Received August 14, 2020, accepted August 31, 2020, date of publication September 10, 2020, date of current version September 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3023376

A Framework for Spatial-Temporal Trajectory Cluster Analysis Based on Dynamic Relationships

IVENS PORTUGAL, PAULO ALENCAR, (Member, IEEE), AND DONALD COWAN, (Life Member, IEEE)
School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Corresponding author: Ivens Portugal (iportugal@uwaterloo.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Ontario Research Fund of the Ontario Ministry of Research, Innovation, and Science.

ABSTRACT In spatial-temporal data analysis, location data and its evolution through time are investigated with the goal of uncovering important information to provide novel insights. These insights, for example, may involve congestion identification in transportation, mobility patterns in urban computing, and storm prediction in weather forecasting. Clustering, one data analysis technique, groups spatial-temporal data based on location. Current spatial-temporal data analysis techniques fail to investigate relationships between spatial-temporal clusters, such as splitting from a cluster and merging with another one because of a change of properties over time. These relationships could hold valuable information about the existence of a cluster and its interactions with other clusters and trajectories. In this paper, we introduce a framework to identify, process, and analyze relationships between clusters of spatial-temporal data (e.g. enter, merge, or split). We describe its architecture and components, as well as a proposed clustering technique, the different approaches for distance calculation, and how we calculate cluster similarity of temporally separated clusters. The result of these operations are used in the identification of cluster relationships over space and time. The analysis of these relationships helps uncover hidden values that could support novel approaches to more effective decision-making. We evaluate our framework with two case studies, based on truck and human trajectories.

INDEX TERMS Cluster analysis, spatial-temporal cluster relationships, spatial-temporal data, spatial-temporal data analysis.

I. INTRODUCTION

Spatial-temporal data analysis is the investigation, processing, and visualization of data whose spatial and temporal dimensions are relevant to a specific problem statement, with the goal of uncovering value [1] within the problem context. For example, predicting the demand for ride-sharing services based on spatial-temporal factors, such as the history of orders, tracking of rides through GPS, and the weather, is a spatial-temporal data analysis task of great value to organizations supplying these services and to the general public [2].

Clustering is a data analysis technique that attempts to group data such that any element in a group or cluster has greater similarity to other elements in the same cluster than to elements in other clusters [3]. These techniques are useful for classification, especially when new data becomes available and does not yet belong to any cluster, and anomaly detection, when an element does not appear to belong to any cluster. Clustering techniques have been applied to several domains and different types of data, including social networks [4] and transportation [5], or both [6].

Although clustering of spatial-temporal data has been performed in several domains, clustering remains limited to static representations. For example, a cluster of residences with similar socio-economic status can be regarded as a spatial-temporal cluster, because its location is relevant as well as the socio-economic information changes that occur over time. However, these residences stay in the same location throughout the entire observation, therefore forming a static cluster. In addition, little is known about the relationships that these clusters have with their own elements and with other clusters. For instance, the information that more residences

The associate editor coordinating the review of this manuscript and approving it for publication was Rashid Mehmood.

being classified as belonging to a particular cluster increases the size of that cluster. In addition, two clusters can merge into a larger cluster. These types of information are generally not captured, but may contain critical insights for explaining future phenomena, and thus, hold great value.

We propose a framework for spatial-temporal data analysis that consider moving clusters and that identify and use cluster relationships to generate value. Moving clusters are those whose location changes across time. Some intuitive examples are clusters of people and vehicles, but clusters of animals or even stars are moving clusters too. Cluster relationships are interpretations of the movement between clusters and elements or other clusters, such as enter, or merge. In this paper, we describe the components of our framework, the challenges we faced, and the solutions we found, as well as some future work to be investigated.

This paper is divided as follows. Section II describes some related work needed to understand each component of the framework. Section III describes the framework and provides details about each component and its use. Section IV presents some case studies that use the framework. Section V concludes the paper with a discussion of opportunities for future work.

II. RELATED WORK

A. SPATIAL-TEMPORAL DATA ANALYSIS

Spatial-temporal data analysis is the application of data analysis techniques to spatial-temporal data [1]. The research area of data analysis has existed for many decades but has become particularly interesting more recently because of advances in technology such as IoT that allow large volumes of data to be generated for analyses. There is not a standard classification of data analysis techniques. There is a large number of data analysis technique proposals in the literature, each extending a previously proposed method or adapting the practice to a new domain. In addition, many recently proposed techniques attempt to combine traditional data analysis techniques, making consistent classification more difficult. Despite these obstacles, the work in [3] attempts a classification. Data analysis techniques are generally and broadly classified as classification, association analysis, anomaly detection, and cluster analysis.

1) CLASSIFICATION

There is a large number of techniques whose goal is to classify data often by extending a traditional data analysis technique. Among the many techniques, we include decision trees [7], rule-based classifiers [8], [9], Bayesian classifiers [10], [11], neural networks [12], support vector machines [13], [14], and ensemble methods [15], [16]. Describing each technique is beyond the scope of this paper, but it is worth highlighting the importance of neural networks, especially in the field of deep learning [17], [18], and the improved performance of ensemble methods for classification tasks [19], [20].

2) ASSOCIATION ANALYSIS

Association analysis attempts to uncover interesting relationships between items of a large dataset [21]–[23]. Association analysis is especially useful in businesses such as supermarkets or online markets where association between items being sold are identified and lead to novel marketing strategies or business decisions. Perhaps, the most well-known example of successful use of association analysis techniques is the case of sales of beer and diapers that were discovered to be correlated. In online markets, the well-known recommendation feature ‘‘customers who bought this also bought that’’ is an example of association analysis. The main techniques involve the *Apriori* algorithm [24], and alternative approaches differ in whether an association tree or graph is used, and how it is pruned.

3) ANOMALY DETECTION

Anomaly detection is the task of finding objects, called outliers, that are different from most other objects [25]. There are very important applications of anomaly detection approaches in several domains, including fraud detection [26], [27] and public health [28], [29]. There are different strategies to identify outliers, which can be divided [3] into statistical, proximity-based, and density-based approaches. Statistical approaches for anomaly detection usually build a data model and rely on a probability distribution model to assess how likely data fits this model. Approaches can be based on the traditional Gaussian distribution [30] or on many other distributions, which is the underlying assumption of the Mixture Models approach [31]. Proximity-based approaches for anomaly detection evaluate the distance between data points and consider outliers those distant from most points. As expected, these approaches are heavily based on the choice of distance function. One popular approach is Nearest Neighbors [32]. Finally, density-based approaches attempt to find regions of the data space where data points are close together. Points that do not belong to such a region are deemed outliers. Density-based approaches rely heavily on the definition of distance as well as density. A popular approach is density-based spatial clustering of applications with noise (DBSCAN) [33]. Both proximity- and density-based approaches are forms of cluster analysis, which is described next.

B. CLUSTER ANALYSIS

In this section, we describe cluster analysis, and some related algorithms that are widely-used. We focus the discussion on one specific algorithm, DBSCAN, because of its importance in our approach. Finally, we link the research areas of data analysis and spatial-temporal data with a description of some spatial-temporal clustering approaches for this type of data.

1) OVERVIEW

Cluster analysis refers to the task of grouping data based on its characteristics or relationships [34]–[36]. These groups are called clusters. As a consequence, data objects that are similar to each other belong to the same cluster, whereas objects that have significant differences reside in different clusters. Once clusters are defined, which is called a clustering, classification or anomaly detection tasks can be performed. To classify data, it is assumed that the data has not been used to define a cluster. Classification, then, is the step of querying each cluster and looking for the most appropriate one for the new data. To detect anomalous data in a dataset, one simply observes the result of a cluster approach and looks for data objects that were not associated with any cluster.

Clusterings, the results of a cluster approach, can be of many different types based on their characteristics [3]. In a partitional clustering, the data space is divided into regions that do not overlap, thus each data object is associated with exactly one region. In contrast, hierarchical clustering allows the existence of nested clusters. In hierarchical clustering, clusters are organized in a tree with each non-leaf cluster being the union of all of its subclusters. Clusterings can also be classified based on the degree of membership that data objects have with clusters. In exclusive clusterings, as the name implies, data objects belong to a single cluster, while in overlapping clusterings data objects can belong to more than one cluster. Real-life examples are a university student who is also an instructor, or an employee who is shared between two departments. A third perspective is fuzzy clustering and it assumes that every data object belongs to every cluster! However, each membership has an associated weight or probability. The last clustering classification relates to the completeness of the assignment of data objects to clusters. Complete clusterings have every data object in a cluster, whereas partial clusterings do not. Which one of them is more suitable for anomaly detection?

Clusters themselves can also be classified based on their characteristics [3]. Well-separated clusters have all of their objects closer (or more similar) to each other than to objects of another cluster. Clusters can be prototype-based, graph-based, or density-based. Prototype-based clusters are created based on a prototype, often the centroid. Data objects closer (or similar) to the prototype are assigned to that cluster. Graph-based clusters are modeled as a graph, where the vertices are the data objects and the edges represent connections between objects. Modeling clusters this way allows the use of several graph processing approaches to be applied to cluster analysis. Lastly, density-based clusters are dense regions of data objects surrounded by a region of low-density. The most important characteristic of these clusters is that they can assume any shape. One of the most popular cluster algorithms that produce density-based clusters is DBSCAN, which is described in the next section.

2) SOME APPROACHES

There are many clustering approaches depending on the data available and the type of cluster and clustering. We do not intend to have a description for the entire set of clustering approaches, but we focus on two: K-Means, because of its popularity, and DBSCAN, because of its importance in our research.

K-Means [37], [38] is a prototype-based, partitional clustering technique that tries to find K clusters in the data, where K is a user-defined parameter. The K-Means algorithm is simple to explain and understand, which probably contributes to its wide use. The algorithm starts with a random selection of K centroids. These do not have to be data objects in the dataset, but they often are chosen that way for reason of performance. The algorithm then proceeds to associate each object with the closest centroid. The centroids are then updated based on the points assigned to each cluster. These steps are then repeated until no point changes clusters, or the centroids remain the same. Pseudocode for K-Means algorithm is shown in Algorithm 1.

Algorithm 1: Basic K-Means Algorithm [3]

```
1 Select $K$ points as initial centroids.
2 repeat
3     Form $K$ clusters by assigning each point to its
        closest centroid.
4     Recompute the centroid of each cluster.
5 until Centroids do not change;
```

The basic K-Means algorithm has a complexity of $O(nkdi)$, where n is the number of d-dimensional data objects in the dataset, k is the number of clusters that the user defined, and i is the number of iterations needed until convergence occurs. The algorithm performs well for small datasets, but it can be quite slow when the number of data objects, dimensions, or clusters is high. In addition, K-Means is highly dependent on the distance function being used. Table 1 lists some popular ones.

Table 1 assumes that two data objects a and b are d-dimensional vectors and, thus can written as $a = [a_1, a_2, \ldots, a_d]$ and $b = [b_1, b_2, \ldots, b_d]$. It is also important to notice that the concepts of distance and similarity are inverses.

TABLE 1. Some distance functions.

Name	Formula
Euclidean (L_2)	$d(a,b) = \sqrt{(b_1 - a_1)^2 + \ldots + (b_d - a_d)^2)}$
Manhattan (L_1)	$d(a,b) = (b_1 - a_1) + \ldots + (b_d - a_d)$
Cosine Similarity	$s(a,b) = \frac{a \cdot b}{\lvert\lvert a \rvert\rvert \lvert\lvert b \rvert\rvert} = \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} a_i^2}\sqrt{\sum_{i=1}^{d} b_i^2}}$

DBSCAN [33] stands for density-based spatial clustering of applications with noise. It is a density-based clustering algorithm that attempts to find regions of high density that are separated from each other by regions of low density. The notion of density is key for the execution of the algorithm. In DBSCAN, density is based on the distance of data points. The algorithm uses two parameters in its density calculations: ε and *minPts*. The parameter ε describes the neighborhood of a data point. It can also be thought as denoting the maximum distance allowed for any two objects in a cluster, therefore defining density. The parameter *minPts* gives a floor to the number of points in the neighborhood of a point and helps classifying points as in a dense region or not. DBSCAN describes three types of data points: core, border, and noise. Core points are in the interior of the cluster. They are in a dense region, which means that there are more than *minPts* points at a distance ε from them. Border points are still in the cluster, fall in the neighborhood of a core point but fail to have more than *minPts* points within a distance of ε. A noise point is a point that is not a core or a border point.

