Graph-Based Spatial-Temporal Cluster Evolution: Representation, Analysis, and Implementation

by

Ivens da Silva Portugal

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: John Easton
Professor, Dept. of Electronic, Electrical and Systems Engineering, University of Birmingham

Supervisor(s): Paulo Alencar
Professor, Dept. of Computer Science, University of Waterloo
Donald Cowan
Professor, Dept. of Computer Science, University of Waterloo
Daniel Berry
Professor, Dept. of Computer Science, University of Waterloo

Internal Member: Khuzaima Daudjee
Professor, Dept. of Computer Science, University of Waterloo

Internal-External Member: Ladan Tahvildari
Professor, Dept. of Electrical and Computer Engineering, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Spatial-temporal data are information about real-world entities that exist in a location, the spatial dimension, and during a period of time, the temporal dimension. These real-world entities, such as vehicles, people, or parcels and called spatial-temporal objects, may move, group, and continue the movement together, forming clusters. Although there have been significant research efforts to understand clusters, there is a lack of research that provides methods and software tools to support the representation, analysis, and implementation of graph-based spatial-temporal cluster evolution. Understanding this evolution is critical for dealing with spatial-temporal problems encountered in domains, such as service supply and demand, supply chain management, traffic and travel flows, human mobility, and city planning.

This thesis presents an approach to graph-based cluster evolution and its representation, analysis, and implementation. The proposed solution introduces a representation of the structure of a spatial-temporal cluster with the identification of the cluster at several timestamps and linkages, and a representation of 14 spatial-temporal relationships clusters have during their existence. The proposed solution also introduces a graph representation of cluster evolution with nodes acting as clusters and edges as relationships. This solution provides analysis methods for the structure of spatial-temporal clusters that monitor the cluster changes in both location and size over time, and analysis methods for the spatial-temporal cluster relationships the clusters have during existence that calculate the frequency or density of such relationships in specific locations. The solution also provides analysis methods for a graph-based representation of spatial-temporal cluster evolution including integrated results that examine spatial-temporal clusters and their connections, and can provide, for example, aggregated results at a location or time of the day, identify ever-increasing or ever-decreasing regions, growth or decay rates, and measure the similarity between the evolution of two clusters. The approach also provides an implementation of the proposed representation and analysis methods. The effectiveness of the approach is evaluated through four case studies using different spatial-temporal datasets to show the results that can be produced, which include, exploratory analyses and specific analyses on ever-increasing and ever-decreasing regions, similarity values, and the movements the clusters represent. Overall, the proposed approach advances research in the spatial-temporal domain by providing novel representation and analysis methods as well as implementation tools that can improve the understanding about how clusters evolve in space and time. Such results can lead to many advantages such as higher income, reduced costs, and better transportation services, as well as the discovery of trends in cluster movement and improved decision-making processes in city planning.

Acknowledgements

I would like to thank my father, Alberto Lima Portugal, my mother, Maria Miriam da Silva Portugal, and my brother, Erlon da Silva Portugal, for their unconditional love and support.

I would also like to thank all the people that encouraged me to conclude this PhD.

Dedication

This is dedicated to my family.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AI Artificial Intelligence 134, 135

APOC Awesome Procedures on Cypher 133

AROC average rate of change 12, 17, 103, 106–108, 131, 179

BCC Bipartite Correlation Clustering 26

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies 28

CC Correlation Clustering 26

DAG Directed Acyclic Graph 53

DBSCAN Density-Based Spatial Clustering of Applications with Noise xi, 11, 25, 27, 28, 42, 43, 49, 62, 68, 153, 154

DTW Dynamic Time Warping 110

EM Expectation-Maximization 26, 41

GB gigabyte 70

GIS Geographic Information System 66

GPS Global Positioning System 1, 39, 79, 108

GSP Generalized Sequential Patterns 35

KB kilobyte 69

MB megabyte 70

ML Machine Learning 2, 3, 17, 62, 65, 66, 91, 108, 117, 118, 131, 134, 135

MRF Markov Random Field 24

OPTICS Ordering Points To Identify the Clustering Structure 25

OSM OpenStreetMap 66

OWL Web Ontology Language xi, 32

QGIS Quantum GIS 66

SQL Structured Query Language 133

ST-DBSCAN Spatial-Temporal Density-Based Spatial Clustering of Applications with Noise 3, 5, 28, 29, 42

SVM Support Vector Machine 3, 24

T-OPTICS Trajectory - Ordering Points To Identify the Clustering Structure 28

TraClus Trajectory Clustering 5, 28

UTC Coordinated Universal Time 61

W3C World Wide Web Consortium 32

List of Symbols

$O = \{o_1, \ldots, o_m\}$ A set of m spatial-temporal objects 153

$C = \{c_1, \ldots, c_n\}$ A set of n spatial-temporal clusters 153

o_k A spatial-temporal object k. 153

c_i A spatial-temporal cluster i. 40, 153

t_j A timestamp j. 40

o_{k,t_j} A spatial-temporal object k during timestamp t_j. 153

c_{i,t_j} A spatial-temporal cluster i during timestamp t_j. 40, 110, 153

rate The size of the regular interval in which spatial-temporal data is queried for processing. 68

min_shared The minimum value for the Jaccard-extended similarity function used in the proposed approach to consider two clusters in consecutive timestamps as the related. 68

ε The radius of the neighborhood of a data point in DBSCAN. 42, 68, 153, 154

min_cluster The minimum number of data points in the neighborhood of a data point for it to be classified as a core point in DBSCAN. 42, 68, 153

Chapter 1

Introduction

1.1 Motivation

Spatial-temporal data hold information about the location of an object at regular time intervals [4]. For example, Global Positioning System (GPS) devices attached to taxis show the location of the taxis in the city at every minute of the day.

The analysis of spatial-temporal data impacts society in different ways [33]. For example, ridesharing applications allow anyone to move in a city by finding a driver who is willing to give a ride [145]. Ridesharing applications set the price of a ride based on many parameters, such as traffic conditions and the supply and demand of rides at the start location. Estimating traffic conditions and supply and demand of rides at a specific location usually requires the analysis of spatial-temporal data of many cars. Therefore, analyses on the spatial-temporal data generated by many cars, some of which are presented in this thesis, contribute to a better estimation of traffic condition and offer and demand of transportation services, which results in improved pricing of these services. Affordable prices mean that more people take rides to their destinations, improving the revenue of car riding companies and offering suitable prices for customers. Another example is analyzing the path of a shopper in a supermarket. The shopper may have a mobile phone with GPS capabilities and may share location and time information with the supermarket. The analysis of the spatial-temporal information of several shoppers inside the supermarket assists the supermarket in analyzing the many different paths created by the shoppers and in offering discounts to a shopper at the time specific products are approached, which increase the supermarket's profit and assures that customers find products they want. A third example is human mobility, characterized by the study of the movement of individuals and the groups they form. Information about human movement can be used in the generation of targeted advertisement and of insights for city planning to improve the well-being of a population.

These impacts relate to some areas. For example, ridesharing apps seeking affordable prices act on the area of spatial-temporal methods, using analysis techniques, such as clustering, to find interesting patterns in data and adaptation strategies. A supermarket seeking to increase profits relates to the area of Machine Learning (ML) because of the ML methods that can be used in the analysis of the movement of several customers to predict the behavior of a new customer. The city hall of a city may attempt to improve mobility of citizens and visitors by leveraging insights developed in the area of graph-based representation and analysis, such as the modeling and analysis of people and the placement of advertisement or tourist groups and points of interest. Each of the related areas described are further explained in the next section.

In summary, spatial-temporal data and its analysis provide a wide range of benefits. It is important to continue investigating novel impacts that this type of data and its analysis has on people with the goal of improving how people interact with the world around them.

1.2 Related Areas

This section describes three major research areas and positions the study of this thesis in the literature. These research areas are: Spatial-Temporal Methods, Machine Learning Approaches, and Graph-Based Representation and Analysis. Details are provided in the following subsections.

1.2.1 Spatial-Temporal Methods

Spatial-temporal analysis contains methods to analyze data described in terms of spatial dimensions and a temporal dimension [39]. Although the number of areas and methods change constantly, the studies in [150] and [88] describe five areas and some methods for spatial-temporal analysis. A description of these areas and methods follows.

Spatial-temporal methods are grouped into five major areas: uncertainty, frequent pattern mining, group pattern mining, classification, and clustering. The uncertainty in the movement of objects exists because of spatial-temporal data about their movements are captured in a discrete manner, as opposed to a continuous manner. Therefore, the location of an object during the time passed between the times of two discrete location measurements must be estimated. Common solutions include probabilistic models [99]. In frequent pattern mining, routes that have been frequently taken are identified. Usually, these routes are modeled based on the sequence of locations visited. The study in [49] introduces the main concepts and reviews several approaches. In group pattern mining, movement patterns of objects that move together are identified and extracted. Common studied patterns are flocks [10], convoys [60], and swarms [81]. Alternative patterns can be found in [77, 76]. Classification, in the spatial-temporal context, relates to the task of categorizing trajectories or segments of a trajectory. The categories can be activities, such as going to the supermarket or stopped at a parking lot, or modes of transportation, such as walking and driving. Several methods can be used to classify spatial-temporal data, including decision trees [68] and Support Vector Machines (SVMs) [24]. Clustering, in the spatial-temporal context, relates to the task of grouping trajectories based on their properties or moving characteristics. They can be used for classification and outlier detection tasks. One popular method is Spatial-Temporal Density-Based Spatial Clustering of Applications with Noise (ST-DBSCAN).

1.2.2 Machine Learning Approaches

Machine Learning (ML) is a research area whose intent is to understand and develop methods that learn, *i.e.* methods that improve their performance based on the amount of available data [136, 118]. There are several ML methods published in the literature and an extensive overview of the methods is a challenging task. However, ML methods can be grouped in three major categories: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is the learning paradigm that describes methods that use labeled data to learn. During the learning phase, methods are presented with data whose final label has been given, usually by humans. For example, when training methods to estimate car prices, data with characteristics of several cars, which are the features, and a final price, which is the label, is presented to the method so it can learn. Once trained, the method can estimate the price of a new car based on features of the new car and past prices it has learned. Some common supervised learning methods are Neural Networks [50, 35, 73], Naïve Bayes Classifier [38, 126, 86], and Linear Regression [45].

Unsupervised learning is the learning paradigm that describes methods that use unlabeled data to learn. During the learning phase, methods are presented data whose final label has not been given. For example, when training methods to identify whether a house is expensive, data with characteristics of several houses, which are the features, including their prices, which is also a feature, is presented to the method so it can learn. Once trained, the method can divide houses in several groups based on the features, and some contextual knowledge is required to name the groups as expensive. Some common unsupervised learning methods are clustering techniques such as K-Means [82, 43] and K-Nearest Neighbors [25].

Reinforcement learning is the learning paradigm that describes methods that use a rewards system to learn. During the execution, methods interact with the environment and are rewarded if they improve their performance for the task at hand. The method then tries to maximize rewards through several trials. For example, when learning to play a game, the method performs several movements that may or may not lead to a victory. At each trial, a reward is calculated based on the distance from the position at the end of the trial and the victory position, expected to be achieved. Movements that end up distant from the victory position receive lower rewards, so that the method discards them. Some common reinforcement methods are Q-Learning [133] and Monte-Carlo methods [70].

1.2.3 Graph-Based Representation and Analysis

A graph is a mathematical structure with nodes and edges connecting these nodes, usually used to model pairwise relationships between objects [31]. Graphs are used in several different domains, such as computer science, mathematics, linguistics, or biology, to represent domain-specific entities and several graph analysis techniques are used to uncover interesting results.

Graph-based representations relates to the modeling of data using graphs. There are several domains where this happens, including the study of social networks [98], natural language [105, 92], prestige [34], molecules [90], or genes [47]. For example, the study in [51] integrates graph data from several different social networks to build a single unified graph representation of user data. Moreover, the study in [115] performs a systematic literature review on studies related to the representation of patient data in graphs. The goal is to investigate the use of decision support systems for diagnosis, medication, or therapy of patients.

Graph-based analysis relates to the task of using analytical tools and algorithms to perform analysis on graphs. Some of these tools are statistical methods while others are algorithms. The study in [44] surveys several statistical methods for graph analysis, including estimations of the size or order of the graph, or the degree distribution of its nodes. The study in [41] and [75] describe several algorithms for graph analysis including shortest path, depth and breadth search, graph coloring, graph matching, and network flow.

1.3 Problem

This section describes the problem addressed in this thesis and identified in graph based spatial-temporal cluster evolution. The problem comprises several issues identified in three main categories: representation, analysis, and implementation. These categories are described in the following subsections.

Spatial-temporal objects may change their location with time. For example, cars in a city may have their location changed when moving on a highway. Moreover, spatial-temporal objects may group or disperse as they move. For example, cars group in the parking lot of a company in the morning and disperse at the end of the day. As they move, group, or disperse, spatial-temporal objects produce spatial-temporal data. Analyzing spatial-temporal data is valuable and can uncover important results, such as the locations where supply or demand of transportation services are high or the general direction of movement in a city. One way to analyze the produced spatial-temporal data is through clustering methods.

Clustering methods such as ST-DBSCAN [14] and Trajectory Clustering (TraClus) [78] identify clusters of objects taking into consideration their spatial and temporal dimensions.

The clusters of spatial-temporal objects identified by these methods do not move. These clusters are formed based on the similarity of the spatial (location) and the temporal (time) dimensions of spatial-temporal data, but they are limited to a given timestamp. For example, consider the situation illustrated in Figure 1.1. In the figure, the spatial dimension (location) is described with a Cartesian coordinate system and the temporal dimension (time) is described implicitly with annotations.

Assuming that a cluster of three data points can be formed, there are two clusters: one cluster whose center is at the location $(1, 1)$, after 1 second, containing three data points, and another cluster whose center is at the location $(10, 10)$, after 10 seconds, containing other three data points. These clusters have indeed been formed based on the spatial and temporal dimensions of the available spatial-temporal data, but each is limited to its own timestamp. Analysis on these clusters of spatial-temporal objects are limited and do not take into consideration the entire movement, or evolution, of the cluster between two different timestamps.

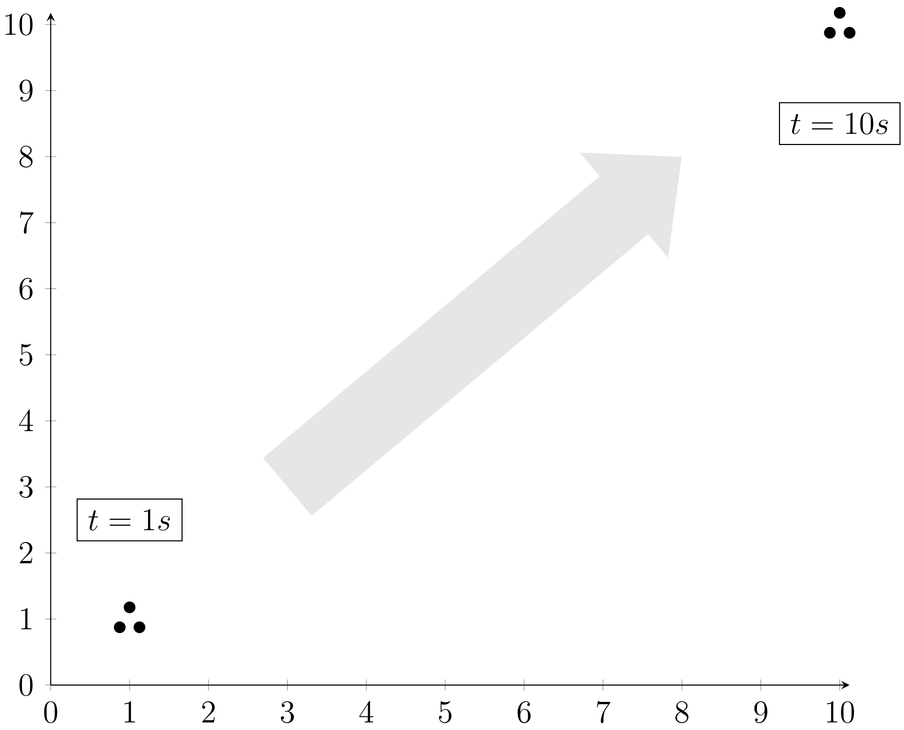


Figure 1.1: A cluster is formed at the bottom left at $t = 1$ second and another cluster is formed at the top right at $t = 10$ seconds. Current cluster analysis methods do not take into consideration the movement or evolution of the cluster between timestamps.

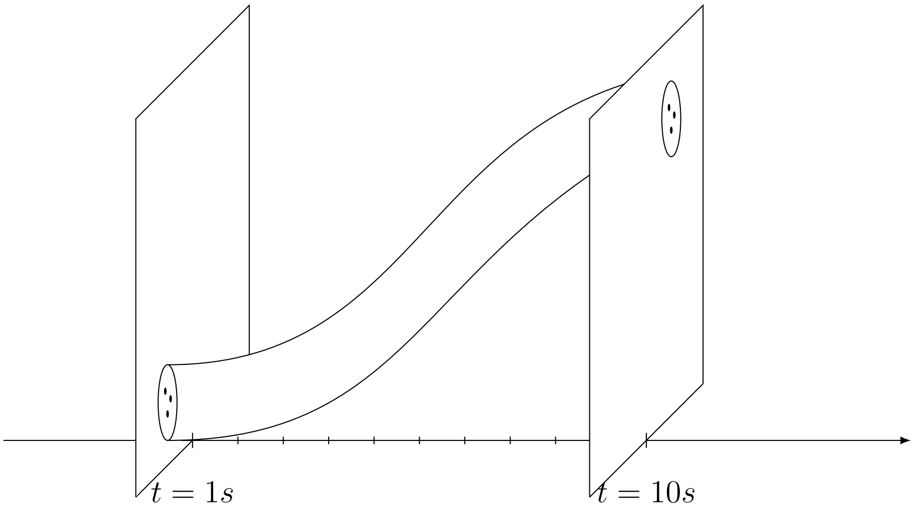


Figure 1.2: An evolving cluster that exists from time $t = 1s$ to time $t = 10s$.

