original dataset is above the compressed dataset lines, but most of them follow the same line shape. In the Hurricanes and GeoLife datasets, the line results from both methods EDR and UMS follow the same shape, once the results are in majority overlapping each other.

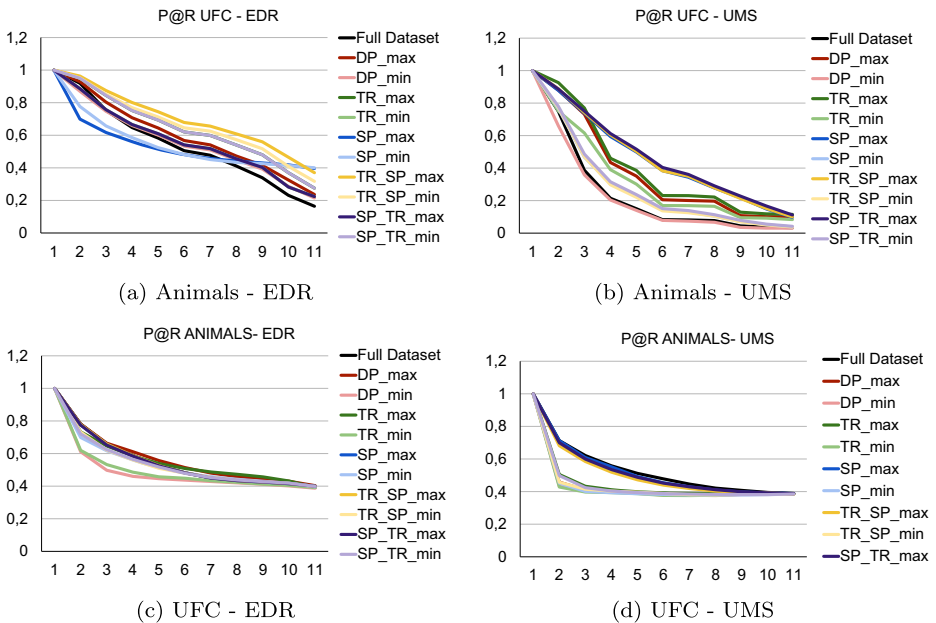


Fig. 12 F-Score results when using the similarity measure EDR and UMS in the Animals and UFC datasets

5.4.4 Discussion

In the UFC dataset we observed that the Movelets could outperform the other methods, and this occurred mainly due to the classification purpose of the UFC dataset, which aims to classify many classes of people. These people often have the same pattern of speed and acceleration when walking, and these are examples of the attributes extracted from these techniques. The Movelets, otherwise, is based on identifying the subtrajectories that better describe each person, the places where only they go, and this is shown to be better discriminant in this task. There is a trade-off between the compression rate and the achieved accuracy which leads to better quality, for example the compression obtained for *SP_TR* in AVG/8 threshold is 35.23% and for *TR* is 72.18% while the accuracy between these algorithms differs only 6% in Movelets.

In GeoLife, the highest accuracy is achieved in case of *SP_TR* for Xiao (83.77%) while Dodge has the lowest accuracy for *SP* (56.01%). *DP* and *TR* are neither in the best or worst results. However, *DP* and *TR* achieve extremely high compression for all thresholds. The lowest value regarding compression is 95.67% in AVG/8 threshold while the accuracy is 79%. The difference from the highest accuracy presented in *SP_TR* is almost only 4% while the compression increased almost twice. Therefore, unlike the other algorithms, in which a choice of threshold is important for the compression results, in the *DP* algorithm this is not the case. This allows choosing a higher threshold value to improve compression while not losing much on the quality.

About the similarity tests, some of the obtained results in the compressed dataset are higher than in the original datasets. It happens in the EDR technique due to its characteristic of evaluating how many changes are necessary in the trajectory until it becomes another trajectory. In a compressed trajectory, where the trajectories are smaller, the number of