Intuitively, DBSCAN execution selects a point and keeps visiting nearby points until the entire region is marked as a cluster. Then it moves to an unmarked point and restarts the process. More precisely, DBSCAN selects an arbitrary point that has not been visited and retrieves a list of points in its neighborhood, that is points within a distance ε from it, using a distance function. If this set of points, including the original point, has enough points (*minPts*), then this represents a dense region. A cluster is started and all points in this set are labeled as belonging to the new cluster. If instead the set of points does not meet the minimum requirement for a dense region (based on *minPts*), the original point is marked as a noise point. Depending on the result of the previous step, the algorithm will either visit other points of the newly discovered region or jump to another arbitrary unvisited point. This process is repeated until all points have been visited. The pseudocode in Algorithm 2 explains DBSCAN.

Algorithm 2: DBSCAN Algorithm [3]

```
1 Label all points as core, border, or noise points.
2 Eliminate noise points.
3 Put an edge between all core points that are within $\varepsilon$ of
    each other.
4 Make each group of connected core points into a
    separated cluster.
5 Assign each border point to one of the clusters of its
    associated core points.
```

In the worst case, DBSCAN has time complexity $O(n^2)$, where n is the number of data points to be clustered. For small datasets, and with the help of data structures, such as kd-trees [39], that allow efficient retrieval of all points within a given distance of a specified point, the time complexity can be reduced to $O(n \log n)$. One important characteristic of DBSCAN is its ability to identify free-form clusters. Many clustering approaches tend to generate globular clusters because they are based on the notion of a radius from centroids. DBSCAN instead uses the distance function from one data point to find others that are in the same cluster and is able to identify clusters in extreme cases, such as two clusters in the form of discs, having the same center, but different radii.

3) SPATIAL-TEMPORAL CLUSTER ANALYSIS

Our discussion so far has been for a generic case. No assumptions about the nature of the data was made. In this section, we focus on the spatial-temporal context. We assume that data has a location that may change with time, therefore spatial-temporal data, and we discuss some data analysis techniques developed specifically for this type of data. We focus on clustering techniques because of their importance to our research.

Spatial-temporal data relates to pieces of information in which both location and time are significant [1]. For example, a person carrying his or her cell phone [40] or a vessel sailing in the sea [41] is generating location data at specific times. Spatial temporal data does not necessarily describe movement, as long as the location and the time are important pieces of information gathered about the phenomena it describes. For example, weather stations are facilities either on land or on the sea where information about the weather, such as temperature, atmospheric pressure, humidity, wind speed, and precipitation amounts, are measured. These stations do not move, but they definitely generate spatial-temporal data [42]. These measurements are reported with the location and the time they were generated. In this paper, we focus on spatial-temporal data that represents movement.

The spatial dimension of the data describes location. For measurements taken on the surface of the Earth, latitude and longitude degrees are often used. Using the conventions of north and south of the planet, divide it in two ‘‘equal’’ (or at least very similar) parts. These are called the hemispheres. The line that runs on the surface of the Earth at the points where the two hemispheres meet is the Equator line. This line represents latitude zero. Now, consider any vector v from the center of the Earth to a point in its surface. The smallest angle between this vector v and any vector v_0 pointing at the Equator line is the latitude. Latitude is the angle ϕ in Figure 1b. Longitude tells us how far to the west or east one location is. It measures the angle between any point on the surface of the Earth and the Prime Meridian. The meridian represents the points separation if Earth was to be divided in equal (or similar) east and west sides. It is a convention, unlike the Equator line, because of the rotation pattern on the Earth. The modern-day meridian, used in GPS devices is the IERS Reference Meridian, maintained by the International Earth Rotation and Reference Systems Service (IERS).[1] Longitude is the angle λ in Figure 1a.

However, the description of spatial data is not limited to the latitude and longitude. Other widely used formats are the addition of altitude, the Cartesian product, thus x and y (and z), polar coordinates, the Z-matrix [43], [44] for atoms, or the celestial coordinate system [45] for celestial objects.

The temporal dimension of spatial-temporal data describes the time. The current research on time describes it as continuous, irreversible, and indefinite. However, some discretization is necessary for computer applications and analyses.

[1]https://www.iers.org/

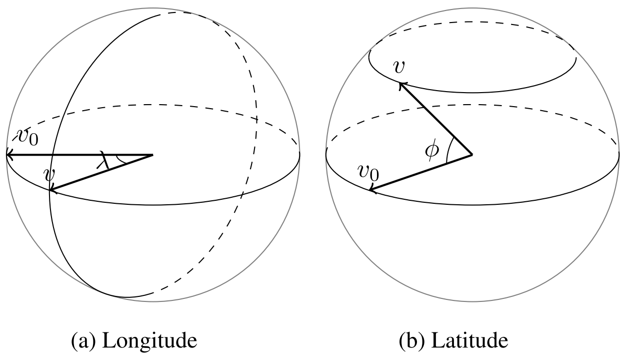

FIGURE 1. Longitude and latitude on the surface of the Earth.

The W3C Consortium[2] maintains an ontology of time.[3] For most analyses, time, an instant, is described in terms of year, month, day, hour, minute, second, and millisecond, although more or less granularity is needed depending on the analysis (e.g. century or millennium for lower granularity or the planck for a theoretical higher granularity). Time can also be described in terms of intervals in two ways, either with a start and end instant, or by calculating their difference. Most spatial-temporal datasets use the term timestamp to denote the discretization of the time at which data was measured.

Spatial-temporal data analysis, as the name implies, refers to the task of analyzing spatial-temporal data looking for hidden patterns that could produce value [1]. For example, by observing people's position at a store, a manager may discover that customers buy products in a certain order. This discovery is valuable because a decision can be taken to optimize (or not) the amount of time customers stay inside the store. There are many analysis techniques aimed at spatial-temporal data [46]–[49], including Spatiotemporal Autoregressive Regression (STAR) for prediction and Spatial-Temporal Density-Based Spatial Clustering of Applications with Noise (ST-DBSCAN) for clustering.

The work in [46] divides the spatial-temporal data analysis domain in several areas based on their goal. These areas are: spatial-temporal outlier, spatial-temporal couplings and tele-couplings, spatial-temporal prediction, spatial-temporal partitioning and summarization, spatial-temporal hotspots, and spatial-temporal change. Note that the visualization of spatial-temporal data [50] is an important step in the data analysis process and can be performed to accomplish the goals described next.

a: SPATIAL-TEMPORAL OUTLIERS

Spatial-temporal outliers are data points whose non-spatial-temporal attributes differs significantly from those in its spatial-temporal neighborhood. It is a local instability or discontinuity, such as an unusually expensive purchase at a particular location or a person who appears to be walking significantly faster than others.

[2]https://www.w3.org/

[3]https://www.w3.org/TR/owl-time/

b: SPATIAL-TEMPORAL COUPLINGS AND TELE-COUPLINGS

Spatial-temporal couplings are patterns of behavior that occur in close geographic and temporal proximity, whereas spatial-temporal tele-couplings relate to behaviors that correlate positively or negatively but at a distance. For example, identifying a sequence of spatial-temporal decisions (visiting places in a certain order) can help understand the underlying reasons for these decisions. There are four approaches: co-occurrence, sequential (''chain reaction''), cascading patterns (sequential and at the same location), or time series (''tele-connection'').

c: SPATIAL-TEMPORAL PREDICTION

Spatial-temporal prediction relates to the task of learning a model and making inferences about attributes (dependent variables) from other attributes (explanatory variables). In case the dependent variables are discrete, the spatial-temporal prediction is called classification; otherwise, in the case that they are continuous, it is called regression. Some application domains are regional climate prediction and real-estate price modeling.

d: SPATIAL-TEMPORAL PARTITIONING AND SUMMARIZATION

Spatial-temporal partitioning and summarization include two related tasks. The first, spatial-temporal partitioning, also known as spatial-temporal clustering refers to arranging data points into groups, thus partitioning the underlying space and time. It has many applications, including demographic analyses and population modeling. Spatial-temporal summarization, done after or together with partitioning, relates to the task of providing a compact representation of spatial-temporal data. For example, the spread of a disease can be summarized with a center and a radius, and the occurrence of traffic jams can be summarized with a list of road segments.

e: SPATIAL-TEMPORAL HOTSPOTS

Spatial-temporal hotspots are regions of the space where the amount of data points is anomalous or excessively high for a certain time interval. Either statistical or clustering techniques are useful to identify those regions. Applications can be found in the domain of public health or public events, by identifying epidemics or sports events.

f: SPATIAL-TEMPORAL CHANGE

Spatial-temporal change refers to change in values of spatial-temporal data points. The task of identifying change is of great importance for uncovering value. There are three types of changes: changes to the statistical parameter, when data has a new mean value; changes to the actual value, when there is a significant difference between values in one data point when compared to its neighbors; and change in models fitted to data, when one of the many models used to explain the data changes. Applications such as quality control or detecting faults in sensor readings require analysis of change.

Our approach focuses on clustering, or partitioning, trajectories, so that relationships between these clusters can be found. There exists two main ways to perform this task. The first way divides time into discrete units (timestamps) and apply density-based clustering approaches on each timestamp. A very common density-based clustering approach used is DBSCAN. The second way reduces all trajectories to sets of spatial-temporal points and uses both dimensions in similarity calculation to find clusters. A popular approach is ST-DBSCAN, which we discuss in the next paragraph.

ST-DBSCAN [51] is a density-based clustering algorithm based on DBSCAN that is able to identify spatial-temporal clusters according to spatial, temporal, and non-spatial-temporal values of objects. Like DBSCAN, the algorithm also identifies core, border, and noise points based on whether they are reachable from core points. Unlike DBSCAN, ST-DBSCAN uses two parameters, ε_1 and ε_2, for spatial and temporal similarity calculation. Therefore, if two data points are within a distance of ε_1 from each other according to some distance function, and the difference between their timestamp values (temporal distance) are within a range of ε_2, these two data points are in the same neighborhood and likely to stay in the same cluster. The parameter *minPts* regulates the required density to form a cluster. ST-DBSCAN also includes a new parameter, *density factor*, to solve DBSCAN's difficulties with clusters with different densities. Its calculation is straightforward. Every data point *p* has the maximum and minimum distance to another point in its neighborhood. Call these distances *density_distance_max(p)* and *density_distance_min(p)*. Thus, the *density_distance* of an object *p* is defined as $density\_distance\_max(p)/density\_distance\_min(p)$. Notice that if a cluster is dense, a small *density_distance_min(p)* increases the final *density_distance* value, whereas points in a low-density cluster have greater *density_distance_min(p)* values and therefore a small *density_distance* value. The new parameter *density factor* of a cluster *C* is defined as:

$$density\_factor(C) = 1/\left[\frac{\sum_{p\in C} density\_distance(p)}{|C|}\right],$$

where $|C|$ is the number of elements in cluster C. Dense clusters will have *density_factor* values that tend to zero, whereas low-density clusters will have *density_factor* values close to one.

Although ST-DBSCAN is widely used in spatial-temporal clustering tasks, it requires significant additional processing to identify relationships between clusters, as we intend in our approach. ST-DBSCAN marks each spatial-temporal point with the cluster to which it belongs. Since no distinction of timestamp or no concept of trajectory is required, the result set has to be reprocessed if we are to identify, for instance, a cluster that enters another. The difficulty in identifying cluster relationships from results led us to choose the original DBSCAN algorithm and the trajectory clustering approach based on timestamps for our work.

C. SPATIAL-TEMPORAL RELATIONSHIPS

Our approach identifies spatial-temporal cluster relationships to uncover properties that were hidden. Here we review the literature of spatial and temporal relationships, as well as some previous work on spatial-temporal cluster relationships. We also give some insights on how they can be extended.

Spatial relationships describe behavior that two elements have towards each other based on their location. A simple example is "a person *enters* a room". The literature describes three basic types of spatial relationships: topological, directional, and distance.