An evolving cluster, on the other hand, is not limited to a timestamp. Consider the situation illustrated in Figure 1.2. In the figure, the temporal dimension is represented by the horizontal axis and the spatial dimension is represented by a rectangle, drawn in perspective, that is the Cartesian coordinate system. The values of the axes x and y are omitted. The figure shows a cluster at the bottom left of the coordinate system at $t = 1$ second and that moved to during nine seconds to the top right part of the coordinate system at $t = 10$ seconds. Notice that an evolving cluster, like the one in the figure, has a different *structure*. It is not limited to one timestamp since it exists for several timestamps. Evolving clusters have their location changed according to the movement of the spatial-temporal objects contained. In addition, evolving clusters have some interactions, or *relationships*, during their evolution: spatial-temporal objects may enter or leave the clusters, clusters may enter or leave another cluster, clusters may merge or split as they evolve in time. Analysis on cluster evolution, with respect to both their structure and relationships, helps uncover novel results.

Finally, the evolution of many such clusters, including changes to their structure and including their relationships can be connected, as in a *graph*, providing an overview of cluster evolution. One example is shown in Figure 1.3. The figure shows one possible way to present the evolution of a cluster c_1 with a graph. Depending on the information presented, details about the structure and the relationships of the cluster can be identified. In this representation, for example, it is possible to identify that cluster c_1 changed its structure at the beginning, thus the repetition. It is also possible to identify that cluster

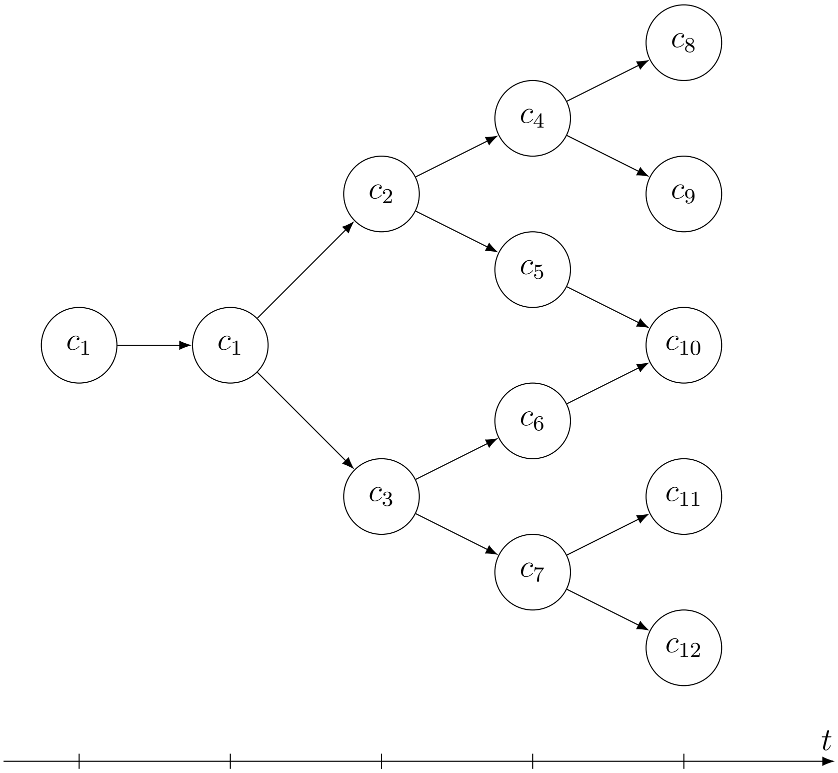


Figure 1.3: This graph is a possible way to represent cluster evolution. The graph represents the evolution of a cluster c_1.

c_1 formed clusters c_2 and c_3, or that cluster c_{10} is part of the evolution of cluster c_1. A complete graph shows how each cluster evolved in time, detailing changes to the structure of each cluster and presenting every relationship each cluster has. Analysis on this graph can improve the understanding of the movement of spatial-temporal clusters and lead to novel results.

1.3.1 Representation

There is a lack of studies on the *representation* of cluster evolution. This evolution is based on the movement of the spatial-temporal objects contained in the evolving clusters.

Representing the *structure* of evolving clusters is challenging because information about these clusters is discretized throughout several timestamps. This requires a way to identify the same cluster in different timestamps and a way to link the identified parts, forming the evolving cluster.

There is also a lack of studies on the representation of the *relationships* the evolving clusters have with spatial-temporal objects or other clusters. Representing these relationships is challenging because identifying them is not trivial. Evolving clusters may change properties at each timestamp, such as the number of spatial-temporal objects contained. These changes may indicate that a relationship, such as a split, is taking place. However, differentiating between a split or a cluster that left a larger evolving cluster based on the properties that change between timestamps is essential and not easy.

Additionally, there is a lack of research on a connected representation of cluster evolution. Since evolving clusters change their structure and have relationships with other clusters during evolution, it is possible to build a connected representation of the evolution of these many clusters. Such representation can show, for example, the name and the time of each relationship evolving clusters had, including when these relationships involve more than one evolving cluster. One possible solution for such representation is the use of a *graph*. However, there are no approaches that can process information about cluster evolution, *i.e.* changes in the structure of clusters and presence of relationships cluster have, and produce a representation in a graph format. Representing cluster evolution in a graph containing all this information creates opportunities for several novel applications.

Some questions about the representation of the cluster evolution, related to the structure of clusters, the relationships clusters have with spatial-temporal objects or other clusters, as well as a graph-based representation of this information are in presented Table 1.1.

1.3.2 Analysis

There is a lack of *analysis* methods for cluster evolution that take into consideration the changes in the structure of evolving clusters, the relationships they have with other spatial-temporal objects or other clusters, or a connected, graph-based representation of cluster evolution.

Current cluster analysis methods consider a restricted representation of clusters, limited to a timestamp. For example, the intracluster distance, *i.e.* the distance between data points inside the cluster, and the intercluster distance, *i.e.* the distance between data points in different clusters, are two cluster analysis methods for the measurement of clusters. These methods do not consider the evolution of clusters with time. For example, changes to the location of the cluster generate a direction of movement, or changes to the number of spatial-temporal objects contained indicate a growth of the cluster. The current analysis methods do not capture information about the *structure* of evolving clusters and miss the opportunity to deliver novel results.

In addition, the current cluster *analysis* methods do not consider the *relationships* evolving clusters have with spatial-temporal objects or other evolving clusters. Information about the location, time, or the number of occurrences of the relationship of a cluster are not captured or analyzed by the current analysis methods and, therefore, remain to have its value assessed.

Moreover, a connected representation of cluster evolution allows several novel analysis techniques to be used. Such representation is suitable for a *graph*. Therefore, analysis on this graph-based representation implies the use of graph analysis methods. However, the current graph analysis methods, such as distance between edges, radius of a graph, and the shortest path, are specific for generic graphs. The current graph analysis methods do not take into consideration cluster evolution, with its changes in cluster structure and the presence of relationships between clusters, represented in a graph, which highlights the need for novel methods. For example, a cluster that, during its evolution, receives many other smaller clusters, and is in a highway may indicate the start of a traffic jam. The current and limited graph analysis techniques do not consider cluster evolution and miss important results. Cluster analysis methods that consider the cluster evolution produce novel results and help reach new conclusions.

Some questions about the lack of analysis methods for the evolution of clusters, including changes in their structure, the presence of relationships during the evolution, and the connected, graph-based representation of the evolution of many clusters of spatial-temporal objects are in presented Table 1.1.

1.3.3 Implementation

There is a lack of *implementation* tools, that is software support, for the representation and analysis of cluster evolution, including changes in cluster structure, presence of relationships and a connected, graph-based representation of the evolution.

The current software tools alone are limited in their *representation* of evolving clusters. For example, one of the most popular libraries for machine learning currently is scikit-learn[1] for Python. This library contains the implementation of 11 clustering methods, including K-Means [82, 43] and DBSCAN [40]. None of them considers an evolving cluster, where its representation spans several timestamps. Moreover, no software tool identifies relationships that evolving clusters have with spatial-temporal objects and other evolving clusters.

The current software tools alone are limited in their *analysis* of evolving clusters. R[2] is a popular language and environment for statistical computing. It contains packages for several analysis methods on graphs. R and the library scikit-learn for Python offers analysis methods for graphs such as the average distance between clusters, the average distance between data points within clusters, the Adjusted Rand Index, the Fowlkes-Mallows Score, the Silhouette Coefficient, and the Dunn index. These analytical methods are limited to a representation of clusters that do not take into consideration an evolving aspect, with changes in the structure of the cluster and the presence of relationships.

Some questions about the lack of software support for the representation and analysis of cluster evolution are in presented Table 1.1.

1.3.4 Research Questions

Here is a list of the research questions related to this thesis.

RQ How to define an approach to support graph-based spatial-temporal cluster evolution representation, analysis, and implementation?

RQ 1 How to represent graph-based spatial-temporal cluster evolution?

RQ 2 How to analyze graph-based spatial-temporal cluster evolution?

RQ 3 How to provide implementation tools to support graph-based spatial-temporal cluster evolution?

[1] https://scikit-learn.org

[2] https://www.r-project.org

Table 1.1: Questions about the problems identified in cluster evolution.

Aspect	Questions
Representation	1. How to model a cluster that evolves during a period of time? 2. How to model the structure, formed by the contained spatial-temporal objects, of a cluster that evolves with time? 3. How to calculate the position of a cluster that evolves with time based on the spatial-temporal objects it contains? 4. How to model the relationships a cluster that evolves with time has with spatial-temporal objects or other clusters? 5. How to model the cluster evolution of several clusters in a connected, graph-based format?
Analysis	1. What is the AROC of the size of a cluster that evolves with time? 2. What is the direction of movement of a cluster that evolves with time? 3. What is the distribution of the direction of movement of a set of clusters that evolve with time? 4. At what time does a cluster that evolves with time start / end its movement? 5. What are the relationships that a cluster has during its evolution? 6. How many clusters split during their evolution into a set of clusters that evolve with time? 7. How to visualize clusters that evolve with time? 8. What graph analysis methods can be used to analyze cluster evolution?
Implementation	1. What implementation tools can be used to identify a cluster that evolves with time based on spatial-temporal data? 2. What implementation tools can be used to identify the same cluster that evolves with time at different times during evolution? 3. What current tools or libraries can be adapted to support the representation and analysis of cluster evolution and the relationships clusters have during their evolution?

1.4 Approach

The approach proposed in this thesis allows for the representation of evolving clusters, including their structure and relationships, and allows for a graph-based representation of cluster evolution. The approach also allows for the analysis of these evolving clusters, including changes to their structure and presence of their relationships, with suitable analytical methods and a visualization of how clusters evolve in time and of their relationships. Finally, the proposed approach provides the full implementation of the structure of evolving clusters and their relationship and of appropriate analytical tools as the software support needed for cluster evolution representation and analysis.

1.4.1 Representation

The proposed approach provides the *representation* of graph-based spatial-temporal cluster evolution by identifying and modeling the structure of evolving clusters, the relationships they have with other clusters, and by using a graph structure to model a connected representation of the evolution of several clusters.

The proposed approach provides the representation of evolving clusters, that is clusters that evolve with time based on the spatial and temporal dimensions of the spatial-temporal objects the clusters contain. For example, the approach identifies clusters at consecutive timestamps and links them based on several rules about the objects that remain in the cluster in both timestamps. This is fundamental for the representation of an evolving cluster as it creates the *structure* of such a cluster. More details are found in Subsection 3.2.1

The approach also identifies the *relationships* these evolving clusters have with spatial-temporal objects and other clusters during the evolution. For example, one cluster may enter another cluster, or merge with another cluster to form a third cluster. Identifying these relationships based on the spatial-temporal data for each timestamp is not trivial because clusters appear, disappear, or change from one timestamp to another. The proposed approach identifies and represents the relationships clusters have while evolving based on a set of rules described in Subsection 3.2.2.

The proposed approach uses a *graph* to represent cluster evolution. Representations of evolving clusters at specific timestamps are transformed into nodes of a graph and relationships are modeled as edges of a graph. This results in a representation of cluster evolution in which evolving clusters are connected based on the relationships they have during their evolution. Details about the construction of such graph are found in Subsection 3.2.3

1.4.2 Analysis

The proposed approach provides *analysis* methods for graph-based spatial-temporal cluster evolution, taking into consideration the structure of the evolving cluster, the relationships, and a graph-based representation of such clusters.

The proposed approach provides analysis methods that examine the *structure* of evolving clusters. For example, it is possible to identify the times when a cluster has the number of contained spatial-temporal objects growing or decaying, indicating for example, if such cluster is expected to grow or decay in the morning, afternoon, or evening. Another example is based on the changes in the location of the cluster. The proposed approach provides analysis methods to calculate the distribution of movement, the distribution of the time where movement happens, or the distance traveled in the movement. Details on the analysis methods for the structure of cluster evolution are found in Subsection 3.3.1.

Analysis of the *relationships* clusters have during evolution can also be performed because the approach provides methods for the analysis of these relationships, such as the ratio of clusters that were formed by the merging of two or more evolving clusters, or the number of times spatial-temporal objects entered or left clusters. The result of these analyses are patterns that can be used for many applications, such as classification. Details on the analysis methods for the relationships of evolving clusters can be found in Subsection 3.3.2.

The proposed approach uses a graph to provide a *graph*-based cluster evolution analysis of spatial-temporal objects. This graph allows for a connected representation of cluster evolution in which evolving clusters are linked based on the relationships they have during evolution. As a result, the approach includes graph-based analysis methods for novel results. For example, when analyzing the location and relationships of clusters, it is possible to identify clusters which spatial-temporal objects migrate to or from and identify, based on the locations of such clusters, regions of high or low offer or demand of services, such as taxis. More details on analysis on a *graph*-based representation of cluster evolution are found in Subsection 3.3.3.

1.4.3 Implementation

The proposed approach provides the *implementation* tools, as a software support, for the representation and analysis of graph-based spatial-temporal cluster evolution.

The proposed approach provides the implementation tools for the *representation* of evolving clusters, including the changes to their structure and the presence of relationships. For example, the approach defines ways to identify and represent the structure of a cluster uniquely as it evolves using libraries such as scikit-learn from Python for clustering and similarity of clusters in consecutive timestamps, ways to represent the relationships that an evolving cluster has in its evolution alongside the identification of other participants, spatial-temporal objects or clusters, in the relationship, using novel rules. The approach also supports the representation of a graph based spatial-temporal cluster evolution using the support of Neo4j[3], a graph data platform. Details about software support provided for the representation of graph based spatial-temporal cluster evolution are found in Subsection 3.4.1.

The proposed approach provides the implementation tools for the *analysis* of evolving clusters, including the changes to their structure and the presence of relationships. Analysis is possible after evolving clusters have been translated to a graph format and imported in Neo4j. For that reason, conversion routines that change the representation of evolving clusters to a graph format, based on nodes and edges, are required and provided. Once imported in Neo4j, several previously saved routines can be executed on the structure of clusters and their relationships to analyze data and generate novel results. Moreover, since the graph is a connected representation of cluster evolution, presaved routines can examine the links between clusters, producing results that were not possible without this support. Presaved routines are written in Cypher[4], a graph query language. Details about software support provided for the analysis of graph based spatial-temporal cluster evolution are found in Subsection 3.4.2

In summary, the proposed approach represents cluster evolution by, first,

1. clustering data at each timestamp to identify parts of spatial-temporal clusters,
2. calculating a similarity between cluster parts in consecutive timestamps,
3. linking similar cluster parts to form a complete spatial-temporal cluster,
4. identifying cluster relationships based on the definition of 14 cluster relationships, and

[3] https://neo4j.com

[4] https://neo4j.com/developer/cypher/

5. transforming the representation of spatial-temporal clusters to a graph format, showing a complete cluster evolution.

Then, the proposed approach analyzes cluster evolution by running methods to

1. examine cluster properties, such as size or location, of several spatial-temporal clusters,
2. evaluate cluster relationships, such as MERGE or SPLIT, and how they contribute to, for example, the demand of taxis, and
3. inspect the graph containing the evolution of several clusters and to identify, for example, improvements to the city transit network.

The proposed approach provides implementation tools to support the representation and analysis of cluster evolution, such as

1. Neo4j, a graph data platform, to visualize and analyze, with custom analysis methods, the graph containing the evolution of clusters,
2. QGIS, a geographic information system to visualize data on maps,
3. cluster-to-graph transformation routines that allow the graph to be built.

1.5 Evaluation

The evaluation of the proposed approach is done in terms of four case studies. These case studies are chosen to discuss interesting phenomena that can be found in spatial-temporal data when using the proposed approach. The case studies deal with the exploratory analysis of the data, the identification of ever-increasing regions, the calculation of a similarity value, and the analysis of the movement represented by cluster evolution. Spatial-data used in this proposed approach comes from four datasets. Each dataset and case study is described briefly in the next paragraphs.

The four datasets used in the case studies describe the movement of vehicles or people in different regions. The dataset Athens Trucks contains the movement of cement trucks in Athens, Greece and the dataset Rome Taxis contains the movement of taxis in Rome, Italy. Geolife and T-Drive contain measurements taken in Beijing, China. The former dataset has the movement of individuals during daily activities, such as commuting or leisure, while the latter dataset has the movement of taxis. Measurements are taken at regular intervals that can be as low as 1 minute. The time period over which these measurements are taken can range from a week to three years.

Case Study 1 is an exploratory analysis. The case study contains 23 questions that investigate the structure and relationships of spatial-temporal clusters as well as their graph-based evolution. These questions are translated into database queries and performed. Their results are discussed and summarized in a table or in a figure. Examples of such questions relate to the identification of spatial-temporal clusters with the greatest number of spatial-temporal objects or the identification of clusters that start at a particular time. The results contribute to several applications, such as offer and demand calculation, or can be used in additional investigation steps.