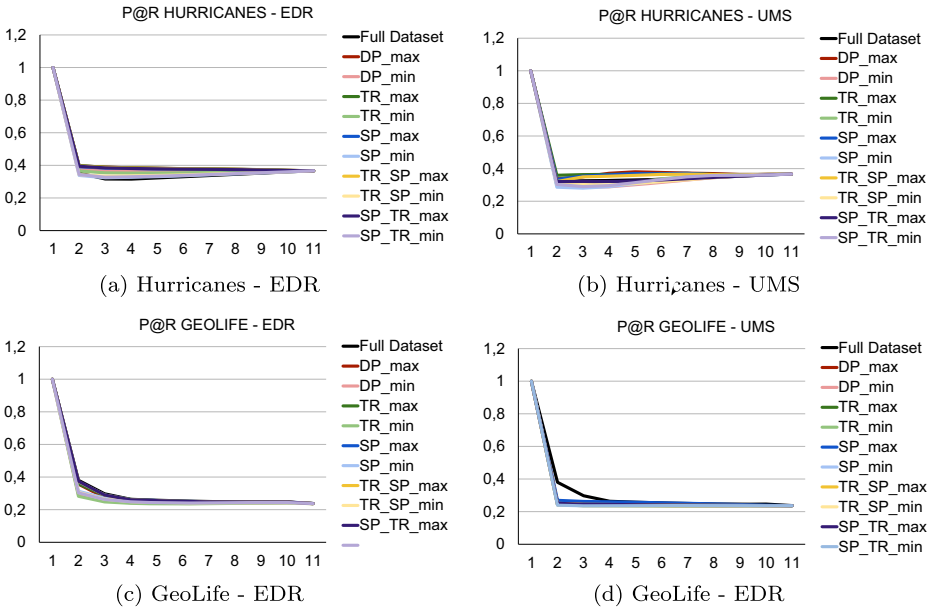


Fig. 13 F-Score results when using the similarity measure EDR and UMS in the Animals and UFC datasets

changes is smaller in comparison to the original trajectory, and has shown to be enough when recovering similar trajectories of the same class. In the UMS technique, otherwise, which was mainly proposed for datasets with low point collection rate, it is possible to observe for the UFC dataset an increase of the results in the compressed datasets, once it consists in a dense dataset.

For all datasets except Animals, the experiments show reasonably high compression for AVG threshold and above as well as high accuracy for *SP-TR* and *TR-SP* algorithms thus can be proven an effective choice. In general terms, *DP* and *TR* exhibit the worst results in the sense that present the smaller accuracy for all the classification algorithms except for GeoLife dataset and Zheng algorithm in Hurricane dataset, as illustrated in Table 5. However, as the results show in Section 5.4.1, above algorithms present the highest compression rates almost in every dataset and threshold value. The compression achieved with *TR* differ only by a small percentage (slightly lower) in comparison with *DP* but exhibits in most of the cases better results than the latter in terms of accuracy. Furthermore, the compression achieved with *TR* present significantly higher values in UFC, Animals and GeoLife datasets in comparison with *SP*, *TR-SP* and *SP-TR* while in most cases the accuracy is slightly behind. The high compression achieved with *DP* is accompanied by worse accuracy. Clearly, obtained results regarding compression, accuracy and similarity depend on the chosen threshold.

6 Conclusion

In this work, we evaluated how the trajectory compression affects the trajectory similarity and classification. In particular, five top-down compression algorithms were compared against six different thresholds. The dynamic determination of threshold values eliminates

the need of arbitrary user-defined thresholds. Instead, in each trajectory is applied a threshold, based on the actual features and peculiarities of the trajectory. The comparison of the compression algorithms was based on the compression ratio achieved and the execution time across four real-world datasets. The examined algorithms are lossy which means that present information loss. Thus, it is important to measure the quality of the compressed data. For this reason, data mining techniques were employed in order to evaluate the capability of extracting meaningful conclusions from the trajectories even after their compression. These techniques include four trajectory classification methods and two trajectory similarity measures.

The results show that there is a trade-off between the compression rate and the quality achieved. *DP* present the highest compression but exhibits the worst results regarding accuracy in most cases. *TR* achieves extremely high compression while in most cases the accuracy presents better results in comparison with *DP*. *SP* is neither in the worst or the best results at least in most cases. For AVG threshold and above *TR_SP* and *SP_TR* present high compression rate as well as high accuracy. As it is shown in the experiments, the choice of threshold is also crucial as the obtained results depend on this. In conclusion, efficiency analysis demonstrated that there is no “best algorithm” and the selection is application-dependent.

Our future plan is to expand the comparison with more compression algorithms that extensively used, such as STTrace, Dead-Reckoning and SQUISH.

Acknowledgements This work was supported by the MASTER and SmartShip Projects through the European Union’s Horizon 2020 research and innovation programme under the Marie-Slodowska Curie grant agreement no 777695 and 823916 respectively. The work reflects only the authors’ view and that the EU Agency is not responsible for any use that may be made of the information it contains.