1) TOPOLOGICAL

Topological relationships are based on the way elements are arranged in the space. The Dimensionally Extended Nine-Intersection Model (DE-9IM) [52]–[55] is a standard that describes relationships between two elements in two dimensions. Elements are divided into three regions, interior, boundary, or exterior, and the intersection between the regions of both elements are mapped in a matrix. Different patterns of these matrices define different spatial relationships. There is a total of eight relationships: *equals*, *disjoint*, *touches*, *contains*, *covers*, *intersects*, *within*, *coveredBy*.

2) DIRECTIONAL

Directional relationships describe behavior properties by using one of the elements as a reference. These relationships are divided into two types: internal and external. Internal directional relationships assume that one of the elements is located inside the other. Some examples are *right*, *on the back*, and *athwart*. External directional relationships, on the other hand, assumes that the element is outside of the reference element. Some examples are *on the right of*, *behind*, and *in front of*.

3) DISTANCE

Distance relationships specify how far one element is from another reference element. Identifying these relationships usually involve a distance function. Some intuitive examples are *at*, *nearby*, *in the vicinity*, and *far away*.

Temporal relationships characterize behavior properties between two time periods. A simple example is "we had lunch *before* the meeting". The W3C's Time Ontology[4] document describes 13 elementary relationships for time periods, be it an instant or a time interval. The relationships are *before*, *after*, *meets*, *metBy*, *overlaps*, *overlappedBy*, *starts*, *startedBy*, *during*, *contains*, *finishes*, *finishedBy*, and *equals*.

Spatial-temporal relationships describe interactions between spatial-temporal elements in which both the spatial and the temporal dimensions are considered. There is not a

[4]https://www.w3.org/TR/owl-time/

standard systematic way to derive all relationships, but an intuitive one is to put together a spatial relationship and a temporal relationship [56], [57]. One example is *move to the right*, when the element is *on the left of* the reference element *before* some time t, and is *on the right of* the reference element *after* time t. Another approach to derive spatial-temporal relationships is to observe how they are used in natural language. The work in [58] describes seven relationships: *stopping*, *approaching*, *moving-away*, *arriving*, *leaving*, *passing*, and *jointly-moving*. The papers in [59], [60] observe motion verbs used to describe situations on videos and derive an interesting set of relationships that includes *go to*, *arrive*, *go into*, *depart*, and *leave*, and relationships derived from combinations of those, as in *enter*, *go out*, *come and go*, *go through*, and *go and come*.

Although these relationships are rich and capture several spatial-temporal behavior patterns between spatial-temporal elements, only a subset of them can be used in our approach. The first reason is that some of the relationships assume that two different clusters are well separated even if at a short distance. For example, the relationships *arriving* or *arrive* assumes that two objects are very close. A density-based clustering algorithm will consider the two close clusters to be the same cluster, since they are about to share data points. The second reason is that some relationships require a comparison between the entire space of clusters to be identified, and if feasible, may still not be meaningful. For example, the relationships *moving-away* and *leave* (as described by this author) requires a comparison between two clusters. Clusters that are significantly distant, such as clusters of taxis on opposite sides of a big city would be classified as either *approaching*, *arrive*, *moving-away*, or *leave* multiple times and they may not be related in the first place. Other relationships such as *enter* and *go out* (as described by the author) are relevant to our approach and are considered in our framework.

The paper in [61] lists spatial-temporal relationships that describe interactions among clusters of trajectories. The authors divide relationships into two types: primary, those that can be identified from raw data, and secondary, those derived from primary relationships. The relationships are shown in Table 2 alongside a brief description. In fact, this study has similarities with our approach, and thus a comparison is presented. This comparison is summarized in Table 3. The work in [61] investigates ways to identify groups of objects that travel together. The study defines a *group* as a cluster of at least m density-connected objects for at least a specified *duration* of time. The concept of density-connectedness is derived from density-based cluster algorithms, in which objects in a cluster can be reached from any object in the same cluster through a path of objects of the cluster such that the distance from any two consecutive objects in this path does not exceed ε. This definition is made to differentiate these clusters from other clusters, such as flocks [62], [63], convoys [64], and swarms [65]. The eight cluster relationships described in the study are used to build a tree of object movement, and this tree is analyzed for group discovery. The authors propose a framework for group identification called GroupFramework that is able to identify clusters in streaming data. They also introduce a cluster similarity metric, shown in Eq. (1), that assists in differentiating the clusters, when the returned clusters are very similar to each other. Note that this metric does not take into consideration the timestamps of the objects in the cluster.

TABLE 2. Spatial-Temporal relationships defined in [61].

Name	Description
Appear	An object appears in the system.
Disappear	An object disappears from the system.
Update	An object update its velocity.
Exit	An object exits from a cluster.
Join	An object joins a cluster.
Expire	A cluster expires.
Merge	Multiple (at least two) clusters merge.
Split	One cluster splits into multiple clusters.

TABLE 3. Comparison between the approach in [61] and our approach.

Property	GroupDiscovery	Our Approach
Goal	To identify groups.	To identify cluster relationships to describe cluster lifetime.
Analysis Type	Online	Batch
relationships	Eight	Fourteen
relationships Identification	Based on the dynamics of core and border objects at several timestamps	Based on the underlying density-based algorithm (e.g DBSCAN) at each timestamp.
Cluster Similarity Function	$\frac{\lvert\lvert C_1 \cap C_2 \rvert\rvert}{\lvert\lvert C_1 \cup C_2 \rvert\rvert}$	$\frac{\lvert\lvert C_1 \rvert\rvert}{\lvert\lvert C_1^* \cup C_2^* \rvert\rvert}$ and $\frac{\lvert\lvert C_2 \rvert\rvert}{\lvert\lvert C_1^* \cup C_2^* \rvert\rvert}$

$$Sim(C_1, C_2) = \frac{||C_1 \cap C_2||}{||C_1 \cup C2||}. \tag{1}$$

We instead attempt to describe a cluster lifetime as a sequence of cluster relationships so that interesting patterns can be identified and novel analyses can occur. Regarding the nature of the data, we focus on batch analysis. As discussed in the next section, we detail each relationship and introduce novel ones, totaling a description of fourteen spatial-temporal cluster relationships [66]. These relationships are identified based on application of a density-based algorithm (DBSCAN) at each timestamp and comparing the resulting set of clusters. To perform this comparison, we propose an improved cluster similarity metric, which will be discussed in the next section.

III. FRAMEWORK

A. OVERVIEW

In this section, we introduce our framework for spatial-temporal cluster analysis. We discuss each one of its modules, the challenges we faced, and opportunities for improvement.

An architectural view of the framework, depicting each one of its six different modules is shown in Figure 2. First, the raw spatial-temporal data is preprocessed, such that it is in a standard format. After preprocessing, the controller

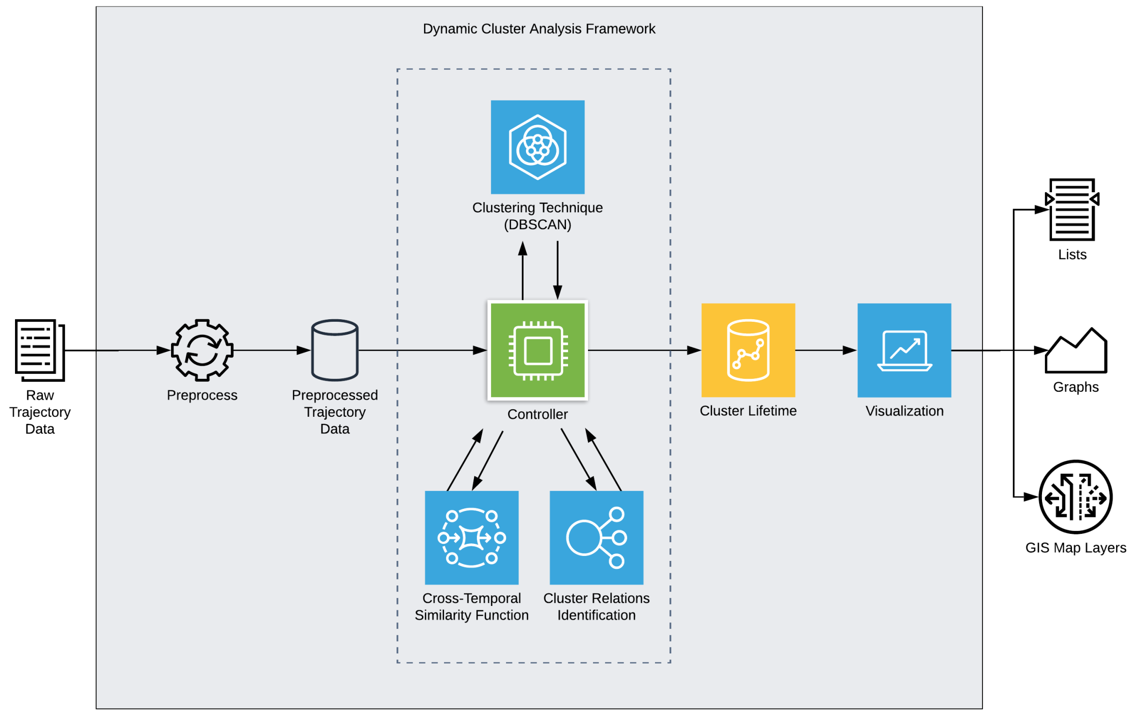
Dynamic Cluster Analysis Framework
Clustering Technique (DBSCAN)
Raw Trajectory Data
Preprocess
Preprocessed Trajectory Data
Controller
Cluster Lifetime
Visualization
Lists
Graphs
GIS Map Layers
Cross-Temporal Similarity Function
Cluster Relations Identification

FIGURE 2. An architecture view of our framework for cluster analysis.

module coordinates execution of the modules. The contoller module executes a clustering technique (DBSCAN) at each timestamp to identify spatial-temporal clusters among the available data. As results become available, the controller applies a cross-temporal similarity function to identify clusters that exist across consecutive timestamps. For example, if the clustering technique identified clusters c_1 and c_2 during timestamp t_i and clusters c_3 and c_4 during timestamp t_j, where $i < j$, this function attempts to measure whether c_3 is the previous c_1, c_2, or a totally new cluster. Once cross-temporal clusters are mapped, the controller requests the cluster relationships identification module to list relationships between these clusters, such as merge or split. Once this information is also available, the controller then stores each relationship for its corresponding cluster for further analysis. We call the set of relationships of a cluster its cluster lifetime. The visualization module translates the raw result data into an intuitive format for analysis.

B. PREPROCESS

Spatial-temporal trajectory datasets follow different formats. Table 4 describes several inconsistencies that can be found. For that reason, we preprocess each new dataset, so that data points of each trajectory follow the format `<id, timestamp, latitude, longitude>`. In our approach, `id` is a number that uniquely identifies a trajectory, `timestamp` is the discretized time when the data was measured following the format `yyyy-mm-dd hh-mm-ss`, and `latitude` and `longitude` are spatial coordinates to describe a place on Earth. One example of a data point of a trajectory is `<246,2019-11-12 06:48:42,-78.366667,165.016667>`. Once all data is standardized, it is then stored for cluster analysis.

TABLE 4. Some inconsistencies found in spatial-temporal datasets.

Type	Description
Non-spatial-temporal data	Presence of data such as brightness, name, color, etc.
Absence of ID	Trajectories should have a unique identifier.
Order of parameters	Data is described as (longitude, latitude), instead of (latitude, longitude).
Missing data	Data may be missing for some timestamps in some trajectories.
Consolidated data	Data about several different trajectories described in the same file.
Timestamp format	Data about time can be described in different formats.

C. CONTROLLER

The controller module coordinates the execution of the framework. In short, it interacts with other modules so that a clustering technique is executed on the data, cross-temporal cluster similarity is calculated, and cluster relationships are identified. Two other secondary, but important features of this module, are the universal timeline function and the coordination of parameters.