Case Study 2 relates to the identification of ever-increasing and ever-decreasing regions in the evolution of a spatial-temporal cluster, as well as the calculation of an AROC of size of a cluster. An ever-increasing region in the evolution of a cluster is a moment in which the cluster either grows or maintains its current size. The definition of an ever-decreasing region is analogous. Identifying these regions help in the assessment of the location of time of important phenomena, such as traffic jams or events in a city. Once ever-increasing regions, or ever-decreasing regions, are identified, the average rate of change of size can be calculated based on the size at the start and end of the region. This rate quantifies the increase, or decrease, in the number of spatial-temporal objects in a cluster and can be used, for example, to assess the readiness of spatial-temporal services. The case study identifies and discusses ever increasing regions of two clusters and calculates the average rate of change of the size of each of them. Later, they are compared and insights for these rates are discussed.

Case Study 3 relates to the calculation of a similarity value between the evolution of two clusters. The evolution of a cluster is represented in a graph. This evolution has a single start, but may have several ends because of the cluster relationships that the cluster may have during its existence. Comparing the evolution of two clusters can be performed based, for example, on the size of each cluster at several timestamps during its evolution between the one start and one end. If one cluster evolution is said to be, for example, an optimal way to navigate in a given day in a city, such similarity value can be used to identify optimal ways to navigate in other days or in other cities. Additionally, the calculation of a similarity value is a fundamental step in several ML tasks, such as classification and clustering. The case study compares the evolution of clusters based on their hour of the day they start and discusses the impacts of the resulting values in the price of profitability of spatial-temporal services, as well as the use of these similarities in other algorithms.

Case Study 4 relates to the analysis of the movement represented by the evolution of spatial-temporal clusters. Movement can be represented in several ways, such as by its start location or time, direction, or distance. Analyzing individual movements, either of spatial-temporal objects or spatial-temporal clusters, may give important insights about spatial-temporal phenomena, but an improved approach is the analysis of an aggregation of movements of several spatial-temporal objects or clusters. In the improved approach the movements can be distributed by space, time, or other extended measurement unit. The case study identifies the evolution of clusters that start at specific position and analyze them based on four questions, related to the direction of movement, the distribution by region and day, the time they start, and the largest distance. The results can be used, for example, in improvements in city infrastructure related to city planning, traffic flow, or the road network.

1.6 Contributions

Cluster evolution analysis offers many opportunities for novel discoveries and conclusions in the spatial-temporal domain. These opportunities may lead to a better understanding of the vehicle network in a city, offer and demand of transport services, similarity of cluster evolution or trends in the movement of spatial-temporal clusters, that can be used to increase revenue, reduce costs, offer better transportation services, or improve decision-making processes in city planning, to name a few. However, the lack of representation, analysis, and implementation tools hinders its exploration and, as a consequence, hides its value. This study is one fundamental step towards the full potential of cluster evolution, proposing the representation, analysis, and implementation tools to be used in studies, laying the groundwork for novel research in the area.

The study in this thesis has several contributions. The study identifies and represents evolving clusters, their structure and their relationships with spatial-temporal objects and other evolving clusters. The study also represents cluster evolution using a connected, graph-based description of evolving clusters. The study has analysis methods for cluster evolution, which analyzes the structure and the relationships of evolving clusters, as well as a graph-based representation of cluster evolution. The study also has implementation tools, providing software support for the representation and analysis of graph-based spatial-temporal cluster evolution.

1.6.1 Representation

The proposed approach has contributions in the *representation* of graph-based spatial-temporal cluster evolution.

The proposed approach has contributions in the representation of the *structure* of evolving clusters and their relationships. Current clustering techniques identify clusters at a particular time. The approach described in this thesis can identify and represent a cluster throughout time, *i.e.* an evolving cluster, including its changes in location or number of participants. The approach combines the execution of a clustering technique at several timestamps with a modified Jaccard similarity index [58] for the similarity of clusters in different timestamps.

The approach can identify when clusters have *relationships* with other clusters. This identification is based on the spatial-temporal objects participating in clusters in each timestamp involved in the relationship. A set of rules was developed to identify the relationship. There are 22 relationships identified and represented, including C_ENTER or C_LEAVE when a cluster enters or leaves another, and MERGE or SPLIT when two or more clusters merge to form a larger cluster or split to form several smaller clusters.

The proposed approach has contributions in the graph-based representation of cluster evolution. Since the representation uses a graph, is based on nodes and edges. Moreover, the approach transforms representation of the cluster structure and the relationships to a graph format, based on nodes and edges.

1.6.2 Analysis

The proposed approach has contributions in the *analysis* of graph based spatial-temporal cluster evolution.

During its evolution, clusters may undergo some changes in *structure*, such as a change in the location or a change in the number of participants. The approach proposed in this thesis provides cluster analysis methods to be used for novel results. For example, the approach allows for the calculation of cluster growth or decay parameters, which relate to moments during the existence of the cluster when the number of participants increase or decrease. The approach also allows for the calculation of ever-increasing or ever-decreasing regions, which are moments when the number of participants of clusters always increase or always decrease, indicating interesting phenomena.

The approach provides analysis methods for the *relationships* evolving clusters have as well. For example, the approach allows for the calculation of the number of MERGE relationships in specific locations, which may indicate the start of a traffic jam or another relevant event. Conversely, the presence of several SPLIT relationships at a specific location may indicate the end of an event.

Using the *graph*-based representation of cluster evolution, in which clusters assume a graph format and are linked based on their relationships, the approach has analysis methods to identify locations where the offer or demand of a particular service, such as taxi, is high or low based on the size of the evolving clusters and the relationships they are having around location of interest.

1.6.3 Implementation

The proposed approach has contributions in the *implementation* tools used to provide software support for the representation and analysis of graph based spatial-temporal cluster evolution.

The proposed approach has contributions in the implementation of, or software support for, the *representation* of cluster evolution. The approach unites Python libraries and Neo4j, a graph data platform, to provide complete support for the representation of evolving clusters. Some examples are the scikit-learn library, for clustering techniques and geopy[5] for the calculation of distance between coordinates considering the curvature of Earth.

The proposed approach has contributions in the implementation of or software support for, the *analysis* of cluster evolution. The approach uses Neo4j's Cypher, a graph querying language, to provide complete support for the analysis of evolving clusters. The approach includes a set of routines for analysis of cluster evolution, including routines for identifying temporal regions during which an evolving cluster has its number of participating spatial-temporal objects increasing or decreasing, evolving clusters that exist for a long time, or that move to a distant place, and geographical regions to which clusters migrate indicating a high demand for a specific service.

1.7 Publications

The following publications were produced as a result of this study (in chronological order):

[1] Ivens Portugal, Paulo Alencar, and Donald Cowan. Towards a provenance-aware spatial-temporal architectural framework for massive data integration and analysis. In *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pages 2686–2691. IEEE, 2016. DOI: 10.1109/BigData.2016.7840912.

[2] Ivens Portugal, Paulo Alencar, and Donald Cowan. Developing a spatial-temporal contextual and semantic trajectory clustering framework, 2017. DOI: 10.48550/arXiv.1712.03900.

[5] https://geopy.readthedocs.io

[3] Ivens Portugal, Paulo Alencar, and Donald Cowan. Cluster lifecycle analysis: challenges, techniques, and framework, 2018. DOI: 10.48550/arXiv.1901.02704.

[4] Ivens Portugal, Paulo Alencar, and Donald Cowan. A software framework for cluster lifecycle analysis in transportation. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, pages 4534–4539. IEEE, 2019. DOI: 10.1109/BigData.2018.8622576.

[5] Ivens Portugal, Paulo Alencar, and Donald Cowan. Modeling dynamic spatial-temporal cluster relations. In *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, pages 3590–3598. IEEE, 2019. DOI: 10.1109/BigData47090.2019.9006496.

[6] Ivens Portugal, Paulo Alencar, and Donald Cowan. Spatial-temporal cluster relations: a foundation for trajectory cluster lifetime analysis, 2019. DOI: 10.48550/arXiv.1911.02105.

[7] Ivens Portugal, Paulo Alencar, and Donald Cowan. Trajectory cluster lifecycle analysis: an evolutionary perspective. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, pages 3452–3455. IEEE, 2019. DOI: 10.1109/BigData.2018.8621966.

[8] Ivens Portugal, Paulo Alencar, and Donald Cowan. A framework for spatial-temporal trajectory cluster analysis based on dynamic relationships. *IEEE Access*, 8:169775–169793, 2020. DOI: 10.1109/ACCESS.2020.3023376.

[9] Ivens Portugal, Paulo Alencar, and Donald Cowan. From spatial-temporal cluster relationships to lifecycles: framework and mobility applications. In *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*, pages 2410–2415. IEEE, 2020. DOI: 10.1109/BigData50022.2020.9377763.

[10] Ivens Portugal, Paulo Alencar, and Donald Cowan. Computational cluster lifetime analysis to capture cluster lifetime dynamics. In *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, pages 2909–2918. IEEE, 2021. DOI: 10.1109/BigData52589.2021.9671317.

[11] Ivens Portugal, Paulo Alencar, and Donald Cowan. A graph-based analysis approach to cluster lifetime dynamics. In *Proceedings - 2022 IEEE International Conference on Big Data, Big Data 2022*, pages 3831–3837. IEEE, 2022. DOI: 10.1109/BigData55660.2022.10020802.

Other related publications produced during this study include (in chronological order):

[1] Ivens Portugal, Paulo Alencar, and Donald Cowan. Requirements engineering for general recommender systems, 2015. DOI: 10.48550/arXiv.1511.05262.

[2] Ivens Portugal, Paulo Alencar, and Donald Cowan. A preliminary survey on domain-specific languages for machine learning in big data. In *Proceedings - 2016 IEEE International Conference on Software Science, Technology and Engineering, SwSTE 2016*, pages 108–110. IEEE, 2016. DOI: 10.1109/SWSTE.2016.23.

[3] Ivens Portugal, Paulo Alencar, and Donald Cowan. A survey on domain-specific languages for machine learning in big data, 2016. DOI: 10.48550/arXiv.1602.07637.

[4] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: a systematic review. *Expert Systems with Applications*, 97:205–227, 2018. DOI: 10.1016/j.eswa.2017.12.020.

[5] Ivens Portugal, Toacy Oliveira, Paulo Alencar, and Donald Cowan. Mylynsdp - process-aware artifact filtering based on interest. *Journal of the Brazilian Computer Society*, 26(1), 2020. DOI: 10.1186/s13173-020-00100-8.

[6] Peng Peng, Ivens Portugal, Paulo Alencar, and Donald Cowan. A face recognition software framework based on principal component analysis. *PLoS ONE*, 16(7), 2021. DOI: 10.1371/journal.pone.0254965.

1.8 Structure of the Thesis

This thesis is structured as follows. Chapter 2 describes the related work and Chapter 3 describes the proposed approach. Chapter 4 describes four case studies and Chapter 5 concludes the thesis.

Chapter 2

Related Work

This chapter describes research initiatives related to the study proposed in this thesis. It starts with an overview of the domain of spatial-temporal data analysis followed by a description of clustering techniques. Later, the chapter describes possible cluster relationships and ends with a discussion of how they are used to identify and construct the notion of cluster evolution.

2.1 Spatial-Temporal Data Analysis

Objects, such as a car or a phone, that move in space are called spatial-temporal objects. When observed, spatial-temporal objects produce spatial-temporal data. Spatial-temporal data refers to data that is described based on location and time [84, 131]. For example, mobile phones with GPS capabilities produce spatial-temporal data about the position of the phone at several moments during the day. The study in [66, 103] describes spatial-temporal data in an extensive way.

The spatial dimension of spatial-temporal data refers to the location of the spatial-temporal object being observed. Important considerations about the spatial dimension are (i) whether the spatial-temporal object moves, (ii) the physical dimensions of the spatial-temporal object, and (iii) what physical dimension is the spatial-temporal object in. These considerations impact the type of the spatial-temporal data.

Some spatial-temporal objects have a fixed location. As described in [66], they are usually spatial-temporal-events, such as earthquakes. The spatial-temporal data that describe the spatial-temporal events include the fixed location and the fixed time these events happened. Some spatial-temporal objects move in space in several directions, such as cars or people moving in a city. Usually, the spatial-temporal data that describe the movement of spatial-temporal objects has the history of the entire movement of the spatial-temporal object. According to [66], when the whole history of the movement of a spatial-temporal object is available for analysis, the spatial-temporal data is said to refer to a trajectory.

Spatial-temporal objects have physical dimensions, like a truck having its length and width. However, for analysis purposes, spatial temporal objects are usually summarized by the spatial-temporal data type point. This enables, for example, the visualization of the location of every truck of a truck company on a map. This summary may not always be possible, and thus lines or areas could be used as the spatial-temporal data type describing a spatial-temporal object. Some examples are trains in a city or states in a country.

In most cases, the spatial-temporal data describing the movement of a spatial-temporal object does so by using a two-dimensional representation for location of the object, such as latitude and longitude coordinates. However, a third physical dimension, the altitude, may be needed for some analyses. For example, the analysis of the position of airplanes in the air may consider the altitude of the planes for more accurate results.

The temporal dimension of spatial-temporal data refers to the time at which observations of the spatial-temporal object happen. According to [103] an important consideration about the temporal dimension is its granularity. Granularity is specified based on a point in the time axis and a partitioning length [61]. Some observations can happen every minute, in the case of the trajectories of cars, or every day, in the case of the trajectories of planets.

Data analysis refers to the task of using techniques on data to uncover patterns and discover value [54]. Using these techniques on spatial-temporal data characterizes spatial-temporal data analysis. There are many different spatial-temporal data analysis techniques, but [88] separates them into two types based on their primary goal: the clustering and the classification types. According to [127] two common spatial-temporal data analysis techniques for classification tasks are Markov Random Field (MRF) and SVM. MRF [65] requires the set of variables observed to follow the Markov property, that is, future states depend only on the present and not on the past, which is not true for spatial-temporal data in many practical situations. SVM [24] is a technique that attempts to create a hyperplane that divides data into a predefined number of classes. The study in [26] creates a new classifier based on SVM and applies it to the trajectory of tourists. Spatial-temporal data analysis techniques with the goal of clustering are relevant to the development of this thesis and have their own section, presented in the next section.

With respect to performance of spatial-temporal data techniques, the study in [148] evaluates 25 trajectory simplification techniques. Trajectory simplification minimizes information for compression and storage purposes. The study in [130] proposes a quantization technique for a compact representation of trajectories. Quantization is the process of mapping continuous infinite values to a smaller set of discrete finite values. The study in [3] investigates the trajectory search problem, which is informally defined as the search for a set of facilities that maximize a service measure for a set of user trajectories. However, the solution discussed here does not address performance issues directly.

2.2 Clustering

In general, clustering is the task of dividing data into groups such that data in the same group are more similar than data in different groups [139, 140, 59]. There are different types of clustering techniques, including centroid-based [37], density-based [12, 83], distribution-based [142, 141], and hierarchical clustering [110].

Centroid-based clustering techniques [55] partition the data space into regions, which are the clusters, and uses a point, which is the cluster centroid, to determine which cluster each data point belongs. Determination is usually based on the distance from the data point to the centroid. The choice of the number of clusters is important in the execution. The most popular centroid-based clustering technique is K-means [82, 43]. Given a fixed number of clusters, the K-means algorithm attempts to minimize the sum of the distances from each data point to each centroid, updating the centroids and the membership of data points if needed. Other centroid-based clustering techniques are K-medoids clustering [64], which uses data points as cluster centers, and Fuzzy c-Means clustering [36, 11], which allows data points to belong to more than one cluster by assigning a membership degree. A centroid-based cluster technique can be used, for example, in customer segmentation to find types of customers with the same characteristics.

Density-based clustering techniques [113] identify clusters based on the density of data points in regions of the data space. Usually, these techniques can identify clusters of any shape, but may not perform well if clusters have different densities. The most popular density-based clustering technique is DBSCAN, which is explained in this section. Another popular density-based clustering technique is Ordering Points To Identify the Clustering Structure (OPTICS) [6, 5]. The algorithm is similar to DBSCAN and solves DBSCAN's problem of varying density by monitoring a core distance of data point p, that is the distance required to make p a core point. Points of a dense cluster have a small core distance. Density-based clustering techniques can be used in the identification of groups of vehicles or different animals in aerial photos.

Distribution-based clustering techniques [141] use the distribution of data to identify clusters. Often, a data point is assigned a probability to belong to a cluster. A popular distribution-based clustering technique is the Expectation-Maximization (EM) algorithm [91, 89]. EM uses a statistical model to assign data points to clusters. The algorithm is divided in two steps. The expectation step estimates the expected value for variables that are missing in the dataset. The maximization step optimizes the parameters of statistical models. A common statistical model is the Gaussian distribution [46, 28, 29]. Distribution-based clustering techniques can be used in applications in which data is missing or incomplete but the distribution of the data is known, such as in the medical or security domains, where not all data is captured for privacy reasons.

Hierarchical clustering techniques [94, 95] group data in clusters such that a hierarchy of clusters is formed. There are two strategies: the divisive and the agglomerative. In the divisive or top-down strategy, all data points start in the same cluster and any clustering algorithm attempts to partition the data in two clusters. The process is repeated until each data point is its own cluster. In the agglomerative or bottom-up strategy, each data point starts as its own cluster and a similarity is calculated between the existing clusters. Then, the two most similar clusters are joined. The process is repeated until there is only a single cluster. Hierarchical clustering techniques can be used in applications where the data follows a hierarchical structure, such as identifying animal species or the development of a disease.