Furthermore, this research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under the HFRI PhD Fellowship grant (GA. no. 2158).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Langran G (1993) Issues of implementing a spatiotemporal system. *Int J Geogr Inf Sci* 7(4):305–314
2. Potamias M, Patroumpas K, Sellis T (2006) Sampling trajectory streams with spatiotemporal criteria. In: *Scientific and statistical database management, 2006. 18th international conference on*. IEEE, pp 275–284
3. Makris A, Tserpes K, Anagnostopoulos D, Nikolaidou M, de Macedo JAF (2019) Database system comparison based on spatiotemporal functionality. In: *Proceedings of the 23rd international database applications & engineering symposium*, pp 1–7
4. Makris A, Tserpes K, Spiliopoulos G, Zissis D, Anagnostopoulos D (2020) MongoDB vs postgresql: A comparative study on performance aspects. In: *GeoInformatica*, pp 1–26
5. Makris A, Tserpes K, Spiliopoulos G, Anagnostopoulos D (2019) Performance evaluation of mongodb and postgresql for spatio-temporal data. In: *EDBT/ICDT Workshops*
6. Leptoukh G (2005) Nasa remote sensing data in earth sciences: Processing, archiving, distribution, applications at the ges disc. In: *Proc. of the 31st Intl Symposium of Remote Sensing of Environment*

7. Michael Prior-Jones (2008) *Satellite communications systems buyers' guide*. British Antarctic Survey, Cambridge
8. Muckell J, Olsen PW, Hwang J-H, Lawson CT, Ravi SS (2014) Compression of trajectory data: A comprehensive evaluation and new approach. *GeoInformatica* 18(3):435–460
9. Muckell J, Hwang J-H, Patil V, Lawson CT, Ping F, Ravi SS (2011) Squish: An online approach for gps trajectory compression. In: *Proceedings of the 2nd international conference on computing for geospatial research & applications*. ACM, p 13
10. Meratnia N, Rolf A (2004) Spatiotemporal compression techniques for moving point objects. In: *International conference on extending database technology*. Springer, pp 765–782
11. Pelekis N, Theodoridis Y (2014) *Mobility data management and exploration*. Springer, New York
12. Sun P, Xia S, Yuan G, Li D (2016) An overview of moving object trajectory compression algorithms. *Math Probl Eng* 2016
13. Zheng Y, Capra L, Wolfson O, Yang H (2014) Urban computing: Concepts, methodologies, and applications. *ACM Trans Intell Syst Technol (TIST)* 5(3):38
14. Muckell J, Hwang J-H, Lawson CT, Ravi SS (2010) Algorithms for compressing gps trajectory data: An empirical evaluation. *ACM*
15. Chen M, Xu M, Franti P (2012) Compression of gps trajectories. In: *2012 data compression conference*. IEEE, pp 62–71
16. Frentzos E, Theodoridis Y (2007) On the effect of trajectory compression in spatiotemporal querying. In: *East european conference on advances in databases and information systems*. Springer, pp 217–233
17. Birnbaum J, Meng H-C, Hwang J-H, Catherine L (2013) Similarity-based compression of gps trajectory data. In: *2013 fourth international conference on computing for geospatial research and application*. IEEE, pp 92–95
18. Cudre-Mauroux P, Wu E, Madden S (2010) Trajstore: An adaptive storage system for very large trajectory data sets. In: *2010 IEEE 26th international conference on data engineering (ICDE 2010)*. IEEE, pp 109–120
19. Leichsenring YE, Baldo F (2019) An evaluation of compression algorithms applied to moving object trajectories. *Int J Geogr Inf Sci* 1–20
20. Zhao Y, Shang S, Wang Y, Zheng B, Nguyen QVH, Zheng K (2018) Rest: A reference-based framework for spatio-temporal trajectory compression. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp 2797–2806
21. Zheng K, Zhao Y, Lian D, Zheng B, Liu G, Zhou X (2019) Reference-based framework for spatio-temporal trajectory compression and query processing. *IEEE Trans Knowl Data Eng* 32(11):2227–2240
22. Schmid F, Richter K-F, Laube P (2009) Semantic trajectory compression. In: *International symposium on spatial and temporal databases*. Springer, pp 411–416
23. Lee J-G, Han J, Li X, Gonzalez H (2008) Traiclass: Trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment* 1(1):1081–1094
24. Leite da Silva C, Petry LM, Bogorny V (2019) A survey and comparison of trajectory classification methods. In: *2019 8th brazilian conference on intelligent systems (BRACIS)*. IEEE, pp 788–793
25. Bolbol A, Cheng T, Tsapakis I, Haworth J (2012) Inferring hybrid transportation modes from sparse gps data using a moving window svm classification. *Comput Environ Urban Syst* 36(6):526–537
26. Soleymani A, Cachat J, Robinson K, Dodge S, Kalueff A, Weibel R (2014) Integrating cross-scale analysis in the spatial and temporal domains for classification of behavioral movement. *J Spatial Inform Sci* 2014(8):1–25
27. Dabiri S, Heaslip K (2018) Inferring transportation modes from gps trajectories using a convolutional neural network. *Transport Res Part C Emerg Technol* 86:360–371
28. Zheng Y, Li Q, Chen Y, Xie X, Ma W-Y (2008) Understanding mobility based on gps data. In: *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, pp 312–321
29. Sharma LK, Vyas OP, Schieder S, Akasapu AK (2010) Nearest neighbour classification for trajectory data. In: *International conference on advances in information and communication technologies*. Springer, pp 180–185
30. Soares AJr, Renso C, Matwin S (2017) Analytic: An active learning system for trajectory classification. *IEEE Comput Graphics Appl* 37(5):28–39
31. Dodge S, Weibel R, Forootan E (2009) Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Comput Environ Urban Syst* 33(6):419–434
32. Xiao Z, Wang Y, Kun F, Fan W (2017) Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS Int J Geo-Inform* 6(2):57
33. Etemad M, Soares AJr, Matwin S (2018) Predicting transportation modes of gps trajectories using feature engineering and noise removal. In: *Advances in artificial intelligence: 31st canadian conference*