Trajectory data is divided into data points describing the location of an object at several different timestamps. Although sensors are reasonably reliable and accurate, two coordination problems still exist: (i) within-trajectory interval irregularity and (ii) between-trajectory interval irregularity. The first problem describes data points that were not captured at regular spaced time intervals. For example, a trajectory *i* can have its first three data points captured at timestamps 10, 15, and 20 seconds, but have its fourth timestamp captured at 26 seconds from the start. The second coordination problem refers to differences of timestamps between two trajectories. A simple example is when trajectory *i* has its data points captured every 5 seconds, while another trajectory *j* has its data points captured every 10 seconds. These two regularity problems led us to adopt two solutions: the universal timeline and the movement assumption. The universal timeline is a function that queries trajectory data files at a user-defined regularly rate regardless of the rate of each individual trajectory. The function keeps track of the current universal time and the time of the most current data point of each trajectory. It then returns new data points only if they exist at the current universal time as shown in Figure 3. Notice that $traj_1$ has regularly spaced intervals, and the most recent data point is used for analysis. Trajectories $traj_2$ and $traj_3$ have, respectively, smaller and larger intervals between data points. In the case of $traj_2$, some data points are never used for analysis, while some data points from $traj_3$ are used more than once. Trajectory $traj_4$ shows the case in which data capture ends, and therefore, no more data points are available for analysis. The movement assumption is used with the universal timeline and states that an object has moved only when new information about its location is available. Therefore, the universal timeline function does not attempt to estimate where an object is when data is not available. Instead it returns the most recent available data.

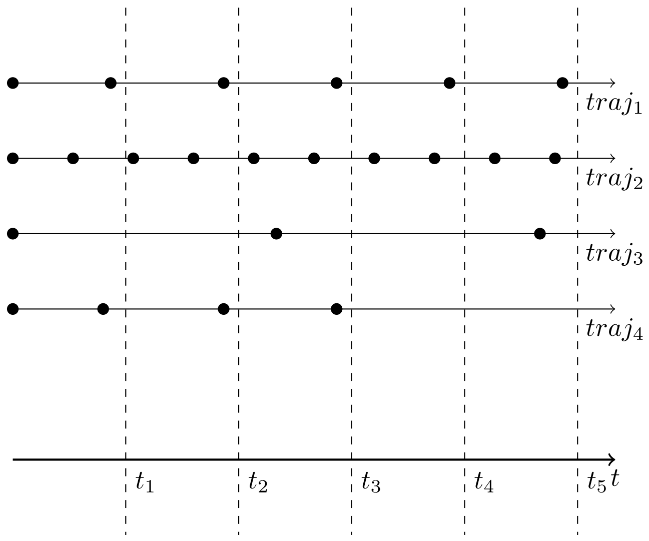

FIGURE 3. The universal timeline. Trajectories have different rates and the most recent one is used in analysis.

The second feature of the controller module is the coordination of parameters. The framework has parameters for the timeline function, the density-based algorithm, and the cross-temporal similarity function. These parameters may be shared between the modules. For example, the clustering technique module needs the *minPts* parameter to calculate whether a group of data points forms a cluster, and the cross-temporal similarity function needs the same parameter to investigate whether the number of trajectories that left a cluster is enough to form a new cluster. For that reason, all user-defined parameters are managed by the controller.

D. CLUSTERING TECHNIQUE

As discussed in previous sections, clustering techniques can be categorized into different types. We chose a density-based approach for our framework because of its ability to identify clusters of different shapes. Some spatial-temporal data, such as the spread of a disease, the weather conditions in a region, or the homes serviced by a hospital tend towards a globular shape because they can be summarized by a radius of influence. Trajectory data, on the other hand, does not always follow this shape, because it depends on many factors, such as the road network or driving decisions. We chose DBSCAN for its performance on large volumes of data and its popularity.

One important step of a density-based clustering approach is the distance calculation. In most applications, the Euclidean distance function is efficient and sufficient. However, spatial-temporal data is usually described in terms of latitude and longitude, that is degrees, minutes, and seconds, instead of miles or meters. The difference between two locations may not be a straight line, but part of a circumference. Distance metrics that consider the curvature of the Earth are called geodesic distance metrics, and there are four popular ones: geodesic, Vincenty, great circle, and haversine. They differ based on the Earth model used in calculations, that can be ellipsoidal or spherical.

1) GEODESIC

The Geodesic distance [67] is an accurate distance metric that assumes that Earth is an ellipse, is accurate to round-off and always converges. However, it requires substantially more time to calculate when compared to simpler metrics and may not be suitable for situations where several calculations are required.

2) VINCENTY

The Vincenty distance [68] is a fairly accurate distance metric that also assumes an ellipsoidal model for Earth. It is accurate to 0.2 millimeter which makes it suitable for several applications. However, it may not converge if points are on the opposite sides of the ellipsoid.

3) GREAT CIRCLE

The Great Circle distance assumes that Earth is a sphere and uses a radius of 6371.009 kilometers, as defined by the

International Union of Geodesy and Geophysics (IUGG),[5] in distance calculations. As expected, the results are not as accurate as the previous methods, but it is more efficient when absolute precision is not a requirement.

4) HAVERSINE

The Haversine distance [69], [70] is a formula used by some geodesic distance calculations, such as the Great Circle. It assumes a spherical model of the Earth. The formula, shown in Eq. (2), is more suitable for modern computers, reducing rounding errors for distances of less than a few meters, and performs better on smaller distances.

$$d=2r\left(\sqrt{sin^2\left(\frac{\varphi_1-\varphi_2}{2}\right)+cos(\varphi_1)cos(\varphi_2)sin^2\left(\frac{\lambda_1-\lambda_2}{2}\right)}\right) \tag{2}$$

- d: the distance between points 1 and 2;
- r: the radius of the sphere (Earth);
- φ_1, φ_2: latitude of points 1 and 2 (in radians);
- λ_1, λ_2: longitude of points 1 and 2 (in radians);

In our approach, the distance between two data points will be calculated several times, for each of the multiple timestamps available. Therefore, the chosen distance function should be as efficient as possible. In addition, our spatial-temporal clusters are groups of taxis, animals, or people, in which very large distances are not meaningful. As a consequence, rounding errors present in the Great Circle and Haversine distances do not significantly impact the overall result. We chose the Haversine distance for its popularity in analysis coding libraries and for its optimized implementation in Cython, a C extension for Python.

E. CROSS-TEMPORAL SIMILARITY FUNCTION

The Cross-Temporal Similarity function assesses if two clusters in different timestamps are the same. In general, cross-temporal cluster similarity can be performed in two ways [71], based on shared objects or cluster characteristics. In the first approach, objects that belong to each cluster are compared, and if a sufficiently large number of objects are present in both clusters, the clusters are deemed to be the same. Two popular metrics or this calculation are Fowlkes-Mallows [72] and Jaccard [73]. In the second method, information, possibly statistical, about two clusters are compared, and a similarity value is calculated. Clusters that exceed a certain similarity threshold are deemed the same. Some statistical tests that can be used include T-test [74] and Chi-square [75].

We chose to follow the first approach to calculating cross-temporal cluster similarity because the framework already handles data objects for cluster identification. Producing statistical information about the cluster would require additional processing time that may not be necessary. The Jaccard index is an intuitive similarity metric to compare two clusters. It is the similarity function used in [61] and is calculated as shown in Eq. (3),

$$J(c_1, c_2) = \frac{|c_1 \cap c_2|}{|c_1 \cup c_2|}, \tag{3}$$

where c_1 and c_2 are clusters in consecutive timestamps. One can choose a threshold for a valid similarity, say 75%. However, the Jaccard index does not perform well when cluster objects change in significant ways, which may be the case for some spatial-temporal situations. Figure 4 illustrates one of these cases. The figure describes two timestamps, the previous one on the left, and the current one on the right of the dashed vertical line in the middle. The figure also shows a cluster c_1 containing 13 objects in the previous timestamp, and a cluster c_3 also containing 13 objects in the current timestamp. The cluster in the middle of the figure is an indication that clusters c_1 and c_3 share 10 objects. Note that during the transition between timestamps, two things happen: three elements leave c_1 and do not form a new cluster, and cluster c_2 with three objects joins c_3. The goal is to evaluate whether clusters c_1 and c_3 are the same, and whether c_2 and c_3 are the same. The Jaccard index is calculated as follows:

$$J(c_1, c_3) = \frac{10}{16} = 0.6250 < 0.75$$
$$J(c_2, c_3) = \frac{3}{16} = 0.1875 < 0.75$$

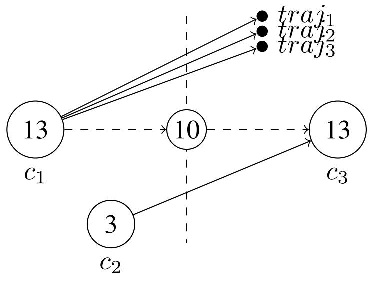

FIGURE 4. A situation where Jaccard fails to identify a cross-temporal cluster, but our approach succeeds.

Thus, the Jaccard index does not consider these clusters the same. However, our intuition indicates that c_1 and c_3 are the same, as they share 10 elements.

Our similarity function also calculates the percent of shared elements but does this for each cluster. Eq. (4) shows our similarity function. Two clusters are deemed similar if the percent of shared elements exceeds a predefined threshold, say 75%, for each of the two clusters.

$$Sim(c_1, c_2) = 1 \text{ if } \begin{cases} \dfrac{|c_1 \cap c_2|}{|c_1|} \geq 0.75, \text{ and} \\ \dfrac{|c_1 \cap c_2|}{|c_2|} \geq 0.75 \end{cases} \tag{4}$$

When using our similarity metric on the situation presented in Figure 4, we obtain the results show in Eqs. (5) and (6). The results follow our intuition and indicate that c_1 and c_3

[5] http://www.iugg.org

TABLE 5. The 14 spatial-temporal cluster relationships and a brief description.

relationship	Description
Start	A cluster starts existing.
End	A cluster stops existing.
Group	Trajectories gather together to form a new cluster.
Disperse	Trajectories follow different paths dismantling a cluster.
T_Enter	A trajectory enters a cluster.
T_Leave	A trajectory leaves a cluster.
C_Enter	A cluster receives another cluster.
C_Leave	A cluster releases another cluster.
Join	A cluster enters another cluster. Notice that this relationship is framed from the perspective of the cluster who enters the other cluster.
Detach	A cluster leaves another cluster. Notice that this relationship is framed from the perspective of the cluster who leaves the other cluster.
Merge	Two or more clusters gather together to form a new cluster.
Split	A cluster breaks apart into two or more clusters.
C_In	A cluster passes trajectories to another cluster, enough to form a new cluster. Framed from the perspective of the cluster receiving trajectories.
C_Out	A cluster passes trajectories to another cluster, enough to form a new cluster. Framed from the perspective of the cluster releasing trajectories.

are indeed the same cluster in different timestamps, but not c_2 and c_3.

$$\text{when comparing } c_1 \text{ and } c_3: \begin{cases} \frac{10}{13} = 0.76 \geq 0.75 \\ \frac{10}{13} = 0.76 \geq 0.75 \end{cases} \tag{5}$$

$$\text{when comparing } c_2 \text{ and } c_3: \begin{cases} \frac{3}{3} = 1 \geq 0.75 \\ \frac{3}{13} \simeq 0.23 < 0.75 \end{cases} \tag{6}$$

In certain extreme situations, a cluster will lose most of its elements and then regain several elements during the transition between timestamps. For example, in Figure 4, cluster c_1 could lose 10 elements and then gain 10 new elements in the next timestamp as c_3. Since the core of the cluster is the same, the three elements that remained, we can still argue that c_1 and c_3 are the same. To account for extreme situations such as this one, we optimize our similarity metric by removing from the calculations trajectories that just entered or left the cluster during the transition from one timestamp to the next. This is limited to the set of trajectories that do not form a new cluster or which came from an existing cluster.

F. CLUSTER RELATIONSHIPS IDENTIFICATION

The Cluster Relationships Identification module receives two set of clusters and trajectories, one for each of the previous and current timestamps, and attempts to identify spatial-temporal relationships between them. For example, if the module detects that a cluster c_1 in the previous timestamp has 10 trajectories, and that five of these trajectories are now in a cluster c_2 and the other five trajectories are in another cluster c_3, and clusters c_2 and c_3 are not cluster c_1, then it concludes that a *split* relationship took place.

We have introduced, formalized, and described 14 spatial-temporal relationships [66], [76], which are presented in Table 5. The two first relationships in the table, *start* and *end*, do not involve a relationship between other clusters, but are necessary to describe the lifetime of a cluster. Cluster relationships are identified based on four pieces of information:

1) Existence in the other timestamp;
2) Number of trajectories the cluster contains in one timestamp, but no cluster contains in the other timestamp;
3) Number of clusters, in the other timestamp, with shared trajectories;
4) Existence of these clusters in this timestamp.