Real-time clustering [1, 52, 53, 122] is the task of clustering data as it becomes available. Typically, a continuous stream of data is captured, and a clustering algorithm is responsible to cluster data while considering restrictions of memory and time. EvolveCluster [100] is a clustering algorithm with the goal of modeling how data evolves in real time. EvolveCluster can identify clusters in a data segment, which is similar to a timestamp, and uses the cluster centroids of the previous segment to identify the clusters of the current data segment. Although EvolveCluster can identify a split relationship between clusters, that is when a cluster splits into two clusters, other relationships are not identified, including a merge relationship. This limits analysis on the evolution of a cluster. The resulting modeling created by EvolveCluster is comparable to the modeling created by the algorithms PivotBiCluster [2] and Merge-Split [15]. PivotBiCluster identifies clusters in a bipartite graph, that is a graph that can be separated in two parts, by treating objects as nodes and relationships as edges and connecting related nodes. The two parts of the graph can be the objects in two different timestamps. This is the problem of Bipartite Correlation Clustering (BCC), a variant to the more general problem of Correlation Clustering (CC). The PivotBiCluster algorithm is based on probabilities, meaning that two runs of the algorithm may produce different results. The Merge-Split algorithm also identifies clusters in a bipartite graph by identifying bi-cliques, which are connections between the two parts of the graph. Although a spatial-temporal clusters can be identified, relationships between these clusters are limited to split and merge, and the representation of spatial-temporal objects is not discussed. The study in [48] contains a more extensive list of real time clustering algorithm.

Graph clustering is the task of finding sets of vertices in a graph that are related [114]. Applications range from analysis on voter behavior to formation of new trends. The literature includes the study of an evolutionary graph clustering algorithm [13]. An evolutionary algorithm, also known as genetic algorithm, use nature mechanisms such as evolution, mutation, and selection to approximate an optimal solution to a problem [134, 135]. In the study, an evolutionary graph clustering algorithm is developed to maximize modularity, which relates to the connections between vertices within a cluster in a graph. Graphs with high modularity have many connections within a cluster and few connections outside of the cluster. Although an evolutionary approach can be used in the identification of spatial-temporal clusters, the amount of spatial-temporal objects in all timestamps need to be analyzed. This may not be feasible especially when considering the number of iterations needed by the algorithm to produce a solution.

There is a plethora of clustering techniques, but one, DBSCAN [40], is further described here because of its relevance for this thesis.

DBSCAN is a density-based clustering technique that groups together a set of points based on density, in other words, that are close to each other. It does so by having two main parameters: ε and $minPts$. The first parameter, ε, specifies the radius of a neighborhood for each point in space. The second parameter, $minPts$, specifies the minimum number of points in the neighborhood of a point. This is used to classify the point in one of the three types: core point, directly-reachable point or outliers. The classification follows these rules:

- A point p is a core point if at least $minPts$ points are within distance ε of it (including p).
- A point q is directly reachable from p if point q is within distance ε from core point p.
- A point q is reachable from p if there is a path p_1, ..., p_n with $p_1 = p$ and $p_n = q$ where each p_{i+1} is directly reachable from p_i.
- All points not reachable from any other point are outliers.

The DBSCAN algorithm starts by picking any point in space that has not been visited. If this point contains sufficiently many points in its neighborhood, ı.e. if there are at least $minPts$ reachable from the initial point, then a new cluster is started. If a point is found to be reachable from some point of the cluster, it is part of the cluster as well, either as a core point or a border point, depending on the number of points in its neighborhood. This process continues until the entire cluster is found. The algorithm then retrieves another point that has not been visited, leading to the discovery of a new cluster or labeling this point as an outlier. One interesting case is when a border point belongs to the neighborhood of core points of different clusters. In theory, the border point can belong to either cluster. However, the original implementation of DBSCAN assigns it to the first cluster that claims it [116].

DBSCAN works on data of different densities and can identify clusters regardless of their densities. This is possible because DBSCAN extends a cluster, including each data point that is reachable from any of the cluster's points based on the parameter ε. However, for clusters of different densities, the parameter ε needs to be carefully chosen such that DBSCAN does not include too many data points in one cluster or too little in another cluster. Usually, the parameter ε is specific to the data under analysis and is chosen after some experimentation.

Spatial-temporal clustering techniques attempt to group spatial-temporal objects based on the similarity of their path, that is, the spatial-temporal data produced [7, 150]. Spatial-temporal data here refers to trajectories because the history of the movement is available for analysis, as opposed to moving points, described in [66]. The study in [7, 120, 88, 127, 144, 66] present an extensive analysis on many spatial-temporal clustering techniques, such as Trajectory - Ordering Points To Identify the Clustering Structure (T-OPTICS) [96, 5], Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [149], and TraClus [78].

Another spatial-temporal clustering technique is ST-DBSCAN [14]. Because of its importance for this thesis, it is detailed in its own paragraph. In general, ST-DBSCAN works like its parent algorithm DBSCAN, but it differs in the calculation of similarity and in the identification of adjacent clusters. To check if a point p has a point q in its neighborhood, a distance function is used. The most common distance function is the Euclidean distance, defined in Equation 2.1, but the Manhattan distance and the Minkowski distance are popular as well [85].

$$dist(p,q) = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2} \tag{2.1}$$

where $p = (x_p, y_p)$ and $q = (x_q, y_q)$ are two 2-dimensional points. Smaller distances mean higher similarity. ST-DBSCAN stores the distance for spatial values in a variable called Eps1 and introduces a new variable for the distance of non-spatial values called Eps2. Eps2 can be used for temporal values or for any non-spatial values, such as temperature. Eps1 and Eps2 are calculated using the Euclidean distance. For the identification of adjacent clusters, the authors of ST-DBSCAN claim that points in opposite sides of a cluster may have very different values and two points in adjacent clusters may not, making it difficult to classify points correctly. ST-DBSCAN solves this problem by comparing the cluster average value and comparing it with the value in the point being visited by the algorithm. This difference should be smaller than a threshold of $\Delta\varepsilon$ set at the beginning of the algorithm and introduced by ST-DBSCAN.

2.3 Cluster Relationships

Spatial-temporal clusters are clusters of spatial-temporal objects. They exist for some period of time and move during existence. During its movement, a spatial-temporal cluster may enter another spatial-temporal cluster, resulting in the end of the first spatial-temporal cluster. Moreover, two spatial-temporal clusters moving in space may meet and merge, creating a third larger spatial-temporal cluster. Situations like the ones presented show that spatial-temporal clusters have relationships with each other. These spatial-temporal cluster relationships are patterns in movement that can be identified, analyzed, and processed to reach useful conclusions. The study of spatial-temporal starts with understanding spatial relationships and temporal relationships before the more advanced spatial-temporal relationships and the analysis of semantics. The next sections detail each step.

2.3.1 Spatial Relationships

In general, spatial relationships refer to the relationships that objects have with respect to their location in space [79, 56]. Note that this definition is generic and applies to objects of any geometry. The study in [79] has an extensive description about spatial relationships. There are three types of spatial relationships: topological, directional, and metric. They are explained in the following paragraphs.

Topological relationships are based on the topology or geometry of the objects. There are three common geometries: point, line, region. The main topological relationships found in the approach in this thesis are between a point and a region, and a region and a region. The study in [79] describes six region and region topological relationships: disjoin, meet,

Table 2.1: Region and region topological relationships. Adapted from [79].

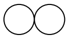
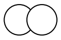
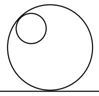
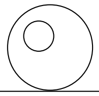
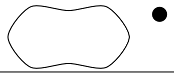
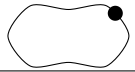
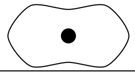
Disjoin	
Meet	
Overlap	
Contain & Meet	
Contain	
Equal	

Table 2.2: Point and region topological relationships. Adapted from [79].

Disjoin	
Meet	
Contain	

overlap, contain, contain & meet, and equal. They can be seen in Figure 2.1. These region and region relationships simulate the relationships that spatial-temporal clusters can have. The study in [79] also describes three point and region topological relationships: disjoin, meet, and contain. They can be seen in Figure 2.2. These point and region relationships simulate the relationships spatial-temporal clusters have with spatial-temporal objects.

Directional relationships are based on the relative location of the objects being compared. In the case of spatial-temporal objects, they are first summarized in a point and then their relative position is analyzed to identify the directional relationship. The study in [56] describes four basic directional relationships: left, right, front, and back. The study in [79] uses cardinal points for a more descriptive set of directional relationships: north, northwest, west, southwest, southeast, east, northeast, and equal. These relationships can be seen in Figure 2.1.

Metric relationships are based on the distance between the objects being compared. In the case of spatial-temporal objects, they are first summarized in a point and then their distance is calculated to identify the metric relationship. The study in [79] describe four relationships: equal, near, medium, far. They can be seen in Figure 2.2.

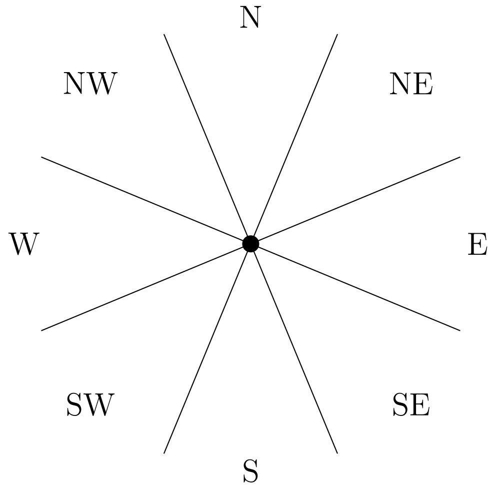


Figure 2.1: Directional relationships based on eight cardinal points. Adapted from [79].

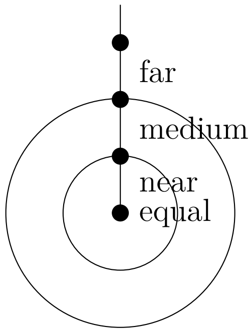


Figure 2.2: Metric relationships. Adapted from [79].

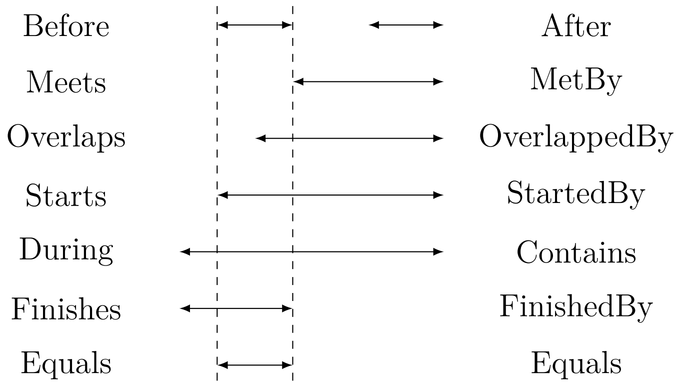


Figure 2.3: Temporal relationships. Adapted from OWL's time ontology[1].

2.3.2 Temporal Relationships

In general, temporal relationships refer to relationships that objects have with respect to the moment they exist in time [56]. The World Wide Web Consortium (W3C) maintains the OWL project which describe ontologies for several domains. The OWL has a time ontology[1], which includes temporal relationships. There are 13 temporal relationships described in OWL's time ontology: before, after, meets, metBy, overlaps, overlappedBy, starts, startedBy, during, contains, finishes, finishedBy, equals. They are visually represented in Figure 2.3. The figure contains seven pairs of relationships, each in one line, starting with the pair Before-After, and ending with the pair Equals-Equals. Each arrow represents the duration of an event of interest, such as a sports match. The uppermost leftmost arrow is the event duration of reference. The vertical dashed lines indicate that the event duration of reference is repeated on each line. Other arrows represent the event duration that is compared to the event duration of reference. Relationships shown on the left side of the figure refer to the event duration of reference and relationships shown on the right side of the figure refer to the event duration compared. For example, on the first line, the event duration of reference happens before the other event duration. On the fifth line, the relationship During indicates that the event duration of reference happens during another event. The same set of temporal relationships is described in [56].

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

Table 2.3: Spatial-temporal relationships described in [80].

Relationship	Description
Appear	An object appears in the system.
Disappear	An object disappears from the system.
Update	An object updates its properties, e.g. velocity.
Exit	An object exits a cluster.
Join	An object joins a cluster.
Expire	A cluster expires.
Merge	At least two clusters merge.
Split	A cluster splits into multiple clusters.

2.3.3 Spatial-Temporal Relationships

In general, spatial-temporal relationships refer to relationships that objects have with respect to both their location in space and the moment they exist in time [56, 18]. One way to discover spatial-temporal relationships is to pick a spatial relationship and a temporal relationship and make sense of the resulting combination. For example, the combination between the spatial relationship north and the temporal relationship before produce a spatial-temporal relationship indicating that an object was to the north of a reference object moments ago. This is the approach followed in [56] and [18]. However, a more robust approach is to identify relationships that objects have based on both the spatial and temporal relationships and natural language. These spatial-temporal relationships are expected to translate to spatial-temporal objects without effort. The study in [27] identifies seven spatial-temporal relationships: appears, disappears, survives, splits, merges, shrinks, and expands. These spatial-temporal relationships describe patterns on moving object data. The study in [80] expands this list of spatial-temporal relationships, identifying eight relationships: appear, disappear, update, exit, join, expire, merge, split. These two lists of spatial-temporal relationships are the starting point for the proposal in this thesis, which is described in Chapter 3.

2.3.4 Semantic Relationships

A higher-level set of spatial-temporal relationships involves the semantic meaning of each relationship. This is the closest to natural language and attempts in this direction benefit non-technical users the most. The study in [21, 20, 22, 93, 19] observes the movement of objects in video and identifies 10 motion verbs that are the semantic relationships: go back, go through, come back, enter, go out, go to, arrive, go into, depart, and leave. Note that one semantic relationship can be described by many spatial-temporal relationships. For example, the semantic relationship go through can be described by the spatial-temporal relationships join and exit described in the previous section. The study in [138, 137] divides the situations in which objects can be observed into 4 groups: move outside, move inside, move on the boundary, and move while englobing. For each of these 4 situations, some semantic relationships are described, totaling 14 semantic relationships. For example, when moving outside, the semantic relationships are approach, aroundOutside, and leave. The study in [69] observes the German language and identifies prepositions related to the description of movement and create a correspondence to semantic relationships. The identified semantic relationships are: towards, up to, into, to, against, away from, out of, along, past, through, and around.

2.4 Cluster Evolution

Clusters, in general, do not evolve in time, that is, cluster properties such as the number of participants or domain-specific properties such as the average height for a cluster of people stay the same once the clustering technique finishes. A novel perspective considers clusters having some properties changed with time, which characterizes an evolving cluster. Therefore, cluster evolution refers to the idea of clusters having their properties changing with time [109]. For example, wild animals may group forming clusters and, since animals enter and leave the cluster, this cluster has its properties changed with time and is said to be an evolving cluster. In the case of spatial-temporal clusters, the location of the cluster may change with time, as well as domain-specific properties.

The study in [109] creates a methodology for the analysis of cluster evolution based on five steps: clustering per time period; between-period similarity identification; forming cluster trajectories; detection of prominent migration patterns; and detailed migration pattern detection. The same model was followed by other researchers in different domains, such as the academic and the software engineering domains [107, 74].

The first step relates to clustering per time period. There are different ways to cluster temporal data or time-series data [132]. One common way is to run a clustering technique at regular time intervals. The main drawback of this approach is that clusters at different time intervals are not easily associated and additional processing is required. In [109] data about corporate bonds, which are one type of security in the financial domain, are divided into five groups based on the year of measurement. The authors use a k-means clustering algorithm on data for each year to cluster bonds based on their risk. An interpretation of this step in the spatial-temporal domain has spatial-temporal data being clustered at regular time intervals. However, additional processing is required in the handling of the spatial-temporal dimension.

The second step relates to the between-period similarity. This is the natural result of running clustering techniques at regular time intervals. The study in [109] describes two ways to compare clusters obtained at different timestamps: comparing shared objects and comparing cluster characteristics. The former way refers to comparing the participants of each cluster and defining a similarity of these clusters. One common way to define such similarity is to use a Jaccard similarity [58]. The Jaccard similarity is calculated as follows:

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

where c_1 and c_2 are clusters and $|\cdot|$ calculates the number of participants in a cluster. The latter way refers to comparing cluster characteristics and checking if they are similar in nature. It is common to use statistical tests such as the T-test [125] or the Chi-square [102, 23] to compare clusters. One interpretation of this step in the spatial-temporal domain is the linkage between the clusters found in each regular interval. However, unlike in the finance domain, changes in the number of objects in clusters can happen rapidly, which poses additional challenges to the linkage task.

The third step relates to identifying the formed cluster trajectories. By cluster trajectories, the authors mean the path that objects, or bonds in the financial domain, generate with its movement between clusters. In [109], the authors build a matrix with the movement of bonds between clusters. The authors use the Generalized Sequential Patterns (GSP) algorithm [123] to identify the most common patterns in the movement of bonds and filters the bonds based on user-defined parameters. The authors seek to identify paths that indicate stability or reduced loss of value among the paths that these bonds create when moving from cluster to cluster. One interpretation of this step in the spatial-temporal domain is to capture the clusters that all spatial-temporal objects belonged to. However, the feasibility of this interpretation is challenged by the amount of data that is available for processing. In the case of [109], the paths are identified from year to year for four years. In contrast, in the spatial-temporal domain, spatial-temporal data about objects, such as cars or people, can be based on much smaller regular intervals, usually involving a period of one to a few minutes. Additionally, spatial-temporal data about objects may involve millions of vehicles or people. A new interpretation that leverages the spatial-dimension of clusters and analyzes their movement needs to be explored.

The fourth step relates to detecting prominent migration patterns. In [109], a migration is when a corporate bond moves from one cluster to another cluster based on risk assessment. The authors seek to identify migration repeated paths that characterize bond deterioration, for example. One interpretation of this step in the spatial-temporal domain relates to spatial-temporal objects that leave a spatial-temporal cluster and enter another. In this case, leave and enter are relationships between a spatial-temporal object and a spatial-temporal cluster. As an area of further investigation, this interpretation can be extended to analyze clusters that leave and enter other spatial-temporal clusters because of the spatial dimension. In summary, this interpretation is related to the identification of spatial-temporal cluster relationships.