- on artificial intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31. Springer, pp 259–264
34. Ferrero CA, Alvares LO, Zalewski W, Bogorny V (2018) Movelets: Exploring relevant subtrajectories for robust trajectory classification. 9–13
 35. Han S, Liu S, Zheng B, Zhou X, Zheng K (2020) A survey of trajectory distance measures and performance evaluation. *VLDB J* 29(1):3–32
 36. Vlachos M, Kollios G (2002) Discovering similar multidimensional trajectories. In: Proceedings 18th international conference on data engineering. IEEE, pp 673–684
 37. Chen L, Tamer ÖM, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp 491–502
 38. Kang H-Y, Kim J-S, Li K-J (2009) Similarity measures for trajectory of moving objects in cellular space. In: Proceedings of the ACM symposium on applied computing. ACM, p 2009
 39. Furtado AS, Alvares LOC, Pelekis N, Theodoridis Y, Bogorny V (2018) Unveiling movement uncertainty for robust trajectory similarity analysis. *Int J Geogr Inf Sci* 32(1):140–168
 40. Furtado AS, Kopanaki D, Alvares LO, Bogorny V (2016) Multidimensional similarity measuring for semantic trajectories. *Trans GIS* 20(2):280–298
 41. Lehmann AL, Alvares LO, Bogorny V (2019) Smsm: A similarity measure for trajectory stops and moves. *Int J Geogr Inf Sci* 33(9):1847–1872
 42. Petry LM, Ferrero CA, Alvares LO, Renso C, Bogorny V (2019) Towards semantic-aware multiple-aspect trajectory similarity measuring. *Trans GIS* 23(5):960–975
 43. Douglas DH, Peucker TK (1973) Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica Int J Geographic Inform Geovisualization* 10(2):112–122
 44. Keogh E, Chu S, Hart D, Michael P (2001) An online algorithm for segmenting time series. In: Proceedings IEEE international conference on data mining. IEEE, p 2001
 45. Zheng Y, Xie X, Ma W-Y et al (2010) Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng Bull* 33(2):32–39
 46. Zheng Y, Zhang L, Xie X, Ma W-Y (2009) Mining interesting locations and travel sequences from gps trajectories. In: Proceedings of the 18th international conference on World Wide Web. ACM, pp 791–800
 47. da Silva CL, Petry LM, Bogorny V (2019) A survey and comparison of trajectory classification methods. In: 8th Brazilian conference on intelligent systems, BRACIS 2019, Salvador, Brazil, October 15–18, 2019. IEEE, pp 788–793

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Antonios Makris is currently a Ph.D. candidate at the Department of Informatics and Telematics of Harokopio University of Athens. He received his BSc from the same department as well as his MSc in the area of Web Engineering. His main research interests include distributed computing, big data management and analysis, NoSQL database systems, high performance computing (HPC) and spatiotemporal analysis. He has participated in numerous EU funded and national projects including Fortissimo (FP7), Asylum (Ministry of Immigration Policy), Productive 4.0 (ECSEL), MASTER and SmartShip (H2020).



Camila Leite da Silva is a researcher at the Big Data Analytics Laboratory at Universidade Federal de Santa Catarina (UFSC), Brazil, with focus on trajectory data classification. She received her M.Sc. degree (2020) in Computer Science in the field of Artificial Intelligence from UFSC. Her B.S. degree (2018) is in Information Systems from Universidade Federal de Mato Grosso do Sul (UFMS), where she was part of interdisciplinary projects. Her main areas of interest are machine learning, data analysis and trajectory data.