To simplify, we give an example. Let c_1 be a cluster in the current timestamp. The list below describes the tests that are necessary to identify relationships. The first test helps the module assess whether a *start* relationship took place. Note that a cluster may start through different ways (e.g. trajectories that grouped together or detached from a larger cluster). The second test can be used to identify *t_enter* relationships. The third test has many possibilities. It may indicate a *merge*, *detach*, or *group* if the cluster c_1 starts existing at this moment, or *c_enter* or *c_in* if the cluster c_1 already existed in the previous timestamp. The difference between the cases relates to the number of clusters with which c_1 has shared objects. The fourth test helps the module distinguish the relationships of *c_enter* and *c_in*. Note that in this example, the cluster c_1 is in the current timestamp and comparisons are being made with the clusters and trajectories that existed in the previous timestamp.

1) Has the cross-temporal similarity function identified a cluster c_0 in the previous timestamp such that c_0 and c_1 are the same?
2) How many trajectories does the clustering technique module assigns to c_1 in the current timestamp, but does not assign to any cluster in the previous timestamp?
3) How many clusters are in the previous timestamp with which c_1 shares trajectories?
4) Assume that cluster c_1 has shared trajectories with cluster c_0 of the previous timestamp. Does the cluster c_0 exist in the current timestamp?

An analogous case of a cluster c_1^* in the previous timestamp compared with the clusters and trajectories of the current timestamp is also performed, and it helps identify the other set of relationships not mentioned in the example.

Once the cluster relationships identification module identifies relationships at each timestamp, it returns these relationships and the corresponding identifier of the clusters that went through these relationships to the controller. The controller then will store each relationship so that a cluster lifetime is constructed. This cluster lifetime is the basis of further analysis, which starts with the visualization module, explained in the next section.

G. VISUALIZATION

The visualization module assists in the translation from the raw, computer-generated information about cluster lifetimes to easy-to-read, intuitive data formats for analysis. After clusters are calculated, their similarities are mapped, and their relationships are identified and stored, a list of several events that happened to the cluster is available for analysis. The visualization module transforms this list into a more intuitive one such as the one in Listing 1, which shows events that an example cluster 1 went through. The module can also produce graphs based on the occurrence of each event in one or many lifetimes.

```
2019-12-09 14:20:00 start(1)
2019-12-09 14:20:00 group([10, 20, 30, 40, 50], 1)
2019-12-09 14:21:00 t_leave(20, 0, 1)
2019-12-09 14:21:00 t_leave(30, 0, 1)
2019-12-09 14:22:00 t_enter(60, 0, 1)
2019-12-09 14:23:00 disperse([10, 40, 50, 60], 3)
2019-12-09 14:24:00 end(1)
```

LISTING 1. An example lifetime of a cluster.

Geographical Information Systems (GIS) are a framework for managing and analyzing geographical information. They provide a set of analysis algorithms and integrations with map providers so that researchers can visualize their spatial-temporal data and run scripts for analyses. One interesting feature of our visualization module is the ability to translate a cluster lifetime into a GIS-readable format. As a consequence, these relationships can easily be understood on a map.

IV. CASE STUDY

This section presents two case studies developed to evaluate our framework. We show that our framework is able to identify spatial-temporal cluster relationships and that these relationships are important to obtain novel insights about the data. For these case studies, we used datasets of truck and people trajectories, freely available online.

A. CASE STUDY 1

In case study 1, we investigate the relationships that a cluster of trucks can produce, and we show the value brought to spatial-temporal data analysis in general. We used a dataset of truck trajectories.[6] The dataset describes 276 trajectories of 50 trucks delivering concrete to several construction sites around Athens, Greece during 33 distinct days. The size of the dataset is relatively small, just 7.7 MB, which might make it challenging to identify valuable cluster relationships.

We set the DBSCAN parameters ε and *minPts* to 50 meters and 3 trajectories, respectively. The parameter for cross-temporal cluster similarity was set to 0.8. These parameter values were chosen based on previous tests performed using the dataset. A value of $\varepsilon = 10$ means that trucks that are at most 10 meters from each other can be deemed a cluster. This value is small and restrictive considering that GPS readings have a margin of error and that our distance calculation is an approximation because of Earth's curvature. We also noticed that a value of $\varepsilon = 100$ may be large and not ideal, since trucks that are 100 meters apart are fairly distant to form a cluster. A value of $minPts = 2$ would mean that any two trucks that passed by each other is a cluster. This creates a high number of clusters that are not meaningful. On the other hand, we observed that $minPts = 4$ cut the number of clusters in almost half (173 clusters were identified in this case). As a guideline, we aim at avoiding situations in which important cluster relationships are not captured because of restrictive values. Regarding the cross-temporal cluster similarity parameter, we noticed that the largest clusters remained the same even after changing this parameter to any values between 0.6 and 0.9. Therefore, we decided to set it to 0.8. After preprocessing the dataset, execution took less than one minute, and the framework identified 398 clusters.

For analysis of the results, we decided to plot each of the 398 clusters based on the number of relationships it contains. The results can be seen in Figure 5. Notice that one cluster has more than 40 events. That is cluster 41. We then decided to investigate the reasons for this discrepancy further. Listing 2 is an excerpt of the list of events in the lifetime of cluster 41. This cluster has two distinct characteristics: (i) it has many events when compared to other clusters; and (ii) most of the events are *t_enter* and *t_leave* events. There is one distinct place where many trucks would gather together multiple times in a day. To confirm our suspicions, we manually inspected the location of these trucks on a map. The result can be seen in Figure 6. The figure shows an

```
2002-09-10 06:40:39 start(41)
2002-09-10 06:40:39 group([870,873,883,887,901], 41)
2002-09-10 06:40:39 t_leave(873, 0, 41)
2002-09-10 06:40:39 t_leave(887, 0, 41)
2002-09-10 06:41:39 t_leave(870, 0, 41)
2002-09-10 06:42:39 t_enter(873, 0, 41)
2002-09-10 06:42:39 t_enter(887, 0, 41)
2002-09-10 06:42:39 t_enter(906, 0, 41)
2002-09-10 06:43:39 t_enter(898, 0, 41)

30 lines of t_enter or t_leave events

2002-09-10 07:24:39 t_leave(892, 0, 41)
2002-09-10 07:31:39 disperse([864, 883, 891], 41)
2002-09-10 07:31:39 end(41)
```

LISTING 2. The lifetime of cluster 41.

[6] http://chorochronos.datastories.org/?q=node/5

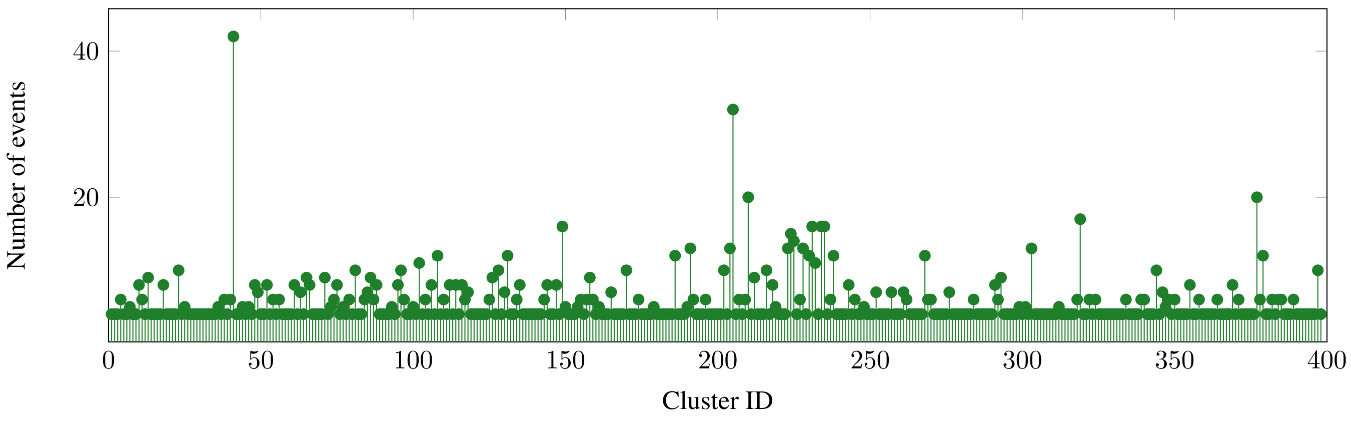

FIGURE 5. The total number of cluster relationships in each cluster of case study 1.

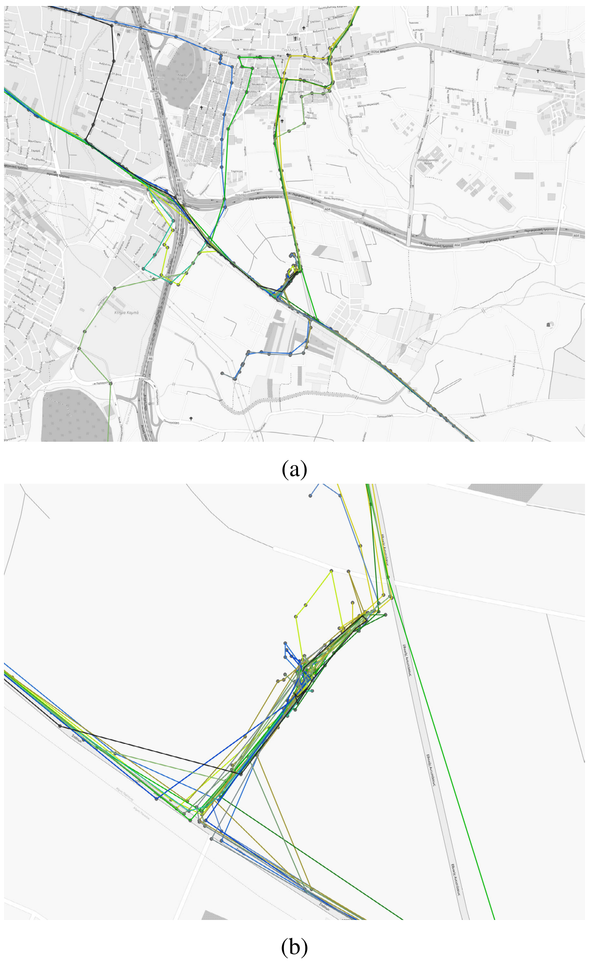

FIGURE 6. Clusters of trucks in the parking lot of a concrete factory.

overview of the location in its upper part. Note the cluster in the bottom middle of Figure 6a. Figure 6b shows a zoomed view of the location, which makes the cluster more evident. The trucks are entering and leaving a cluster that represents the parking lot of a concrete factory. This is a good result, but perhaps there are simpler ways to achieving it. Here is the real value. In another one of our studies [77], with a dataset[7] of Roman taxis, we identified a cluster with the same characteristics. In that study, the cluster turned out to be the parking lot of a taxi company. Some interesting insights start to appear. Do clusters that represent parking lots share similar characteristics? Can a parking lot search be automated? Can we predict or optimize parking lot occupancy levels?

B. CASE STUDY 2

In case study 2, we show how the analysis of spatial-temporal clusters, from many perspectives, can help us to identify hidden insights and opportunities for improvement. We inspect the Geolife dataset[8] collected by Microsoft Research Asia,[9] which describes the trajectories of 182 individuals in a period of over three years. It contains 17,621 trajectories and a total distance of over 1.2 million kilometers. After decompression, the dataset has a size of over 1.5 GB, which demands performance considerations.

For this case study, we also set the DBSCAN's parameter ε to 50 meters, but changed *minPts* to 4, in order to obtain more meaningful clusters. The cross-temporal cluster similarity parameter stayed at 0.8. This choice of parameters is explained as follows. Because this dataset describes the movement of people either on foot or in a vehicle, we need to adjust the parameter ε taking into account both situations. If a person is walking, ε can be lower than in the case in which the person is in a vehicle. However, upon inspection, we noticed that a considerable part of the movements is made using vehicles (e.g. car, taxi). Therefore, we decided to keep $\varepsilon = 50$. The parameter *minPts* was increased to 4 because the framework identified many trivial clusters when the value of *minPts* was lower than 4. This is because we wanted to investigate clusters that had a longer lifetime, with more relationships. We also decided to keep the cross-temporal cluster similarity parameter at 0.8 because our previous case study showed us appropriate results for the cluster similarity.