The fifth step relates to detailing the detected migration patterns. In [109], the authors use well-defined risk groups for corporate bonds and create a diagram with bars and arrows indicating the movement of bonds among these risk groups. The authors seek a visual representation of the movement of bonds. One interpretation of this step in the spatial-temporal domain assumes that the risk groups are spatial-temporal clusters and that corporate bonds are spatial-temporal objects, so they can be represented in the diagram. However, such a diagram with bars and arrows is very limited to represent the sheer amount of relationships that happen in the spatial-temporal domain. Additionally, this visual representation limits analysis on the movement of spatial-temporal clusters. In summary, this interpretation motivates the need for an improved visualization of spatial-temporal clusters and spatial-temporal cluster relationships for analysis.

The methodology for cluster evolution analysis described in [109] serves as an inspiration for the proposal of this thesis. The several limitations of the methodology are overcome with novel steps, algorithms, and approaches that consider the spatial and temporal dimensions to represent and analyze graph-based cluster evolution. A detailed description is given in Chapter 3.

Chapter 3

Graph-Based Spatial-Temporal Cluster Evolution

This chapter describes the proposed approach for graph-based spatial-temporal cluster evolution. It starts with an overview of the approach, followed by a detailed description. Next, individual contributions and solutions are discussed.

3.1 Overview

The proposed approach captures spatial-temporal data and delivers cluster evolution representation, analysis, and implementation tools through a series of steps, which are described in this section. Figure 3.1 shows a workflow including the steps of the proposed approach.

The first step is to preprocess the data. Because spatial-temporal data can be represented in different formats, the approach has a preprocess step to standardize data. The approach reads spatial-temporal data from a repository, represented in Figure 3.1 by the entity called “Original ST data” and stores the resulting data in a repository, represented in Figure 3.1 by the entity called “Preprocessed ST data”. Activities performed during this step are mainly related to the format and order that latitude and longitude coordinates and the time are represented. Because of the technical nature of the explanation, a detailed description of these activities is deferred to Section 3.4.1.

The second step is to process the spatial-temporal data. Three major tasks happen during this step: the preprocessed spatial-temporal data is clustered, spatial-temporal clusters

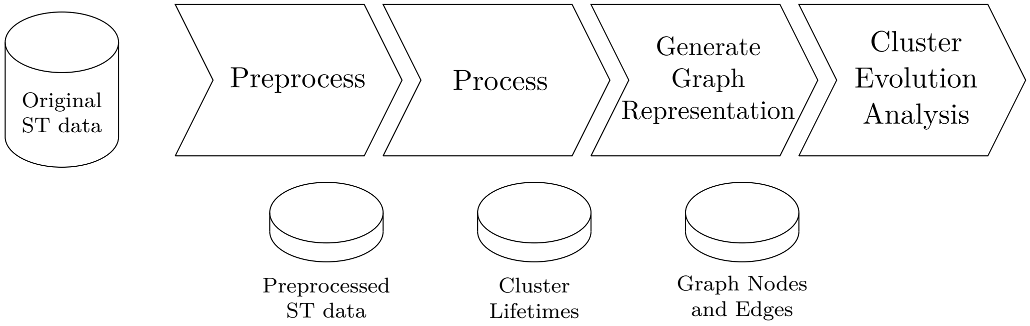


Figure 3.1: A workflow describing the steps performed by the proposed approach.

of different timestamps are discovered and linked, and spatial-temporal cluster relationships are identified. In simple terms, spatial-temporal data is clustered at each timestamp, then clusters at different timestamps are linked, forming spatial-temporal clusters, and finally spatial-temporal cluster relationships are identified. Details about these tasks are given in Sections 3.2.1 and 3.2.2. At the end of this step, each spatial-temporal cluster has a set of all spatial-temporal cluster relationships from the beginning to the end of its existence described. This set of relationships is its spatial-temporal cluster lifetime. Spatial-temporal cluster lifetimes are then stored in a repository represented in Figure 3.1 by the entity called "Cluster Lifetimes".

The third step is to generate a graph representation. Spatial-temporal cluster lifetimes of all spatial-temporal clusters are processed to create a graph representation, in which it is possible to identify the evolution of a spatial-temporal cluster based on the spatial-temporal cluster relationships it has. Details about the graph representation are given in Section 3.2.3. At the end of this step, a graph with its nodes and edges is created and stored in a repository represented in Figure 3.1 by the entity called "Graph Nodes and Edges".

The fourth step is to perform cluster evolution analysis. There are many analytical techniques that can be used on graphs for analysis, ranging from visualization tools or calculating relevant metrics, to classification. In this step, analysis methods are used to generate results. Details about the analysis methods used on the structure of spatial-temporal clusters are given in Section 3.3.1, while those on the relationships of clusters are given in Section 3.3.2. Details about the analysis methods on the graph-based representation of cluster evolution are given in 3.3.3. The result of this step is a set of analysis results that are stored in a repository represented in Figure 3.1 by the entity called "Analysis Results".

Notice that spatial-temporal data resides in the repository at the beginning of the workflow to be processed. This means that the entire dataset has to be available before the workflow describing the solution is started. This need not be the case. An alternative, more continuous, strategy can be adopted, since the solution takes only data about two timestamps at a time to process. However, this strategy is not adopted in the solution because the primary goal is to provide representation, analysis, and implementation tools for spatial-temporal cluster evolution. The continuous strategy, with stream data, available in real time is discussed as future work.

Notice also that to represent clusters and identify relationships, the solution needs only the latitude and longitude coordinates of the location of each spatial-temporal object and its identification. Other types of information about spatial-temporal objects or the clusters they form, such as speed, number of passengers, purpose of trip, or other information is passed along the cluster to the third step for analysis tasks. This means that higher dimensional datasets can still be processed with little to no additional accommodations required.

The implementation tools to support the tasks of representing and analyzing graph-based spatial-temporal cluster evolution are not included in Figure 3.1. However, they are detailed in Sections 3.4.1 and 3.4.2.

The source code of the approach is available at https://git.uwaterloo.ca/ivens/phd.

3.2 Representation

This section describes the representation of spatial-temporal clusters with respect to their structure and relationships. The section also describes the representation of a connected, graph-based cluster evolution.

3.2.1 Structure

Spatial-temporal clusters differ from other types of clusters in that they exist during a period of time. Time is a continuous dimension. Devices, such as those acquiring spatial coordinates GPS, do not acquire data continuously. Instead, these devices acquire data at discrete time steps, associating a timestamp with each reading. Therefore, spatial-temporal clusters really exist at several timestamps. Because of this discrete acquisition of position, identifying a spatial-temporal cluster means that an approach has to first identify the cluster in several timestamps and then link the many clusters creating a complete representation of the spatial temporal cluster. This task is challenging because spatial-temporal clusters change their structure, such as location or size, between timestamps. To solve this problem of change, a similarity function is used. The following subsections give more detail about how the proposed approach identifies and represents spatial-temporal clusters.

One note about the notation used. In the subsections and sections that follow, examples of spatial-temporal clusters moving in space and time are given. These examples follow a notation that is explained now for better understanding. A spatial-temporal cluster is represented by the simplified notation c_i. This refers to the spatial-temporal cluster as a whole and is irrespective of the timestamps during which the cluster exists. Examples are c_1, c_2, or even c_i itself for a generic spatial-temporal cluster. When appropriate, a reference to a particular timestamp uses the notation t_j. A timestamp is really a slice of the continuous time. Examples are t_1, t_2, or even t_j itself for a generic timestamp. Lastly, since spatial-temporal clusters exist and change during many timestamps, sometimes it is required to refer to the existence of a cluster during a particular timestamp. For that end, the notation c_{i,t_j} is used. Examples are c_{1,t_1} for the first cluster during the first timestamp or c_{2,t_2} for the second cluster during the second timestamp. Note that t_j can be used as a basis timestamp, and other timestamps can refer to it, such as t_{j-1} and t_{j+1} for the previous and next timestamps. Thus, the notation for a spatial-temporal cluster during a specific timestamp is updated accordingly. A $c_{i,t_{j-1}}$ refers to the spatial-temporal cluster c_i during the previous timestamp and $c_{i,t_{j+1}}$ refers to spatial-temporal cluster c_i during the next timestamp.

Clustering Technique

Identifying and representing a spatial-temporal cluster starts with the choice of a clustering algorithm. In general, different clustering algorithms produce different clusters with their own characteristics, such as shape, density, rules for accepting data points, or hierarchies.

Clusters are groups of objects and are, usually, assumed to have a circular shape. For instance, a flock of wild animals walking together to protect themselves from predators is somewhat circular. However, clusters can assume any shape. For instance, a flock of migrating birds sometimes assume a V-shape or are linear. Clusters that are constructed from generic data points tend to assume any shape because data points are not limited by geographical barriers, like animals are. However, data points of other types, such as spatial-temporal data, are limited by geographical barriers and must conform to the space available around them, thus influencing the shape of the clusters.

Several clustering approaches, such as the popular centroid-based clustering method K-Means [82, 43], do not work well with clusters of arbitrary shape [55]. The reason is that they assume that data points cluster around a central point, a centroid, making it difficult to identify clusters in other formats, such as clusters in a V-shape or concentric clusters, especially when they are close to other clusters.

Spatial-temporal data usually comes from real life entities, such as cars, buses, or people. These entities cluster based on the real-life limitations. For example, a traffic jam is a cluster of cars and is limited by the shape of the road. People in a train form a cluster that is limited by the elongated shape of the train. It is not reasonable to expect that a traffic jam or a train will have a circular shape, or that cars in a traffic jam or people in a train cluster around a centroid. Therefore, spatial-temporal data, can group into clusters of many different shapes and an appropriate clustering method is necessary to identify clusters of spatial-temporal data. For this reason, centroid-based clustering techniques, such as K-Means are not used in the solution.

In summary, the workflow is as follows. Given a specific dataset, the solution preprocesses the data to standardize it, making sure that dates and latitude and longitude are formatted as expected. To process data, the solution queries the preprocessed dataset at regular intervals, usually 1 minute, called timestamps, and performs DBSCAN. This is done to identify clusters at each timestamp. The solution links clusters of consecutive timestamps using a similarity metric and forms spatial-temporal clusters. The solution proceeds to identify relationships between these clusters. After that, the solution persists a list of events for each cluster, called cluster lifetimes. The solution then generates a graph representation of all clusters for analysis, which is passed to an appropriate graph data platform. Analyses are performed and results are stored in a dataset.

Often, spatial-temporal objects in a space, such as vehicles in a city, do not follow a distribution. Different from the number or the movement of planets or atoms, for example, the number of taxis grouping at certain locations or the movement of taxis in the city are unique in a day and may not repeat a specific pattern. This is because each passenger has a different reason to move and a different destination or city events happen at different times impacting the traffic flow in unpredictable ways. This undermines the use of a distribution-based clustering technique in the solution, as distributions are not suitable to describe the movement of spatial-temporal objects. For this reason, distribution-based clustering techniques, such as EM are not used in the solution.

Clusters of spatial-temporal objects and their relationships can be observed from many different perspectives. For example, special characteristics of a transportation service can differentiate a cluster of taxis from a cluster of ridesharing cars. Novel relationships can be identified when these characteristics are taken into consideration. For example, a taxi that enters a cluster of ridesharing cars can be different from a taxi that enters a cluster of taxis. This means that, in a higher perspective, every vehicle is a spatial-temporal object, but in a fine-grained perspective, the difference between taxis and ridesharing cars is made explicit. However, more complicated representation and analysis techniques are required. For this reason, hierarchical clustering techniques, either divisive or agglomerative, are not used in the solution.

DBSCAN [40] is a density-based clustering method that identifies clusters based on their proximity, or density, irrespective of a centroid. DBSCAN examines the number of data points that exist in the neighborhood of a chosen data point, effective analyzing the density around the chosen data point. The DBSCAN analysis method then can identify clusters of arbitrary shape because it does not assume that data points cluster around a centroid.

These characteristics of the DBSCAN clustering method support its use in the proposed approach.

There also exists an extended version of DBSCAN for the spatial-temporal domain called ST-DBSCAN. ST-DBSCAN [14] is a spatial-temporal density-based clustering algorithm that identifies clusters based on the similarity of their space and time dimensions. However, ST-DBSCAN acts on the entire dataset of spatial-temporal data points, which results in substantial memory use. DBSCAN, on the other hand, acts on the slice of the dataset defined by the timestamp under investigation. This reduces the memory requirements and eliminates the use of ST-DBSCAN as the clustering method used in the approach. The unavailability of ST-DBSCAN, and availability of DBSCAN, on one of the main Python libraries for data analysis is another main reason the latter method is chosen in the solution.

Therefore, the first step to represent a spatial-temporal cluster, that exists throughout several timestamps, is to use DBSCAN at each timestamp. Figure 3.2 illustrates this process. The Figure shows slices of time, the timestamps, from 1 to n. In each slice, the Figure shows clusters that exist at that slice. The step just discussed is to use DBSCAN on each of these time slices to identify the clusters that exist in them.

DBSCAN uses two parameters: ε for the radius of the neighborhood of a point and *minPts* to classify a given point, here called *min_cluster*. The first parameter, ε, is used to define the size of the neighborhood of a point. The value of this parameter is domain-specific, but, in the context of spatial-temporal objects, it is reasonable to expect the neighborhood around a person to be a few meters and the neighborhood around a car to be a few tens of meters. The second parameter, *min_cluster* represents the minimum

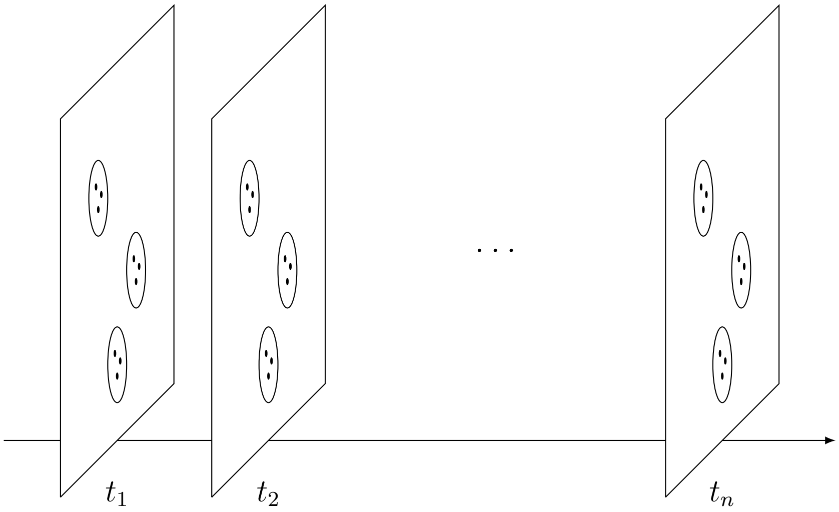


Figure 3.2: Slices of time, the timestamps, in which DBSCAN is used to identify clusters.

number of points in the neighborhood of a point, including itself, for it to be classified as a core point or an outlier. The value of this parameter is domain-specific or even specific to the task at hand, given size limitations. It is not reasonable to expect to have 50 people within 1 meter from a person, or 500 cars within 1 meter from a car. Since DBSCAN is a density-based clustering method, it uses these two parameters, ε and $min\_cluster$, to calculate the density around a data point. A high-density data point and the points in its neighborhood are a candidate to form a cluster, if they are not part of a cluster already. A low-density point is a candidate to not take part in any cluster and be deemed an outlier.

Once clusters at consecutive timestamps have been identified, the approach then has the task of linking appropriate clusters to form a spatial-temporal cluster. This new task is explained in the next subsection.

Between-Timestamp Cluster Similarity

Once clusters have been identified at each timestamp, the approach attempts to link them, forming a true spatial-temporal cluster. The identification of clusters is done for each consecutive pair of timestamps and, therefore the linkage of clusters happens at the same time. Figure 3.3 illustrates this process. Cluster c_i exists from timestamp t_1 to timestamp t_n. At timestamp t_j, the approach has identified three clusters, c_{i,t_j} and two others. At timestamp t_{j+1}, the approach has identified three clusters, $c_{\alpha,t_{j+1}}$, $c_{\beta,t_{j+1}}$, and $c_{\gamma,t_{j+1}}$. The task is to identify which of the three clusters at timestamp t_{j+1} is the continuation of cluster

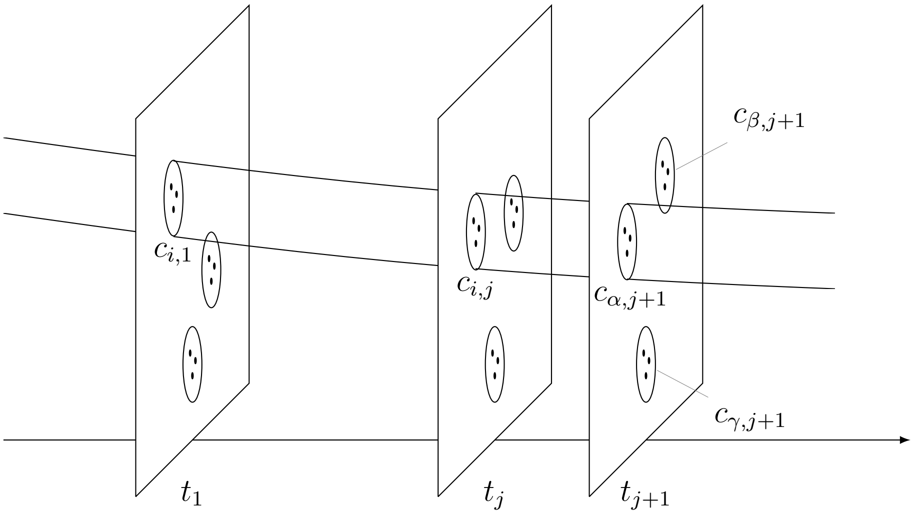


Figure 3.3: The identification of a spatial-temporal cluster in consecutive timestamps.

c_i. The task seems trivial when having the location of clusters and if they are distant from each other, but since spatial-temporal clusters enter, leave, merge with, or split into other clusters, their locations are frequently very similar. The solution to the problem of identifying the same spatial-temporal cluster in different timestamps used in the approach is a similarity metric. The similarity metric is an extension of Jaccard similarity [58] and is based on the spatial-temporal objects that clusters have at each timestamp. Once similarities have been calculated they are compared with a similarity threshold. Similarity values equal or greater than a certain threshold mean that the clusters they relate to are the same spatial-temporal cluster. Therefore, the approach links clusters accordingly.