Vania Bogorny works at the Department of Informatics and Statistics of Federal University of Santa Catarina since July 2009, and is currently the head of the Graduate Program in Computer Science at UFSC. She holds a Ph.D. (2006) and a master's degree (2001) in Computer Science from Universidade Federal do Rio Grande do Sul. In 2007, she received the best Ph.D. thesis award from the Brazilian Computer Society. She holds a CNPq Research Productivity Scholarship since 2008 and has been a post-doctoral researcher in 2014 at INRIA Sophia Antipolis, France; in 2008 at II / UFRGS; and in 2007 at the University of Hasselt, Belgium, in the context of the European project GeopKDD, financed by the European Union. She was a visiting researcher funded by the European Union at CNR / Italy (2013 and 2017 - SOBIGDATA.EU project), Ca'Foscari University in Venice / Italy (2013) and at the University of Piraeus / Greece (2015). In 2012, she published a book on her current research topic (Trajectories of Moving Objects). Since 2010, she has worked as a project reviewer for national agencies such as CNPq, CAPES, Espírito Santo Research Support Foundation, FAPEMIG, FAPESP, and international agencies such as Czech Science Foundation, Chile's CONICYT, and Israel's Ministry of Science and Technology. Since 2009, she has been involved in international research projects such as MODAP, SEEK [FP7], and MASTER [H2020], sponsored by the European Union, establishing research partnerships with the CNR of Pisa / Italy, Ca'Foscari University of Venice / Italy, University of Piraeus / Greece, University of Versailles / France, and University of Dalhousie / Canada, promoting research exchange for professors, master's and doctorate students at UFSC. Her scientific production is co-authored by more than 30 foreign co-authors, she reviews articles for more than 20 international journals, and has an h-index equal to 22.



Luis Otavio Alvares is professor at the Department of Informatics and Statistics (Departamento de Inform?tica e Estatística) at Universidade Federal de Santa Catarina, Florianópolis, Brazil. Until 2010 he had been professor at Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil. He received his Ph.D. in Computer Science from Université Joseph Fourier, Grenoble, France, in 1988, and his M.Sc. in Computer Science from PPGC/UFRGS in 1982. He did a post-doctoral stage at Laboratoire LEIBNIZ/IMAG, Grenoble, France, with Yves Demazeau in 1994/1995. He has served as reviewer, organizing, and technical committee member of several conferences and journals. His research interest include trajectory analysis, spatial and spatio-temporal Data Mining, and Multiagent Systems. He advised 24 M.Sc. and 4 Ph.D. thesis.




Jose Antonio Macedo is a Professor of Computer Science in Computer Science Department at the Federal University of Ceará. He holds a Ph.D. (2005) and an MS (2000) in Computer Science from The Pontifical Catholic University of Rio de Janeiro. Dr. Macedo's research is on data management following two threads: large-scale data processing and distribution, and management of trajectory data. He also founded Nex2Me, a company specialized in data analysis products and services. He authored more than 80 papers in peer-reviewed international journal and conferences. He is now the chief scientist on the public security of the State of Ceara in Brazil.



Konstantinos Tserpes is an Assistant Professor at the Department of Informatics and Telematics of the Harokopio University of Athens. He holds a PhD in the area of Distributed Systems from the school of Electrical and Computer Engineering of the National Technical University of Athens (2008). His research interests revolve around distributed systems, software and service engineering, Big Data analytics and social systems. He has been involved in several EU and National funded projects leading research for solving issues related to scalability, interoperability, fault tolerance, and extensibility in application domains such as multimedia, e-governance, post-production, finance, e-health and others. He is a member of the editorial board of Future Generation Computer Systems.

Affiliations

Antonios Makris¹  · **Camila Leite da Silva²** · **Vania Bogorny²** · **Luis Otavio Alvares²** · **Jose Antonio Macedo³** · **Konstantinos Tserpes¹**

Camila Leite da Silva
camila.leite.ls@posgrad.ufsc.br

Vania Bogorny
vania.bogorny@ufsc.br

Luis Otavio Alvares
luis.alvares@ufsc.br

Jose Antonio Macedo
jose.macedo@insightlab.ufc.br

Konstantinos Tserpes
tserpes@hua.gr

¹ Department of Informatics, Telematics, Harokopio University of Athens, Athens, Greece

² Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil

³ Departamento de Computação, Federal University of Ceará, Fortaleza, Brazil