[7]https://crawdad.org/roma/taxi/20140717/taxicabs/
[8]https://www.microsoft.com/download/details.aspx?id=52367
[9]https://www.microsoft.com/en-us/research/lab/microsoft-research-asia/

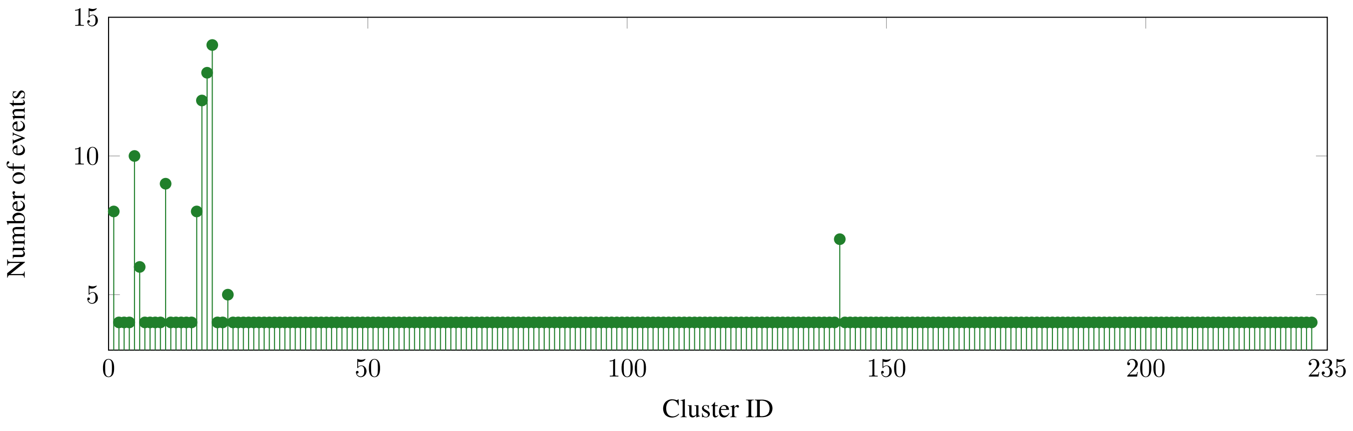

FIGURE 7. The total number of cluster relationships in each cluster of case study 2.

Execution took considerably more time, when compared to the previous case study, but did not exceed 20 minutes. The framework was able to identify 232 spatial-temporal clusters.

Similar to the previous case study, we have plotted in Figure 7 each cluster based on the number of identified relationships. Notice that most clusters have the same number of relationships. These are clusters that do not exist of a long time. However, similarly to case study 1, one cluster contains more relationships than any other cluster. In case study 2, this is cluster 20. Listing 3 shows the list of relationships of the lifetime of cluster 20. Notice how straighforward it is to identify that trajectories 112 and 136 enter and leave the cluster multiple times. We have then decided to also manually inspect the location of the individuals related to these trajectories on a map. Figure 8 contains two parts describing the movement of all trajectories that formed cluster 20 as they move in space. Figure 8a provides an overview of this movement. Notice that, in contrast with Listing 3, visual inspection does not make it clear that trajectories 112 and 136 enter and leave the cluster many times. Figure 8b shows the destination of the trajectories in detail, showing where the cluster 20 ends.

```
2008-05-30 12:38:32 start(20)
2008-05-30 12:38:32 group([73, 78, 112, 128, 153,
167], 20)
2008-05-30 12:43:32 t_enter(136, 0, 20)
2008-05-30 12:43:32 t_leave(136, 0, 20)
2008-05-30 12:44:32 t_leave(112, 0, 20)
2008-05-30 12:46:32 t_enter(112, 0, 20)
2008-05-30 12:46:32 t_leave(112, 0, 20)
2008-05-30 12:48:32 t_enter(112, 0, 20)
2008-05-30 12:48:32 t_enter(136, 0, 20)
2008-05-30 12:50:32 t_leave(78, 0, 20)
2008-05-30 12:50:32 t_leave(167, 0, 20)
2008-05-30 12:52:32 t_leave(136, 0, 20)
2008-05-30 12:59:32 disperse([73, 112, 128, 153],
20)
2008-05-30 12:59:32 end(20)
```

LISTING 3. The lifetime of cluster 20.

To further improve the analysis, we have decided to visualize the lifetime of the clusters as a timeline.

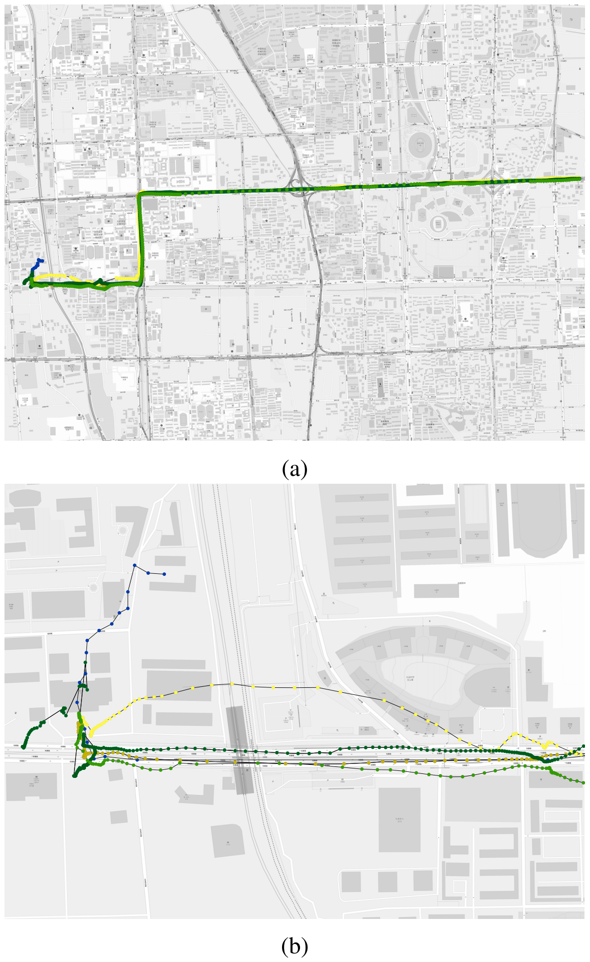

FIGURE 8. Clusters of individuals moving in a city.

Figure 9 illustrates this view. Cluster existence, the time between the *start* and *end* events of a cluster lifetime, is represented by a horizontal line. The horizontal axis is the time, and the vertical one is the cluster id. The number above each horizontal line is the cluster id whose lifetime the line

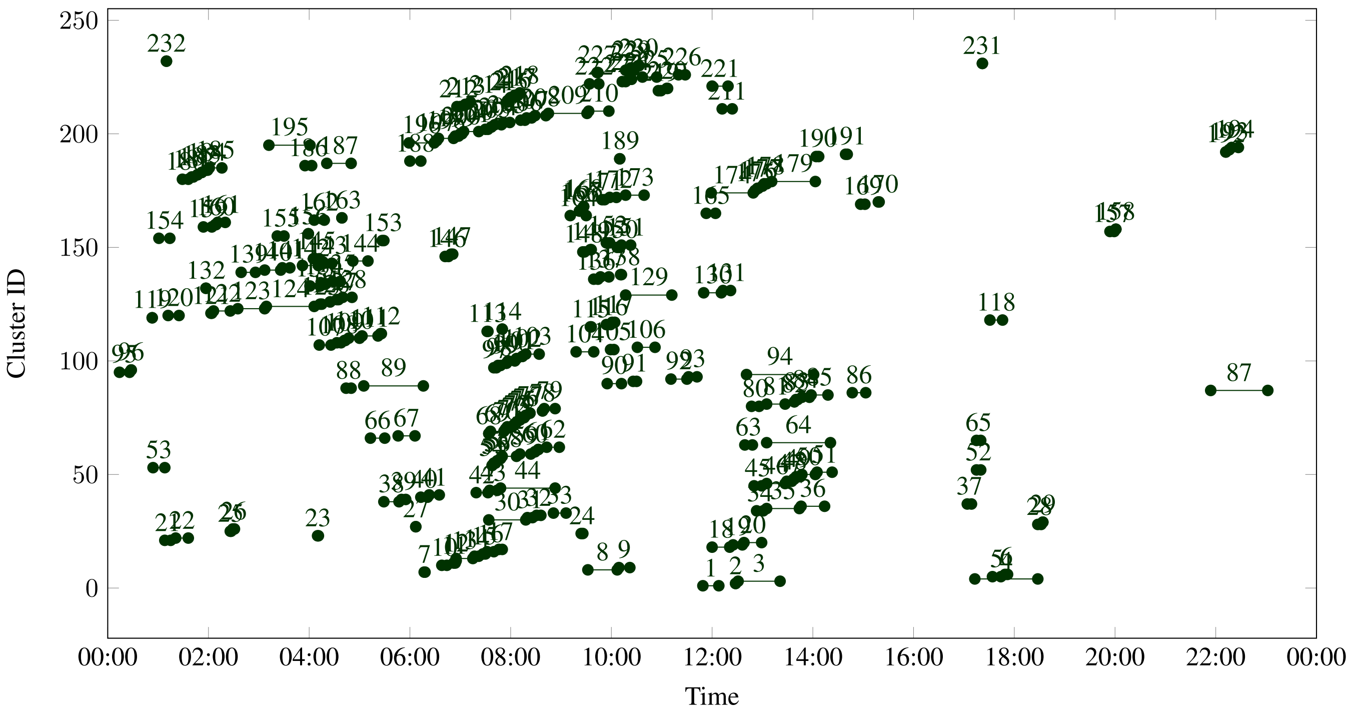

FIGURE 9. A timeline with the lifetime of several clusters.

represents. For example, the line at the right side of the figure represents the lifetime of cluster 87, which started around 22:00 and ended an hour later. In addition, note that these clusters happened on different days, but here they are plotted without this information, as if they all happened on the same day. The time these clusters existed is intact though. This is done to visualize the lifetimes in a meaningful way.

However, based on Figure 9, it is not clear if clusters are related or not. For that reason, we have extracted a few timelines and show them separately in Figures 10 and 11. If we focus on Figure 10, the first thing we should note is that time axis. Figure 10 shows clusters that existed from 12:30pm to 2:30pm, but on different days. One should also notice that these clusters have sequential numbering. Note, for example, clusters 45, 46, 47, 48, 49, 50, and 51. Each one seems to happen right after the other. In fact, this sequential pattern could occur on other days, such as the case of clusters 34, 35, and 36. We are not exactly sure about the event that these clusters represent, but one thing can be said. They seem to start and end at relatively similar timestamps. This is a good indication that these clusters are representing the same event, but in different days, such as people going to have lunch together at a restaurant. In fact, assuming that these fragments are fixed and compose one single cluster, one can use use a modified Jaccard distance function and calculate a degree of coexistence between these clusters. Eq. (7) shows this distance function. Consider the existence of a cluster as a set C_i of points, say one at every minute. The modified Jaccard distance calculates the degree of coexistence between n clusters by calculating the ratio between the number of minutes *all* clusters existed and the number of minutes *any* cluster existed. In Figure 10, all clusters coexist from 12:53 (start time of cluster 34) until 14:01 (end time of cluster 94). In addition, at least one cluster exists from 12:39 (start time of cluster 63) until 14:23 (end time of cluster 51). This represents a degree of coexistence of $\frac{68}{104} \approx 0.65$. Calibration is needed to identify the correct threshold, but a degree of coexistence of approximately 65% is a good indication of similarity.

$$J(C_1, \cdots, C_n) = \frac{|C_1 \cap \cdots \cap C_n|}{|C_1 \cup \cdots \cup C_n|} \tag{7}$$