The Jaccard similarity [58] is a similarity metric used to calculate the similarity between two sets of objects. The calculation is based on the number of shared objects in each set. Let A and B be two sets of objects. Therefore, the Jaccard similarity is calculated according to Equation 3.1, where $|\cdot|$ calculates the number of objects in a set.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

Consider the situation illustrated on the left-hand part of Figure 3.4. Cluster c_{1,t_j} contains spatial-temporal objects 1 through 6 and is to be compared with cluster $c_{\alpha,t_{j+1}}$, which exists one timestamp later and contains spatial-temporal objects 2 through 7. The objective is to check if they are the same spatial-temporal cluster. Let the sets of the

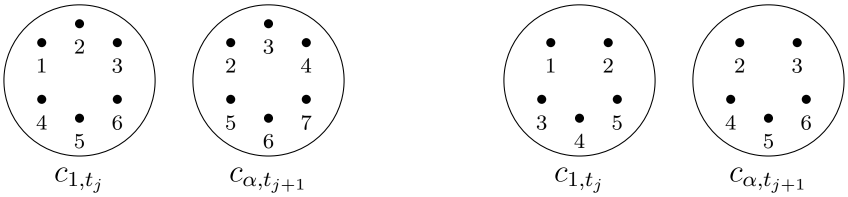


Figure 3.4: Two situations in which clusters are somewhat similar. The Jaccard similarity works well for the situation on the left-hand side, but not for the situation on the right-hand side.

Jaccard similarity formula be clusters and the calculation is:

$$J(c_{1,t_j}, c_{\alpha,t_{j+1}}) = \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j} \cup c_{\alpha,t_{j+1}}|} = \frac{5}{7} \approx 0.71$$

Assuming that clusters whose Jaccard similarity are equal or greater than 0.7 are the same spatial-temporal cluster, clusters c_{1,t_j} and $c_{\alpha,t_{j+1}}$ would be deemed to be the same spatial-temporal cluster by the approach.

However, Jaccard similarity does not perform well for smaller clusters. Consider the situation illustrated on right-hand side of Figure 3.4. Cluster c_{1,t_j} contains spatial-temporal objects 1 through 5, one less than in the previous situation, and is to be compared with cluster $c_{\alpha,t_{j+1}}$, which exists one timestamp later and contains spatial-temporal objects 2 through 6, one less that in the previous situation as well, to check if they are the same spatial-temporal cluster. The Jaccard similarity calculation is:

$$J(c_{1,t_j}, c_{\alpha,t_{j+1}}) = \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j} \cup c_{\alpha,t_{j+1}}|} = \frac{4}{6} \approx 0.67$$

Assuming a similarity threshold of 0.7, clusters c_{1,t_j} and $c_{\alpha,t_{j+1}}$ would be deemed different spatial-temporal clusters by the approach, although they share a significant number of spatial-temporal objects. One simple solution is to lower the similarity threshold to 0.65, but doing this has other implications. For example, if a similar situation is under analysis, in which clusters had two less spatial-temporal objects than the original situation, their Jaccard similarity would be approximately 0.6. This raises a question about lowering the threshold again to accept even more situations. In addition, lowering the threshold leads to accepting cluster links that are not supposed to happen.

This problem is solved with an extended similarity calculation. In fact, that are two similarity values S_1 and S_2 and only if both of them are equal to or greater than a certain similarity threshold, are the two clusters deemed to be the same spatial-temporal cluster. Equation 3.2 shows how S_1 and S_2 are calculated for sets. A discussion using the examples of Figure 3.4 follows.

Let A and B be two sets. The new similarity values S_1 and S_2 are calculated as Equation 3.2 shows, where $|\cdot|$ counts the number of objects in a set.

$$S_1(A,B) = \frac{|A \cap B|}{|A|} \qquad \text{and} \qquad S_2(A,B) = \frac{|A \cap B|}{|B|} \tag{3.2}$$

Consider again the situation presented on the left-hand side of Figure 3.4. The new similarity values are $S_1 = S_2 \approx 0.83$. Since both S_1 and S_2 are greater than the assumed similarity threshold of 0.7, the clusters are deemed to be the same spatial-temporal cluster by the approach.

$$\begin{aligned}
S_1(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j}|} = \frac{5}{6} \approx 0.83 \quad \text{and} \\
S_2(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{\alpha,t_{j+1}}|} = \frac{5}{6} \approx 0.83
\end{aligned}$$

Now consider the situation presented on the right-hand side of Figure 3.4. The new similarity values are $S_1 = S_2 = 0.8$. Since both S_1 and S_2 are greater than the assumed similarity threshold of 0.7, the clusters are deemed to be the same spatial-temporal cluster by the approach. Notice that the new similarity metric is more stable than the Jaccard similarity for smaller values. A small change in the number of spatial-temporal objects in the cluster did not perturb the final similarity values significantly.

$$\begin{aligned}
S_1(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j}|} = \frac{4}{5} = 0.8 \quad \text{and} \\
S_2(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{\alpha,t_{j+1}}|} = \frac{4}{5} = 0.8
\end{aligned}$$

As a final example and to see how the Jaccard similarity and the new similarity metric differs, consider the situation illustrated in Figure 3.5. Cluster c_{1,t_j} contains spatial-temporal objects 1 through 6 and is to be compared with cluster $c_{\alpha,t_{j+1}}$, which exists one

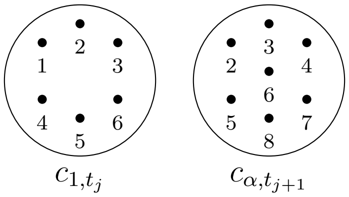


Figure 3.5: A situation in which a cluster loses one spatial-temporal object, wins other two, but Jaccard similarity fails to consider them as the same cluster, and the similarity values S_1 and S_2 succeed.

timestamp later and contains spatial-temporal objects 2 through 8, to check if they are the same spatial-temporal cluster. The Jaccard and the S_1 and S_2 similarity values are calculated below.

$$J(c_{1,t_j}, c_{\alpha,t_{j+1}}) \;= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j} \cup c_{\alpha,t_{j+1}}|} = \frac{5}{8} = 0.625$$

$$\begin{aligned} S_1(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{1,t_j}|} = \frac{5}{6} \approx 0.83 \quad \text{and} \\ S_2(c_{1,t_j}, c_{\alpha,t_{j+1}}) &= \frac{|c_{1,t_j} \cap c_{\alpha,t_{j+1}}|}{|c_{\alpha,t_{j+1}}|} = \frac{5}{7} \approx 0.71 \end{aligned}$$

These results show that, assuming a similarity threshold of 0.7, using Jaccard similarity, the clusters are deemed different spatial-temporal clusters. The conclusion is different if the proposed similarity value is used. Both S_1 and S_2 are greater than the assumed similarity threshold of 0.7, and therefore clusters are deemed the same spatial-temporal cluster.

Note that the original cluster c_{1,t_j} loses one spatial-temporal object and wins two others. This indeed is not enough to state that a new cluster has started. Note also that situations where spatial-temporal clusters lose or win spatial-temporal objects are frequent during cluster evolution analysis. Therefore, choosing the right similarity value impacts the final result.

The current method to identify spatial-temporal cluster relationships takes into consideration only the membership of a data point to a cluster. This means that the method checks only whether a data point belongs or not to a cluster. A more advanced method to identify spatial-temporal cluster relationships take into consideration the type of data points: core points, border points, or noise points. This impacts the Jaccard-based similarity calculation as well as the similarity threshold. Although this method is not the one used in the solution, further investigations are encouraged so that new relationships can be identified.

Once the structures of spatial-temporal clusters are built, that is the several parts of the clusters are identified and linked, the approach can then start to identify the relationships these clusters have with spatial-temporal objects and other spatial-temporal clusters. This is explained in the next section.

3.2.2 Relationships

This section discusses the representation of spatial-temporal cluster relationships. It starts with an explanation of what these relationships are, then it describes the rules used to identify these relationships, and ends with the 14 relationships that the approach can identify.

Spatial-Temporal Cluster Relationships

Spatial-temporal cluster relationships are the interactions spatial-temporal clusters have with other clusters or spatial-temporal objects. For example, if a spatial-temporal object leaves a spatial-temporal cluster, a relationship takes place. Similarly, if a spatial-temporal cluster leaves another cluster, another relationship takes place. For a visual illustration, consider the situation described in Figure 3.6. The figure shows two spatial-temporal clusters c_1 and c_2 approaching each other during the first two timestamps. But the figure shows only one cluster during timestamp t_3. The reason is that from timestamp t_2 to timestamp t_3, spatial-temporal clusters c_1 and c_2 merged to form c_3. Figure 3.6 shows a MERGE relationship.

Identifying and representing spatial-temporal cluster relationships is challenging because they happen between timestamps, where cluster structure properties such as the number of contained spatial-temporal objects change. Another reason for why this task is challenging is the similarity between these relationships. As an example, consider again the MERGE relationship shown in Figure 3.6. Another interpretation of this relationship is that cluster c_2 entered cluster c_1, producing another spatial-temporal relationship called C_ENTER. This highlights the need for clear rules that differentiate spatial-temporal cluster relationships.

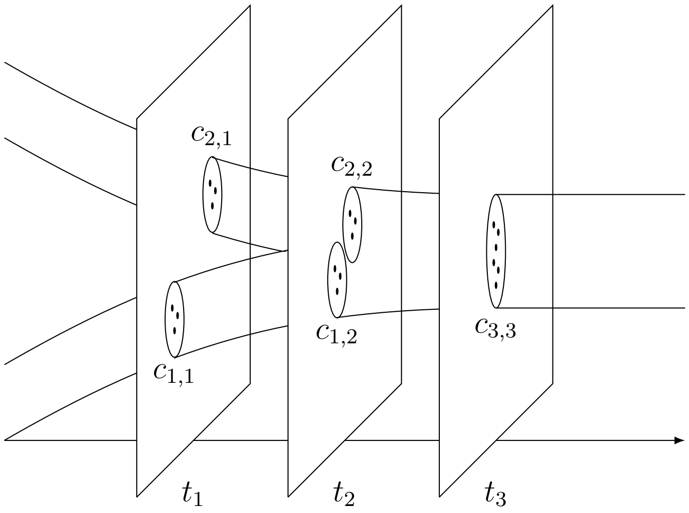


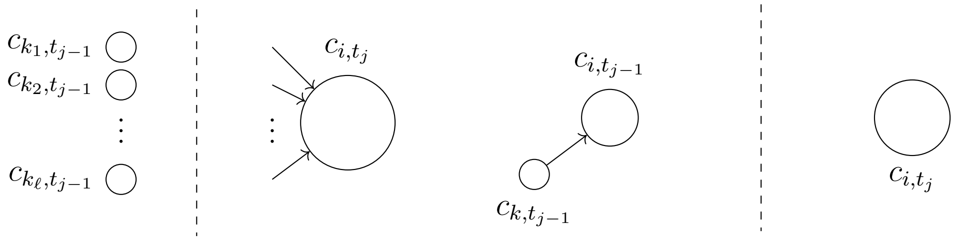
Figure 3.6: A MERGE spatial-temporal cluster relationship.

The proposed approach uses rules, similar to first-order logic rules, to describe the conditions necessary for a spatial-temporal cluster relationship to be identified and represented. The rules are described in the form of one or more statements, all of which need to be true for a spatial-temporal cluster relationship to be present. Spatial-temporal clusters, like c_i are assumed to be sets of spatial-temporal objects.

As an example, consider the rules to identify and represent a MERGE relationship. A visual representation is presented in Figure 3.7a. A cluster c_i undergoes a MERGE relationship from timestamp t_{j-1} to timestamp t_j if

- $c_{i,t_{j-1}} = \varnothing$;
- $c_{i,t_j} \neq \varnothing$;
- there exists at least two clusters $c_{k,t_{j-1}}$ and $c_{\ell,t_{j-1}}$, $k \neq i$, $\ell \neq i$, such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$ and $|c_{i,t_j} \cap c_{\ell,t_{j-1}}| \geq min\_cluster$.

The two first conditions simply assert that c_i does not exist at timestamp t_{j-1} and exists at timestamp t_j. The third condition has two parts. It first asserts that c_k and c_ℓ are different from c_i based on the similarity metric described in the previous subsection, and it also asserts that clusters c_k and c_ℓ share a minimum number of spatial-temporal objects with c_i. This minimum number is $min\_cluster$, described in subsection 3.2.1 as one parameter for DBSCAN. This is done to verify that the clusters are indeed participating in a relationship.



(a) The MERGE relationship.

(b) The C_ENTER relationship.

Figure 3.7: A visual representation of spatial-temporal relationships MERGE and C_ENTER.

For comparison purposes, consider the rules to identify and represent a C_ENTER relationship. A visual representation is presented in Figure 3.7b. A cluster c_i undergoes a C_ENTER relationship from timestamp t_{j-1} to timestamp t_j if

- $c_{i,t_{j-1}} \neq \varnothing$;
- $c_{i,t_j} \neq \varnothing$;
- there exists a cluster $c_{k,t_{j-1}}$, $k \neq i$, such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$;
- $c_{k,t_j} = \varnothing$.

The two first conditions simply assert that c_i exists at timestamps t_{j-1} and t_j. This is somewhat different from the representation of a MERGE relationship that required a cluster to exist only in timestamp t_j. The third condition also has two parts. It first asserts that c_k is different from c_i based on the similarity metric described in the previous subsection, and it also asserts that the cluster c_k shares a minimum number of spatial-temporal objects with c_i. A fourth condition asserts that cluster c_k does not exist in timestamp t_j.

Note the importance of the similarity function in the identification of relationships. One major difference between the MERGE and the C_ENTER relationships is the similarity of cluster c_{i,t_j} with the clusters in the previous timestamp. If the approach, based on the similarity value, deems that c_{i,t_j} is the same spatial-temporal cluster than one of the clusters in the previous timestamp, then the approach identifies and represents a C_ENTER relationship because cluster c_i already existed. Conversely, if the approach, based on the similarity value, deems that c_{i,t_j} is *not* the same spatial-temporal cluster than *any* of the clusters in the previous timestamp, then the approach identifies and represents a MERGE relationship because c_i just started to exist.

The previous discussion compared the rules to identify and represent two spatial-temporal cluster relationships. They were chosen because of their similarity. However, there are more relationships and therefore more rules. A full list of rules for spatial-temporal cluster relationships can be found in [106] and is also available in Appendix A.

The proposed approach identifies and represents 14 spatial-temporal cluster relationships. Table 3.1 contains all relationships with a small description about each one. One important consideration is that these relationships are used to build a spatial-temporal cluster lifetime. Therefore, a relationship such as C_ENTER must be recorded in the cluster lifetime of both participating clusters. However, the approach differentiates the relationship being recorded based on the perspective of the cluster from which it is being described. For instance, suppose cluster c_2 enters cluster c_1. Recording the relationship C_ENTER on both cluster lifetimes will lead to ambiguous interpretations about which cluster entered which. Instead, the approach records C_ENTER and JOIN based on how the relationship took place.

The literature contains the description of a MOVE relationship [124]. This relationship describes the event where a cluster changes one of its properties. For example, if the location of a cluster changes from one timestamp to another, a MOVE relationship takes place. The proposed approach does not consider the MOVE relationship for two reasons. The first reason is the lack of interaction between a spatial-temporal cluster and another cluster or object. The second reason is the large number of relationships created, since it is expected that one MOVE relationship happens between each pair of timestamps. If cluster properties, such as location, are to be analyzed, they can be accessed directly instead of using a relationship for it.

3.2.3 Graph

This section discusses the graph-based representation of spatial-temporal cluster evolution. It starts with a discussion about the types of graphs and how clusters and relationships can be represented in such structure. The section ends with some details about how the conversion between formats can be addressed.

Graph Transformation

A graph-based representation of cluster evolution is one in which spatial-temporal clusters are represented from their start to their end and have their relationships connecting them at suitable timestamps. An example is shown in Figure 3.8. If the spatial-temporal

Table 3.1: The 14 spatial-temporal cluster relationships identified and represented by the proposed approach.

Relationship	Description
START	The spatial-temporal cluster begins to exist.
END	The spatial-temporal cluster ceases to exist.
GROUP	A number of spatial-temporal objects assemble to form a spatial-temporal cluster.
DISPERSE	A spatial-temporal cluster disassembles in a number of spatial-temporal objects.
T_ENTER	The spatial-temporal cluster accepts a spatial-temporal object.
T_LEAVE	The spatial-temporal cluster releases a spatial-temporal object.
C_ENTER	The spatial-temporal cluster accepts a cluster. This relationship is written from the perspective of the receiving cluster.
C_LEAVE	The spatial-temporal cluster releases the cluster. This relationship is written from the perspective of the expelling cluster.
JOIN	The spatial-temporal cluster enters a cluster. This relationship is written from the perspective of the incoming cluster.
DETACH	The spatial-temporal cluster leaves a cluster. This relationship is written from the perspective of the outgoing cluster.
MERGE	Two or more spatial-temporal clusters assemble to form a cluster.
SPLIT	A spatial-temporal cluster divides into two or more clusters.
C_IN	Two spatial-temporal clusters transfer a considerable number of spatial-temporal objects, enough to form a cluster, but quick enough for this cluster to not exist. This relationship is written from the perspective of the receiving cluster.
C_OUT	Two spatial-temporal clusters transfer a considerable number of spatial-temporal objects, enough to form a cluster, but quick enough for this cluster to not exist. This relationship is written from the perspective of the expelling cluster.

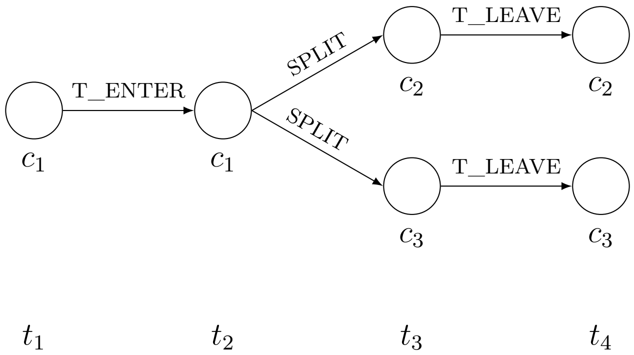


Figure 3.8: The connected, graph-based representation of spatial-temporal cluster evolution.

cluster c_1 undergoes a SPLIT relationship, forming spatial-temporal clusters c_2 and c_3, the graph-based representation shows the beginning of cluster c_1, the SPLIT relationship as a connection between cluster c_1 and clusters c_2 and c_3, which is also the end of cluster c_1, and finally the end of clusters c_2 and c_3. Notice that the representation of the structure and the relationships of spatial-temporal clusters are now connected. The structure chosen for this connected representation is a graph.

A graph is a mathematical structure that represents the connection between objects [31]. Formally, a graph is a pair $G = (V, E)$ where V is a set of vertices and E is a set of paired vertices, called edges. Graphs can be undirected or directed. In undirected graphs, edges do not have a direction and are represented by lines. In directed graphs, edges do have a direction and are presented by arrows. A cycle in a graph is a sequence of vertices connected by edges such that only the first and the last vertices repeat. In some graphs, a vertex has a connection with itself, which creates an edge that starts and ends on the same vertex. This is called a loop. Loops are cycles, but not all cycles are loops. A Directed Acyclic Graph (DAG) is a graph whose edges are directed and does not contain a cycle. DAGs are important and useful in the representation and organization of flows.

To have a graph-based representation of cluster evolution, the approach uses a directed graph that can include loops. This graph is directed because it represents cluster evolution in time, which creates a temporal order between vertices. It sometimes includes loops because of limitations on the representation of several spatial-temporal cluster relationships between timestamps. The graph used to represent spatial-temporal cluster evolution includes loops and, for that reason, is not a DAG.

One note about the terminology. Graph vertices are also called nodes. Since this is a valid term, and this is the term used by the graph platform used in this approach, the description that follows also uses the term nodes to refer to graph vertices.

The approach has the task of turning spatial-temporal cluster lifetimes into a graph format, that is reading a list of spatial-temporal cluster relationship for each cluster, processing these many lists, and producing a set of nodes and edges. The process has two main foci, one for nodes and one for edges. When a spatial-temporal cluster lifetime is ready to be processed, each relationship is inspected for nodes and edges, such inspection may generate nodes, edges, or both. For example, the spatial-temporal relationship T_ENTER, when inspected for nodes, generates two nodes, one for the previous timestamp and one for the current timestamp. The structure of the cluster changes because the cluster has a new spatial-temporal object. This is reflected on these nodes. The same relationship, T_ENTER, when inspected for edges, generates one edge, connecting the nodes for this cluster in the previous and current timestamps.

The example just discussed illustrates the process of transforming one spatial-temporal relationship into a set of nodes and edges for a graph-based cluster evolution. The approach can transform all of the fourteen spatial-temporal relationships described in subsection 3.2.2, including more complex relationships such as MERGE and SPLIT that involve at least three clusters. When spatial-temporal clusters are represented, including their structure and relationships, and a graph-based representation is constructed, the approach turns to the analysis of spatial-temporal cluster evolution. This is detailed in the next section.

3.3 Analysis

This section describes analysis methods that can be used on the structure of clusters or on their relationships to produce novel results. This section also describes analysis methods for a graph-based representation of cluster evolution. Analyses on the representation are at the core of the approach described here.

3.3.1 Structure

The analysis of the structure of spatial-temporal clusters is based on the changes to the clusters over time. The analysis is on a single cluster, from start to end, and cluster relationships are not considered.

Changes to the structure of spatial-temporal clusters are usually related to their size or location. For example, an increase or decrease of the number of spatial-temporal objects

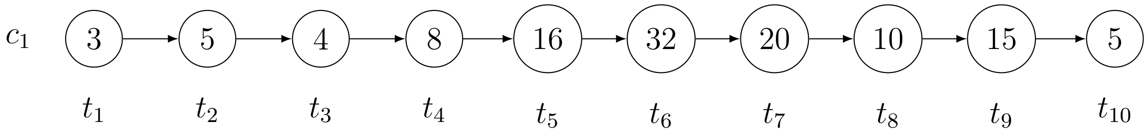


Figure 3.9: Analysis on a spatial-temporal cluster that has its structure changed with time.

contained in the cluster, the cluster size, are changes to the structure of the cluster. In addition, alterations to the location of the cluster, based on latitude and longitude coordinates, because of its movement are also changes to the structure of the spatial-temporal cluster.

These changes happen from one timestamp to another and are relatively easy to detect. However, analyzing long-term changes that span several timestamps is more challenging, but more valuable.

Describing an extensive list of analysis methods is not the goal of this subsection, but it includes some possible options. Examples of analysis methods for the structure of spatial-temporal clusters include the detection of growing or decaying temporal regions, the calculation of a growth or decay factors, and inferences about the movement of a cluster, such as its direction or time of the day.

For instance, consider a cluster c_1 that has its number of contained spatial-temporal objects, or size, changed with time according to Figure 3.9. Notice that the figure shows a spatial-temporal cluster throughout several timestamps having its number of contained spatial-temporal objects changed. In addition, notice that its size is shown in the middle of the cluster. Analysis of the structure of the spatial-temporal cluster based on its size can identify a temporal region of growth and a temporal region of decay of spatial-temporal objects. The growth temporal region starts at timestamp t_3 and ends at timestamp t_6 and the decay temporal region starts at timestamp t_6 and ends at timestamp t_8, as illustrated in Figure 3.9. Depending on the interpretation of the analysis method, growth and decay temporal regions can admit some small decreases or increases inside the region. A growth or decay parameter can also be calculated based on the size of the cluster at its start and end. Figure 3.9 shows the value of these parameters for the regions identified in the example.

3.3.2 Relationships

The analysis of the relationships of spatial-temporal clusters is based on the occurrences of the relationships with time. The analysis is focused on a single cluster, from start to end,

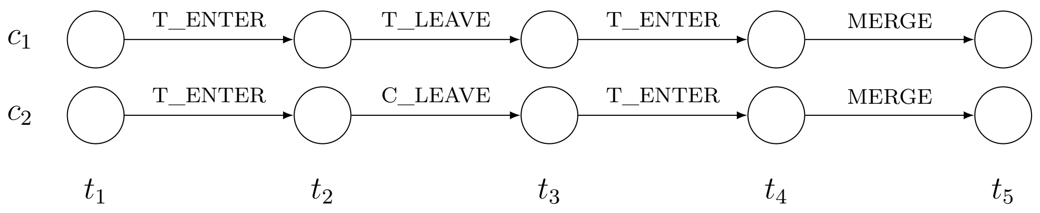


Figure 3.10: Two spatial-temporal clusters can be compared for similarity based on their spatial-temporal cluster relationships.

although conclusions of the analysis of some relationships, such as C_ENTER or MERGE, may involve other spatial-temporal clusters.

The occurrences of spatial-temporal cluster relationships can be analyzed individually and are relatively simple, but analyzing these occurrences based on time is what generates novel results. An even more valuable analysis happens when individual results are compared.

Describing an extensive list of analysis methods is not the goal of this subsection, but it includes some possible options. Examples of analysis methods for the relationships of spatial-temporal clusters include the identification of the starting or ending relationship, the frequency of certain relationships during cluster existence, and the similarity of clusters based on their relationships.

For instance, consider Figure 3.10. The figure shows two spatial-temporal clusters c_1 and c_2 and the relationships they have in time. The other clusters that participate in these relationships are not shown as the analysis methods are concerned with just the relationships. Notice that, for each cluster, the relationships can be represented as an ordered list, or sequence, similar to a spatial-temporal cluster lifetime. There are several analysis methods for the comparison of sequences [97]. Most of these methods are based on an edit distance [97], which calculates how similar strings are, based on their letters. One such analysis method is the Wagner-Fischer algorithm [97].

3.3.3 Graph

The analysis of a graph-based representation of spatial-temporal cluster evolution is based on the structure and relationships of spatial-temporal clusters and their representation in a graph. The analysis is focused on several clusters, from start to end.

Spatial-temporal clusters move based on the movement of the spatial-temporal objects contained in the clusters. These objects move freely and are not limited by the boundaries of the spatial-temporal cluster. As a consequence, at some point in time, spatial-temporal objects may decide to enter or leave a cluster, or migrate to a new cluster. These decisions create spatial-temporal cluster relationships. Spatial-temporal cluster relationships are somewhat bound by the cluster relationship in which they take part. For example, a SPLIT relationship binds, or connects, the original clusters with the two or more new clusters. These connections can be represented in a graph, creating a connected representation for graph-based cluster evolution.

Analysis of this graph extends the analysis described in the previous subsections 3.3.1 and 3.3.2, in which only a single spatial-temporal cluster evolution is considered. Therefore, if a spatial-temporal cluster c_1 SPLITs into clusters c_2 and c_3, the analysis starts at c_1 and continues with data about c_2 or c_3. Evidently, if a spatial-temporal cluster does not have cluster relationships with any other cluster, then the analysis is limited to only this cluster.

Analysis on the graph-based representation of spatial-temporal cluster evolution can be done considering a graph path, a tree, or a subgraph. In graph theory, a path is a sequence of vertices in a graph such that every two consecutive vertices are connected by an edge, vertices are distinct, and edges are distinct. Some authors do not require that all vertices of a path be distinct. Furthermore, a directed path is a path where all edges have the same direction. Given these definitions, analysis on the graph-based representation of spatial-temporal cluster evolution considers a graph directed path, which, simply put, is a path from a vertex to another. In the context of spatial-temporal clusters, a graph directed path is a path of nodes from the occurrence of a cluster during a timestamp to the occurrence of a cluster during another timestamp, or a path from c_{i,t_j} to c_{k,t_j+m}, $m > 0$, without restrictions on i or j. Consider the situation illustrated in Figure 3.11. The figure shows two spatial-temporal clusters c_1 and c_2 that merge, forming the spatial-temporal cluster c_3. Cluster c_3 then splits, creating spatial-temporal clusters c_4, c_5, and c_6. A path from c_1 to c_6 is $(c_{1,t_1}, c_{1,t_2}, c_{3,t_3}, c_{3,t_4}, c_{6,t_5}, c_{6,t_6})$. The term path is borrowed from graph theory and is used in the graph-based cluster evolution as a spatial-temporal cluster evolution path, a cluster evolution path, or simply a path. Note, however, that not every graph directed path is a cluster evolution path, as some conditions must be met. The conditions assert that the first and last clusters in a cluster evolution path are truly starting and ending clusters. The conditions are:

i) if c_{i,t_j} is the start of a cluster evolution path, there should not be a cluster $c_{k,t_{j-1}}$ with a relationship with ci, t_j; and

ii) if c_{i,t_j} is the end of a cluster evolution path, there should not be a cluster $c_{k,t_{j+1}}$ with which c_{i,t_j} has a relationship with.

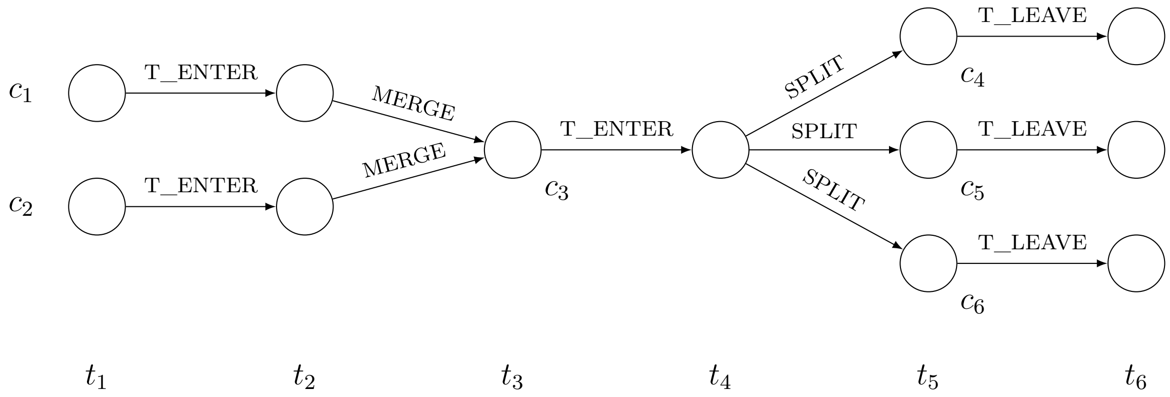


Figure 3.11: A graph-based representation of the evolution of three clusters. A path, a tree, and a subgraph are shown.

Under these definitions, the directed graph path $(c_{1,t_1}, c_{1,t_2}, c_{3,t_3}, c_{3,t_4}, c_{6,t_5}, c_{6,t_6})$ is a cluster evolution path but the graph directed path $(c_{1,t_2}, c_{3,t_3}, c_{3,t_4}, c_{6,t_5})$ is not.

Analysis on the graph-based representation of spatial-temporal cluster evolution also considers a tree. In graph theory, a tree is an undirected graph in which any two vertices are connected by a single path. Furthermore, a polytree, or directed tree is a tree in which edges have directions. In the context of spatial-temporal clusters, analysis happens in a structure similar to a tree, in which it has a clear starting point, and several end points. Consider, again, the situation illustrated in Figure 3.11. A tree can be found from the start of the spatial-temporal cluster c_3 to the end of the spatial-temporal clusters c_4, c_5, and c_6. Trees usually refer to future relationships.

Analysis on the graph-based representation of spatial-temporal cluster evolution also considers a subgraph. In graph theory, a subgraph S of a graph G is a graph whose set of vertices is a subset of the set of vertices of G and whose set of edges is a subset of the set of edges of G. In the context of spatial-temporal clusters, analysis happens in a structure that is not similar to a tree or a path, having several starting points and one or many end points. Consider, one more time, Figure 3.11. A subtree can be found from the start of the spatial-temporal clusters c_1 and c_2 to the end of the spatial-temporal cluster c_3. Another subgraph considers the entire graph. Usually, subgraphs refer to relationships of the past, but they can also refer to situations in which several staring points and end points exist.

Describing an extensive list of analysis methods for the graph-based representation of spatial-temporal cluster evolution is the goal of this subsection, but it includes some possible options. Examples of analysis methods include the identification of points of

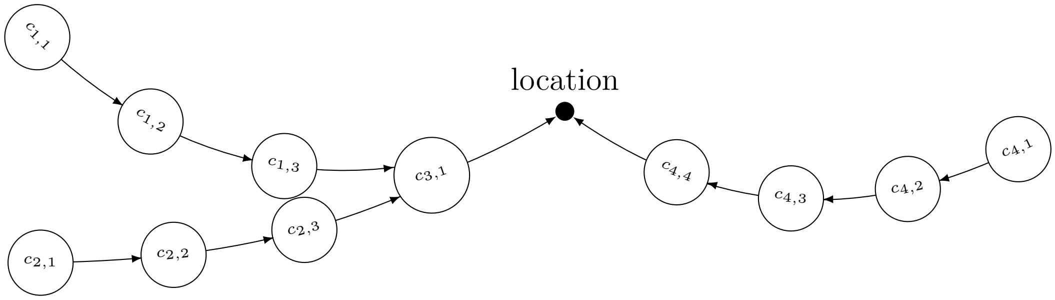


Figure 3.12: A map with the movement of spatial-temporal clusters. Demand is high at a specific location.

interest, the identification of peaks of growth or decay during spatial-temporal cluster evolution, the distribution of movement of spatial-temporal clusters based on the paths and the cardinal points or the time of the day, and the analysis of offer or demand of spatial-temporal services at specific locations.

For instance, consider the situation illustrated in Figure 3.12. The figure shows the movement of spatial-temporal clusters c_1, c_2, and c_3. The figure is a bit different from previous ones because it does not show the timestamps for the time of the movement, but this can be understood based on the occurrences of the cluster on each node of the graph. If the task at hand is to identify regions of high-demand of spatial-temporal services, such as taxis, it is reasonable to expect that these regions attract more and more spatial-temporal objects. An analysis method then can observe the movement of groups of spatial-temporal clusters by querying data about the direction of movement of clusters and identify these regions. Notice that, since regions of high-demand attract more spatial-temporal objects, it is likely that regions turn into new, large, spatial-temporal clusters that receive many other clusters. An analysis method can use this information and identify regions of high demand of spatial-temporal services by querying the graph about clusters with rapid growth. Lastly, it is expected that several cluster relationships of the type C_ENTER or MERGE happen next to regions of high demand. An analysis method can then use this information and identify regions of high demand of spatial-temporal services based on the density of these spatial-temporal cluster relationships.

The analysis of either the structure or the relationships of spatial-temporal clusters or the analysis of a graph-based representation of spatial-temporal cluster evolution is possible with the implementation tools, or the software support, that underlines it. The next section details these tools, or this support, for the representation and analysis of graph-based cluster evolution.