The hidden value in the plot of Figure 10 is in the broken pattern of, for instance, clusters 45, 46, 47, 48, 49, 50, and 51. This is an indication that a group of trajectories are close enough to form a cluster, contains just the minimum number of trajectories, but frequently loses and regains one trajectory, leading to a cluster end and the start of another cluster. The loss of a trajectory happens perhaps because of the road network or the distance between these trajectories. These broken patterns, in the case of vehicles, such as the one in this dataset, indicate a fragmented movement within the same activity, and an opportunity for combining them for a more cohesive movement. In the case of vehicles, this translates to ride-sharing opportunities, either by having more people in the same car, or by using a van. In addition, notice that the same broken pattern occurs in Figure 11, in which the set of all the timelines have a degree of coexistence of $54/99 \approx 0.54$ or around 54%. Figure 9 may have other

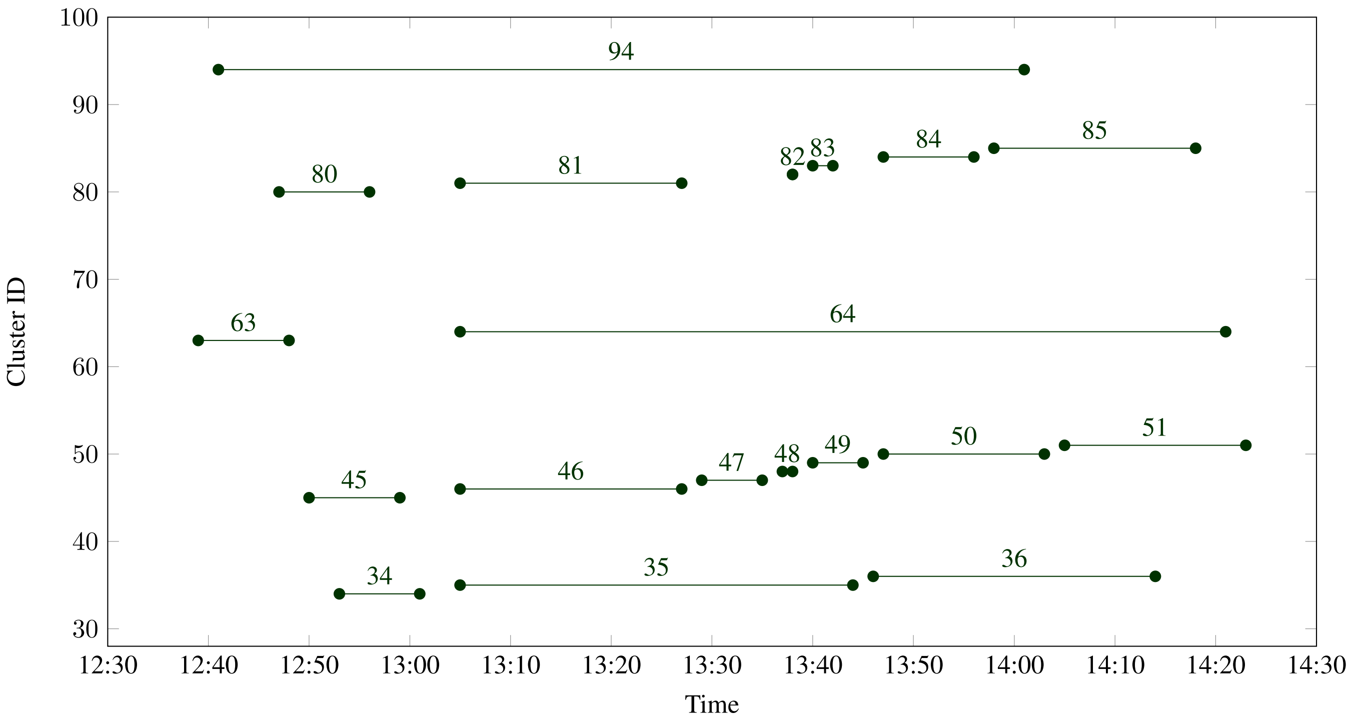

FIGURE 10. A timeline with the lifetime of several clusters.

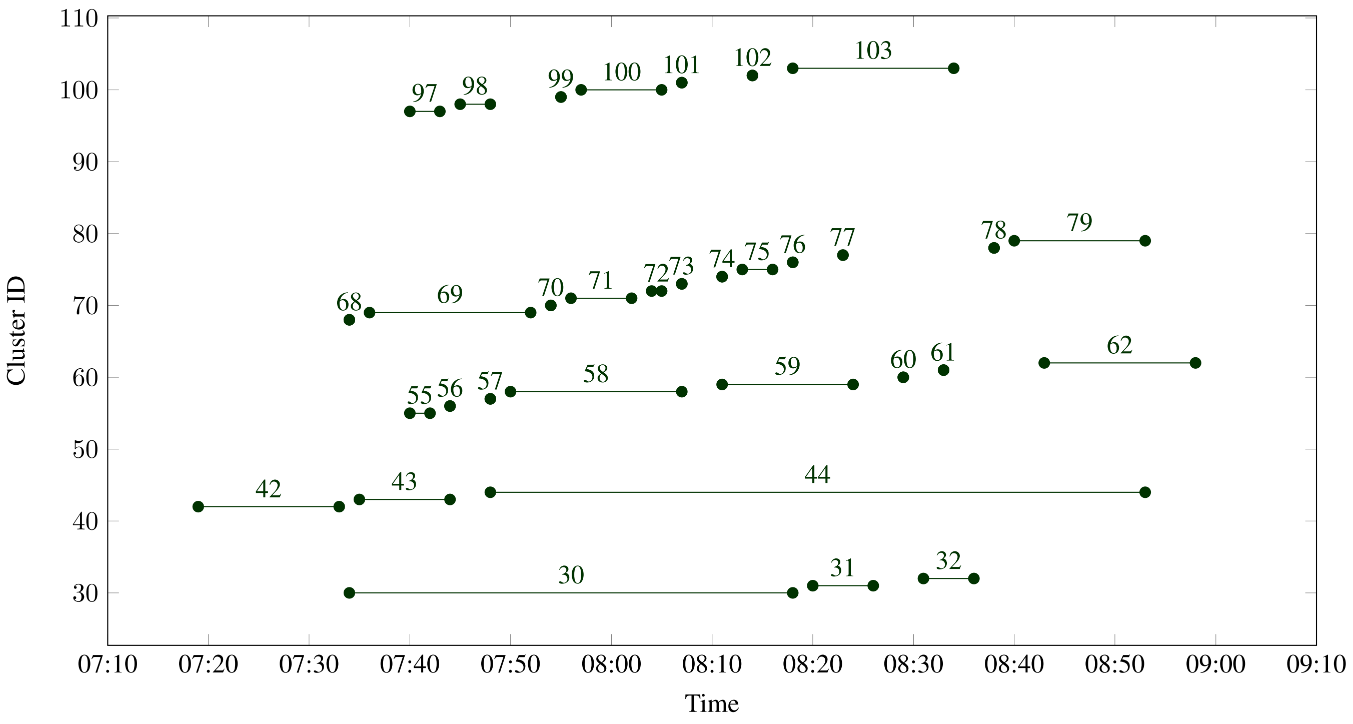

FIGURE 11. Another timeline with the lifetime of several clusters.

examples of the same phenomenon, but these two cases were chosen for being clear to inspect. Insights such as the one discussed here can lead to more cost-effective solutions or better decision-making.

V. CONCLUSION AND FUTURE WORK

Spatial-temporal data analysis is the task of analyzing data whose spatial and temporal information is considered. Clustering approaches group spatial-temporal data for analysis based on these dimensions. However, relationships between these spatial-temporal clusters are not analyzed and valuable information that may lead to novel insights remain hidden. To solve this problem, we developed a framework for spatial-temporal cluster analysis, that identifies, processes, and analyzes cluster relationships, such as *merge* or *split*. We evaluated the framework with two case studies and showed their applicability and the value they can uncover.

In the future, we expect to follow a number of research directions that could improve the current state of the framework. One of these directions is the adaptation of the current computations for a distributed computational environment. The current version of the framework runs efficiently with substantial amounts of data, but as spatial-temporal data becomes more frequently available, a distributed environment is required. The most important challenge is the coordination of a distributed clustering approach, and the relationship identification. In a distributed environment, data remains in different nodes of a computer cluster, and retrieving these pieces of information efficiently is challenging.

Next, we plan to extend the amount of spatial-temporal cluster relationships that can be identified. There are two ways to perform this extension. The first one is to investigate a systematic way to derive novel spatial-temporal cluster relationships. The second way is to identify composite cluster relationships, such as *return*, which can be composed of the relationship *c_leave* followed by a relationship *c_enter*.

In addition, new spatial-temporal dimensions can be included in the calculations. There are two ways that this can be performed. Currently, data points are points that have no extension in any dimension. They may represent vehicles, homes, or people. However, some real-world objects, such as trains and cities do have additional space dimensions that may change the calculations. Therefore, analyzing these objects is the first way to include new spatial-temporal dimensions into the calculations. The second way is to investigate altitude as another spatial dimension, or even generalize the calculations for a theoretical n-dimensional object or world.

Finally, the framework was built with a real-world applicability mindset. This means that we assume a geographical world, and that spatial-temporal data represents real-world objects. However, spatial-temporal data is not limited to the geographical domain. Small and big structures can also be analyzed, such as molecules or planets. Thus, it is interesting to investigate cluster relationships in new domains such as medicine and astronomy.

REFERENCES

[1] G. Eshel, *Spatiotemporal Data Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[2] Y. Wang, X. Lin, H. Wei, T. Wo, Z. Huang, Y. Zhang, and J. Xu, "A unified framework with multi-source data for predicting passenger demands of ride services," *ACM Trans. Knowl. Discovery from Data*, vol. 13, no. 6, pp. 1–24, Dec. 2019.

[3] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*. London, U.K.: Pearson, 2018.

[4] T. M. Sweet, A. Flynt, and D. Choi, "Clustering ensembles of social networks," *Netw. Sci.*, vol. 7, no. 2, pp. 141–159, Jun. 2019.

[5] X. Ma, R. Cao, and Y. Jin, "Spatiotemporal clustering analysis of bicycle sharing system with data mining approach," *Information*, vol. 10, no. 5, p. 163, May 2019.

[6] A. T. Akabane, R. Immich, R. W. Pazzi, E. R. M. Madeira, and L. A. Villas, "Exploiting vehicular social networks and dynamic clustering to enhance urban mobility management," *Sensors*, vol. 19, no. 16, p. 3558, Aug. 2019.

[7] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[8] X.-L. Li and B. Liu, *Rule-Based Classification*. Boca Raton, FL, USA: CRC Press, 2014.

[9] C. Ravi and N. Khare, "Review of fuzzy rule based classification systems," *Res. J. Pharmacy Technol.*, vol. 9, no. 8, pp. 1299–1302, 2016.

[10] J. Bernardo and A. Smith, *Bayesian Theory*. Hoboken, NJ, USA: Wiley, 2008.

[11] P. Congdon, *Applied Bayesian Modelling*. Hoboken, NJ, USA: Wiley, 2003.

[12] G. Dreyfus, *Neural Networks: Methodology and Applications*. Berlin, Germany: Springer, 2005.

[13] N. Deng, Y. Tian, and C. Zhang, *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. Boca Raton, FL, USA: CRC Press, 2012.

[14] Y. Ma and G. Guo, *Support Vector Machines Applications*. Cham, Switzerland: Springer, 2014, Art. no. 9783319023007.

[15] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.

[16] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. New York, NY, USA: Springer, 2012.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[18] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, Dec. 2015.

[19] R. Maclin and D. Opitz, "Empirical evaluation of bagging and boosting," in *Proc. Nat. Conf. Artif. Intell.*, AAAI: AAAI, Menlo Park, CA, USA, 1997, pp. 546–551.

[20] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural network ensembles: Evaluation of aggregation algorithms," *Artif. Intell.*, vol. 163, no. 2, pp. 139–162, Apr. 2005.

[21] M. Hegland, "Algorithms for association rules," in *Advanced Lectures on Machine Learning* (Lecture Notes in Computer Science), vol. 2600. Berlin, Germany: Springer, 2003, pp. 226–234.

[22] M. Zhang and C. He, "Survey on association rules mining algorithms," *Advancing Computing, Communication, Control and Management* (Lecture Notes in Electrical Engineering), vol. 56. Berlin, Germany: Springer, 2010, pp. 111–118.

[23] T.-P. Hong and Y.-C. Lee, "An overview of mining fuzzy association rules," *Stud. Fuzziness Soft Comput.*, vol. 220, pp. 397–410, 2008.

[24] R. Agrawal and S. Ramakrishnan, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases*. San Mateo, CA, USA: Morgan Kaufmann, 1994, pp. 487–499.

[25] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 2009.

[26] M. Anderka, T. Klerx, S. Priesterjahn, and H. Büning, "Automatic atm fraud detection as a sequence-based anomaly detection problem," in *Proc. ICPRAM 3rd Int. Conf. Pattern Recognit. Appl. Methods*. Setúbal Municipality, Portugal: SciTePress, 2014, pp. 759–764.

[27] J. Xu, A. Sung, Q. Liu, and S. Mukkamala, "Fraud detection system based on behavior mining and anomaly detection," in *Proc. 2nd Indian Int. Conf. Artif. Intell., IICAI*, 2005, pp. 3487–3501.

[28] S. Haque, M. Rahman, and S. Aziz, "Sensor anomaly detection in wireless sensor networks for healthcare," *Sensors*, vol. 15, no. 4, pp. 8764–8786, Apr. 2015.

[29] Z. Huang, X. Lu, and H. Duan, "Anomaly detection in clinical processes," in *Proc. AMIA. Annu. Symp. AMIA Symp., AMIA Symp.*, 2012, pp. 370–379.

[30] K. Krishnamoorthy, *Handbook of Statistical Distributions With Applications*. Boca Raton, FL, USA: CRC Press, 2006.

[31] P. McNicholas, *Mixture Model-Based Classification*. Boca Raton, FL, USA: CRC Press, 2016.

[32] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[33] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[34] B. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*, 5th ed. Hoboken, NJ, USA: Wiley, 2011.

[35] C. Hennig, M. Meila, F. Murtagh, and R. Rocci, *Handbook of Cluster Analysis*. Boca Raton, FL, USA: CRC Press, 2015.

[36] D. Wilks, "Cluster analysis," *Int. Geophys.*, vol. 100, pp. 603–616, 2011, doi: 10.1016/B978-0-12-385022-5.00015-4.

[37] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[38] E. W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classifications," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.

[39] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[40] Y. Mao, H. Zhong, H. Qi, P. Ping, and X. Li, "An adaptive trajectory clustering method based on grid and density in mobile pattern analysis," *Sensors*, vol. 17, no. 9, p. 2013, Sep. 2017.

[41] H. Li, J. Liu, K. Wu, Z. Yang, R. W. Liu, and N. Xiong, "Spatio-temporal vessel trajectory clustering based on data mapping and density," *IEEE Access*, vol. 6, pp. 58939–58954, 2018.

[42] Y.-H. Kim and Y. Yoon, "Spatiotemporal pattern networks of heavy rain among automatic weather stations and very-short-term heavy-rain prediction," *Adv. Meteorol.*, vol. 2016, pp. 1–13, Jan. 2016.

[43] J. Parsons, J. B. Holmes, J. M. Rojas, J. Tsai, and C. E. M. Strauss, "Practical conversion from torsion space to Cartesian space forin silico protein synthesis," *J. Comput. Chem.*, vol. 26, no. 10, pp. 1063–1068, 2005.

[44] J. A. Pople and G. A. Segal, "Approximate self-consistent molecular orbital Theory. III. CNDO results for AB2 and AB3 systems," *J. Chem. Phys.*, vol. 44, no. 9, pp. 3289–3296, May 1966.

[45] N. Capitaine and M. Stavinschi, *Celestial Coordinate Systems and Positions*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[46] S. Shekhar, Z. Jiang, R. Ali, E. Eftelioglu, X. Tang, V. Gunturi, and X. Zhou, "Spatiotemporal data mining: A computational perspective," *ISPRS Int. J. Geo-Inf.*, vol. 4, no. 4, pp. 2306–2338, Oct. 2015.

[47] R. R. Vatsavai, A. Ganguly, V. Chandola, A. Stefanidis, S. Klasky, and S. Shekhar, "Spatiotemporal data mining in the era of big spatial data: Algorithms and applications," in *Proc. 1st ACM SIGSPATIAL Int. Workshop Anal. Big Geospatial Data BigSpatial*, 2012, pp. 1–10.

[48] H. Li, J. Liu, Z. Yang, R. W. Liu, K. Wu, and Y. Wan, "Adaptively constrained dynamic time warping for time series classification and clustering," *Inf. Sci.*, vol. 534, pp. 97–116, Sep. 2020.

[49] J. Cuenca-Jara, F. Terroso-Sáenz, M. Valdés-Vela, and A. F. Skarmeta, "Classification of spatio-temporal trajectories from volunteer geographic information through fuzzy rules," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105916.

[50] Y. Huang, Y. Li, Z. Zhang, and R. W. Liu, "GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries," *IEEE Internet Things J.*, early access, Apr. 21, 2020, doi: 10.1109/JIOT.2020.2989398.

[51] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[52] E. Clementini, P. Di Felice, and P. Van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," in *Advances in Spatial Databases* (Lecture Notes in Computer Science), vol. 692. Berlin, Germany: Springer, 1993, pp. 277–295.

[53] E. Clementini, J. Sharma, and M. J. Egenhofer, "Modelling topological spatial relations: Strategies for query processing," *Comput. Graph.*, vol. 18, no. 6, pp. 815–822, Nov. 1994.

[54] M. J. Egenhofer and R. D. Franzosa, "Point-set topological spatial relations," *Int. J. Geograph. Inf. Syst.*, vol. 5, no. 2, pp. 161–174, Jan. 1991.

[55] M. Egenhofer and J. Herring, "A mathematical framework for the definition of topological relationships," in *Proc. 4th Int. Symp. Spatial Data Handling*, 1990, pp. 803–813.

[56] C. Holzmann, "Rule-based reasoning about qualitative spatiotemporal relations," in *Proc. 5th Int. workshop Middleware for Pervas. ad-hoc Comput. held at ACM/IFIP/USENIX 8th Int. Middleware Conf. MPAC*, 2007, pp. 49–54.

[57] X. Chen, W. Li, and L. Yan, "A UML-based representation of fuzzy spatiotemporal relations," in *Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Jul. 2017, pp. 1090–1098.

[58] J. D. Mazimpaka, "A framework for analysing trajectories of movement in a dynamic geographic context," Ph.D. dissertation, Dept. Appl. Comput. Sci., Univ. Augsburg, Augsburg, Germany, 2017.

[59] C. Choi, M. Cho, and P. Kim, "The new modeling for semantic representation of moving objects in video," in *Proc. Int. Conf. Adv. Commun. Technol., ICACT*, vol. 1, 2007, pp. 363–367.

[60] M. Cho, D. Song, C. Choi, J. Choi, J. Park, and P. Kim, "Comparison between motion verbs using similarity measure for the semantic representation of moving object," in *Image and Video Retrieval* (Lecture Notes in Computer Science), vol. 4071. Berlin, Germany: Springer, 2006, pp. 281–290.

[61] X. Li, V. Ceikute, C. S. Jensen, and K.-L. Tan, "Effective online group discovery in trajectory databases," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2752–2766, Dec. 2013.

[62] J. Gudmundsson and M. van Kreveld, "Computing longest duration flocks in trajectory data," in *Proc. 14th Annu. ACM Int. Symp. Adv. Geographic Inf. Syst. GIS*, 2006, pp. 35–42.

[63] J. Gudmundsson, M. van Kreveld, and B. Speckmann, "Efficient detection of motion patterns in Spatio-temporal data sets," in *Proc. 12th Annu. ACM Int. workshop Geographic Inf. Syst. GIS*, 2004, pp. 250–257.

[64] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," *Proc. PVLDB*, vol. 1, no. 1, pp. 1068–1080, 2008.

[65] Z. Li, B. Ding, J. Han, and R. Kays, "Swarm: Mining relaxed temporal moving object clusters," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 723–734, Sep. 2010.

[66] I. Portugal, P. Alencar, and D. Cowan, "Spatial-temporal cluster relations: A foundation for trajectory cluster lifetime analysis," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., 2019.

[67] C. F. F. Karney, "Algorithms for geodesics," *J. Geodesy*, vol. 87, no. 1, pp. 43–55, Jan. 2013.

[68] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Surv. Rev.*, vol. 23, no. 176, pp. 88–93, Apr. 1975.

[69] G. Van Brummelen, *Heavenly Mathematics: The Forgotten Art Spherical Trigonometry*. Princeton, NJ, USA: Princeton Univ. Press, 2012.

[70] J. De Mendoza, "Memoria sobre algunos metodos nuevos de calcular la longitud por las distancias lunares: Y aplicacion de su teórica á la solucion de otros problemas de navegacion," *La Imprenta Real*, 1795.

[71] R. Ramon-Gonen and R. Gelbard, "Cluster evolution analysis: Identification and detection of similar clusters and migration patterns," *Expert Syst. Appl.*, vol. 83, pp. 363–378, Oct. 2017.

[72] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Stat. Assoc.*, vol. 78, no. 383, pp. 553–569, Sep. 1983.

[73] P. Jaccard, "Etude comparative de la distribution florale dans une portion des alpes et des jura," *Bull. de la Société Vaudoise des Sciences Natturelles*, vol. 37, pp. 547–579, 1901.

[74] Student, "The probable error of a mean," *Biometrika*, vol. 6, no. 1, pp. 1–25, Mar. 1908.

[75] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *Philos. Mag. J. Sci.*, vol. 50, no. 302, pp. 157–175, 1900.

[76] I. Portugal, P. Alencar, and D. Cowan, "Modeling dynamic spatial-temporal cluster relationships," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 3590–3598.

[77] I. Portugal, P. Alencar, and D. Cowan, "Computational cluster lifetime analysis to capture cluster lifetime dynamics," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., 2019.

IVENS PORTUGAL, photography and biography not available at the time of publication.

PAULO ALENCAR (Member, IEEE) is currently a Research Professor with the David R. Cheriton School of Computer Science, University of Waterloo, where he is also the Associate Director of the Computer Systems Group (CSG). He has published over 200 refereed publications and has been a member of program committees of numerous highly-regarded conferences and workshops. His recent research in information technology and software engineering has focused on high-level software architectures, design, components and their interfaces, software frameworks and application families, software processes, automated workflows and work graphs, and evolution, Web-based approaches and applications, open and big data applications, context-aware and event-based systems, software agents, machine learning, cognitive chatbots, artificial intelligence, and formal methods. He is a member of the Association of Computing Machinery (ACM), the Association for the Advancement of Artificial Intelligence (AAAI), Waterloo Water Institute (part of the Global Water Futures initiative), and Waterloo Artificial Intelligence Institute. He has received international research awards from organizations, such as Compaq and IBM. He has been a Principal or Co-Principal Investigator in projects supported by NSERC, ORF-RE, IBM, SAP, CITO, CSER, Bell, and funding agencies in Canada, USA, Brazil, Germany, and Argentina.

DONALD COWAN (Life Member, IEEE) is currently a Distinguished Professor Emeritus of computer science with the University of Waterloo, where he is also the Director of the Computer Systems Group. He has made contributions to computer science, such as computer science, software engineering, and complex applications. He is the author/coauthor or editor of over 280 refereed papers and 17 books in computer/communications, software engineering, education, environmental information systems, and mathematics. He has supervised over 120 graduate students and Postdoctoral Fellows. His group has developed over 80 Web-based and mobile software systems for many applications, such as volunteerism, environment, socioeconomic development, tourist, population health, aboriginal affairs, arts and culture, and built heritage. He received the Award of Excellence in Graduate Supervision from the University of Waterloo, for his contributions to the development of graduate students. He was recognized for his research and support of the development of computer science in Brazil by being awarded the National Order of Scientific Merit (Grand Cross), the country's highest civilian scientific honor, by the President of Brazil. In 2009, he received the Waterloo Award, the City of Waterloo's highest civic honor, for his contributions to the City of Waterloo. In 2010, he was named as a Distinguished Scientist by the Association for Computing Machinery, and he received also the Doctor of Science *(Honoris Causa)* from the University of Guelph, in 2011, for his contributions to computer science and software engineering.