3.4 Implementation

This section describes the implementation tools that are developed and used in the proposed approach. The section also describes existing implementation tools that, together with the developed tools, contribute to graph-based cluster evolution. The discussion is divided into the representation and the analysis of spatial-temporal data and follows the steps that spatial-temporal data take from its raw format, through transformations and manipulations that create different representations, to its analysis and the extraction of its value. The repository containing the source code of the proposed approach is available at https://git.uwaterloo.ca/ivens/phd.

Figure 3.13 guides the discussion and shows the internal steps that data goes through. The figure has four main steps: “Preprocess”, “Process”, “Graph Transformation”, and “Graph Processing”, which are detailed in this section. The figure also shows the implementation tools used in each step to provide the software support for graph-based cluster evolution.

In Figure 3.13, a dataset of raw spatial-temporal data available for analysis is represented by the repository named “Raw Data”. From a general perspective, there is only one internal repository for data, represented by the repository named “Internal Storage”. Figure 3.13 includes entities for perspectives for analysis methods named “Graph Visualization”, “Structure”, “Relationship”, and “Graph” and “Others” as the approach allows for these tasks to be implemented and performed. Each of the four main steps are detailed in the next subsections.

3.4.1 Representation

In the proposed approach, spatial-temporal data assumes different representations. It starts with the raw format and then is preprocessed to prepare for manipulation. Data is then processed and the structure of spatial-temporal clusters are identified as well as their spatial-temporal relationships. A spatial-temporal cluster lifetime is built. Lastly, the cluster lifetime is converted to a graph format so that a connected, graph-based representation can be constructed and used for analysis. The steps of this process are described in Figure 3.13 and in the next paragraphs.

The first step is “Preprocess”, which is responsible for preprocessing spatial-temporal data. Spatial-temporal data can be described in many ways. For example, for the spatial dimension of data, latitude and longitude can be expressed in degrees, minutes, and seconds, such as `43°28'22.0''N 80°32'32.0''W`, or in decimal degrees, such as `80.542222`,

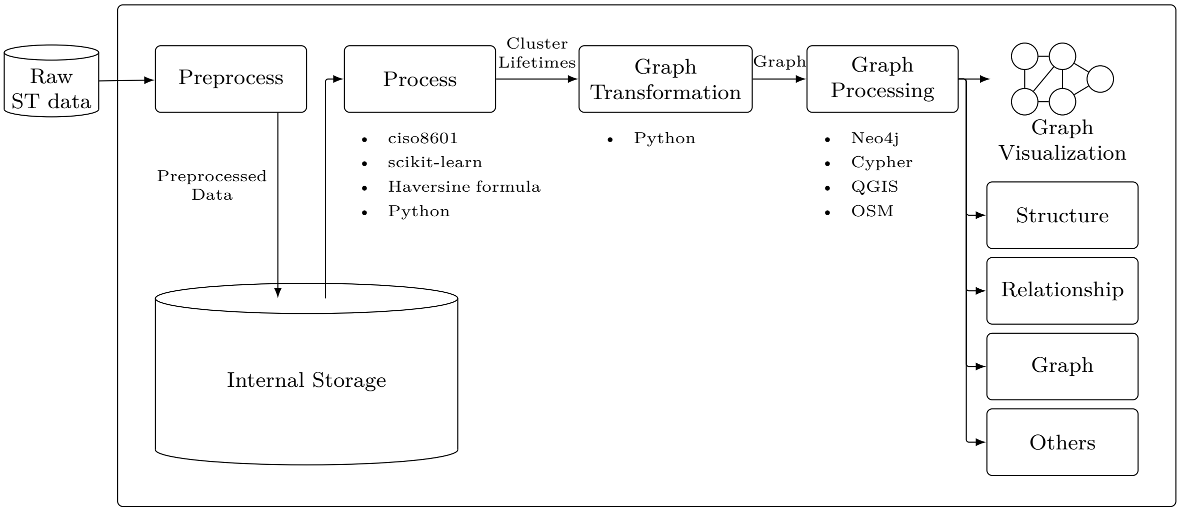


Figure 3.13: The implementation and tools used in the proposed approach and the process that spatial-temporal data goes through from its original representation to analysis and value extraction.

`43.472778`. The temporal dimension of data can be described in several different formats, including the difference between the representation of dates, `dd/mm/yyyy` or `mm/dd/yyyy`, the local time zone or the conversion to the Coordinated Universal Time (UTC), and the inclusion of the day of the week. In addition to this, spatial-temporal data can be described in either regular intervals, such as every minute, or irregular intervals. Because of the many ways spatial-temporal data can be described, in the “Preprocess” step, data coming from the “Raw Data” dataset is preprocessed such that it follows the format degrees, minutes, and seconds for the spatial dimension and the format `dd-mm-yyyy hh:mm:ss` in either the local or the UTC time zone for the temporal dimension. No treatment on the regularity of data is given in this step as it is left for the next step. The resulting preprocessed spatial-temporal data is stored in the internal storage.

The second step is “Process”, which is responsible for identifying spatial-temporal clusters and spatial-temporal cluster relationships. To query spatial-temporal data, it must be noted that the data is divided into timestamps. These timestamps can be regular, such as one measurement per minute, or irregular. Spatial-temporal data available to the approach is queried at regular intervals. If the time the query is seeking does not match with the time in the timestamp, the most recent timestamp is used, provided that it is not obsolete, *i.e.*, it has not been processed already. This condition is necessary because the approach may query data at a regular interval that is smaller than the regular interval of the actual data, for instance seeking timestamps for each minute when timestamps in data have intervals of 5 minutes. The choice of a regular interval to query data is important and the impact on the representation and analysis, its sensitivity, is dependent on the data available and the domain. For example, data produced by GPS devices in vehicles may have a 1-minute regular interval. A vehicle may enter, remain a few minutes, and leave a cluster of cars stopped at a traffic light in less than 3 minutes. In that case, choosing a query regular interval of 5 minutes makes some interactions to go unnoticed. Analysis on racing cars is different, since data is available at a smaller regular interval and clusters relationships happen constantly. A GPS device on a vessel crossing an ocean may produce spatial-temporal data at every minute, but cluster relationships in this domain take longer to happen because of the slow movement of vessels. In that case, the choice of a query regular interval of 10 minutes or more may not have significant impacts in the analysis results. Note that there is a difference between the timestamp in the spatial-temporal data and the regular interval the approach uses to query the data, which is also called a timestamp. To simplify the terminology, the remaining of this section uses the term timestamp to refer to the internal regular interval the approach uses to query data. The Python library ciso8601[1][2] is used for high-performance time manipulation in this task.

To identify spatial-temporal clusters, spatial-temporal data is queried at each regular interval, or timestamp, a clustering technique is executed on the spatial-temporal data retrieved, producing results per timestamp. A precise description of how it is done in the proposed approach is found in Section 3.2.1. Once the clustering technique has identified clusters at each timestamp, the approach links clusters that exist on a particular timestamp t_j with clusters that exist on the timestamp t_{j+1} based on the similarity of clusters. This results in the actual identification of spatial-temporal clusters. A precise description of how the similarity is calculated and the spatial-temporal cluster identification is done is found in Section 3.2.1. The Python library scikit-learn[3][4][5] for ML in Python provides the implementation of DBSCAN for clustering spatial-temporal data retrieved for a specific timestamp. The same library provides the implementation of the Haversine formula [30, 128] for distance measurements on the surface of the Earth, whose results are used by DBSCAN to calculate similarities and perform clustering tasks.

Given that parts of spatial-temporal clusters are identified, the approach can examine their interactions and identify spatial-temporal cluster relationships. This identification

[1] https://pypi.org/project/ciso8601/

[2] https://github.com/closeio/ciso8601

[3] https://scikit-learn.org

[4] https://pypi.org/project/scikit-learn/

[5] https://github.com/scikit-learn/scikit-learn

is based on rules about the number of participants of each cluster in consecutive timestamps. The proposed approach can identify 14 different types of spatial-temporal cluster relationships. A precise description of how it is done is found in Section 3.2.2. In the end, each spatial-temporal cluster can be represented by a set of spatial-temporal cluster relationships it has from its start to its end. This is the spatial-temporal cluster lifetime of a cluster.

In fact, a spatial-temporal cluster lifetime contains more important information about the relationships of spatial-temporal clusters, such as the location and time it happened, the spatial-temporal objects that entered or left the cluster, and the other spatial-temporal clusters included in the relationship, either because of merging, splitting, having a cluster entering, or leaving. Spatial-temporal lifetimes are represented in simple text format and are human-readable. Listing 3.1 shows one such spatial-temporal cluster lifetime. The listing shows the lifetime of a spatial-temporal cluster c_1. This is the reason why every line in Listing 3.1 ends in 1. Note that each line starts with the date, the time, and the location, represented by its latitude and longitude coordinates, of the moment when or place where each spatial-temporal cluster relationship happened. Each line also contains the name of the relationship followed by a pair of parentheses. The content between the parentheses is based on the relationship. If the relationship is a T_ENTER, which is when a spatial-temporal object enters the cluster, then the content between the parentheses identifies the spatial-temporal object, followed by a zero, to indicate that the spatial-temporal object did not come from any other cluster, followed by the identification of the cluster. If the relationship is a C_ENTER, which is when a cluster receives another cluster, then the content between the parentheses identifies the spatial-temporal objects that entered the cluster with a notation using square brackets, followed by the identification of the entering spatial-temporal cluster, followed by the identification of the receiving spatial-temporal cluster. In Listing 3.1, cluster c_1 started with the grouping of spatial temporal objects 100, 101, 102, and 103. A minute later, spatial-temporal object 108 also enters the cluster. Notice that at 14:03:00 two spatial-temporal objects, namely 109 and 110, enter the cluster at the same time. Then spatial-temporal object 103 leaves the cluster. At 14:05:00, cluster c_2, containing spatial-temporal objects 155, 156, and 157, enters cluster c_1. This is followed by the spatial-temporal objects 108, 109, and 110, leaving the cluster at the same time and forming a new spatial-temporal cluster c_3. Cluster c_1 ends its existence with a SPLIT relationship, creating spatial-temporal clusters c_4 and c_5.

Each spatial-temporal cluster has its own cluster lifetime, which is stored in the internal storage. These files contain important information about the existence of all clusters and are the starting point for an improved representation. Because of this, they are then passed on to the next step.

Listing 3.1: Spatial-temporal cluster lifetime of a given spatial-temporal cluster.

```
2023-07-24 14:00:00 043.473026 -080.541530 start(1)
2023-07-24 14:00:00 043.473026 -080.541530 group([100, 101, 102, 103], 1)
2023-07-24 14:01:00 043.472454 -080.540848 t_enter(108, 0, 1)
2023-07-24 14:03:00 043.468978 -080.539810 t_enter(109, 0, 1)
2023-07-24 14:03:00 043.468978 -080.539810 t_enter(110, 0, 1)
2023-07-24 14:04:00 043.468129 -080.540369 t_leave(103, 0, 1)
2023-07-24 14:05:00 043.466825 -080.540998 c_enter([155, 156, 157], 2, 1)
2023-07-24 14:06:00 043.467371 -080.543486 c_leave([108, 109, 110], 3, 1)
2023-07-24 14:08:00 043.470352 -080.544379 split([100, 101, 102], 4, 1)
2023-07-24 14:08:00 043.470352 -080.544379 split([155, 156, 157], 5, 1)
2023-07-24 14:08:00 043.470352 -080.544379 end(1)
```

The third step is “Graph Transformation”, which is responsible for converting spatial-temporal cluster lifetimes to a graph format. This graph is used for a connected representation of cluster evolution. A graph is represented by its vertices, also called nodes, and edges. The proposed approach represents the structure of spatial-temporal clusters in a graph format by a sequence of nodes, each one for the timestamps in which the cluster has a spatial-temporal relationship. The information for this is extracted from the spatial-temporal cluster lifetime, which contains the time, location, and the relationship of a spatial-temporal cluster stored in each line. Therefore, the proposed approach starts the transformation by reading each cluster lifetime file and writing specific information in the lines of a file for nodes. The information extracted is the identification of the cluster, the date, time, location, represented by the latitude and longitude coordinates, of spatial-temporal clusters at each occurrence of a relationship. The information extracted also includes the number of spatial-temporal objects that the cluster has when the relationship happens, which is its size, and the identification of such spatial-temporal objects. Note that, since spatial-temporal relationships are identified between two consecutive timestamps, the approach creates two nodes, one with information about the spatial-temporal cluster before the relationship happens and another node with information about the cluster after the relationship happens.

The proposed approach represents the relationships of spatial-temporal clusters in a graph format by edges connecting nodes. These edges may connect nodes related to the same spatial-temporal cluster, such as when the relationship is a T_ENTER or a T_LEAVE, or they may connect nodes related to different spatial-temporal clusters, such as when the relationship is a C_ENTER or a C_LEAVE. The information for this is extracted from the spatial-temporal cluster lifetime, which contains the time, location, and the relationships of a spatial-temporal cluster stored in each line. Therefore, the proposed approach ends the transformation by reading each cluster lifetime file and writing specific information in the lines of a file for edges. The information extracted is the identification of the cluster at the start and end of the edge, the date and time at the start and end of the edge, the name of the spatial-temporal cluster relationship, and the identification of the spatial-temporal objects that relate to the relationship. Note that, for performance optimization, spatial-temporal cluster lifetime files are read only once, and therefore nodes and edges are created at the same time. For this reason, sometimes, edges refer to spatial-temporal clusters whose lifetime file is read later.

The information for the nodes and edges of a graph are stored in comma separated value (.csv) files. Once the transformation process ends, the structure of spatial-temporal clusters, their relationships, and therefore their evolution can be represented in a graph format. The files for nodes and edges are then imported into Neo4j, a graph data platform, for a complete and graph-based representation of spatial-temporal cluster evolution.

3.4.2 Analysis

In the proposed approach, spatial-temporal data is analyzed after it is converted to a graph format. The graph contains the evolution of all spatial-temporal clusters. This is the last of the four main steps in Figure 3.13. The simplest analysis that can be performed is a visualization of the data, but improved analysis methods can also be performed on the structure or the relationships of spatial-temporal clusters, or on the graph-based representation of spatial-temporal cluster evolution. The proposed approach also enables other analysis methods, for example ML analysis methods, to be created and performed on the available data.

The fourth step is “Graph Processing”, which is responsible for providing analysis methods and support for the graph-based representation of cluster evolution. An important step in any analysis task is the visualization of data. When visualizing data, similarities, differences, or patterns can be more easily identified or understood. The proposed approach provides a visualization of the nodes and edges of the graph. The nodes and edges can refer to the evolution of a single cluster, or the evolution of several spatial-temporal clusters that have relationships. Graph data visualization happens in Neo4j after spatial-temporal data in a graph format has been imported.

Analysis on the graph containing the spatial-temporal cluster evolution happens on the structure of a spatial-temporal cluster, the spatial-temporal relationships that these clusters have, on the entire connected, graph-based representation of spatial-temporal cluster evolution. Analysis on the structure of clusters relates to the changes in location or in the number of spatial-temporal objects clusters contain, such as identifying the distance traveled by the spatial-temporal objects that a cluster contains. Analysis on the relationships of clusters relates to the frequency or density of the occurrence of such relationships, such as identifying the rate of T_ENTER relationships that happened during the existence of the cluster. Analysis of the graph-based representation of cluster evolution relates to a complete examination of the data available, including the calculation of the distribution of clusters, their movement, and similarity. Because analysis methods are performed in Neo4j, they are written in Cypher, the graph query-language created for the Neo4j platform.

During analysis, it is often necessary to visualize data in many different perspectives. A graph visualization is one of these perspectives, but it does not include a geographic map of the location being studied, and some important information may be missed. For this reason, the approach includes support for data to be visualized in a Geographic Information System (GIS). Any GIS is useful to visualize spatial-temporal data about clusters or objects. Quantum GIS (QGIS)[6] is an open-source GIS that supports the visualization of spatial-temporal data. In QGIS, an OpenStreetMap (OSM) map can be opened and data can be imported to be visualized on this map. OSM[7] is a collaborative geographic database that includes a world map. In QGIS, some operations can be performed on spatial-temporal data, such as changing the size and color of markers, observing the movement of individual spatial-temporal objects in time, connecting timestamp-specific markers with arrows, or even making use of a Python console for further processing.

An explanation of the types of analysis that can be performed in the proposed approach is not extensive because novel methods are created. Figure 3.13 represents the analysis methods not discussed here with an entity named “Others”. These analysis methods include those that extend the methods discussed for the structure and relationships of clusters and the graph-based representation of cluster evolution, but it also includes traditional graph analysis methods or ML methods, such as the shortest path in a graph or neural network.

There are a number of graph data platforms where the converted spatial-temporal data can be imported and a graph-based cluster evolution can be analyzed. The list includes Apache Giraph[8], ArangoDB[9], and Amazon Neptune[10]. The choice of Neo4j relates to the graph-querying language Cypher, which became a standardized query language for graph

[6] https://www.qgis.org

[7] https://www.openstreetmap.org

[8] https://giraph.apache.org

[9] https://www.arangodb.com

[10] https://aws.amazon.com/neptune/

processing together with its open-source implementation openCypher[11].

In general, graph data platforms, such as Neo4j, are built to process many nodes or edges efficiently. However, some optimizations are necessary to make sure that results are computed in a timely manner. These optimizations are now discussed. First, several analysis methods look for the beginning and the end of a cluster or a sequence of clusters in an evolution, called a spatial-temporal path. These nodes are marked with additional properties called `cStart`, `cEnd`, `pStart`, and `pEnd`. Second, these properties are indexed. Database indexes are structures kept by database management systems, usually in the format of a table, with important values that are frequently retrieved. These indexes enhance the performance of queries because a search on the table of indexes is much smaller than a search on the entire database. Therefore, the second optimization stores the `cStart`, `cEnd`, `pStart`, and `pEnd` properties in indexes for faster retrieval of nodes. Third, some restrictions are included to limit the search space when retrieving nodes for analysis. Since every spatial-temporal cluster has exactly one start node and exactly one end node, the pairs (`clusterId`, `cStart`), (`clusterId`, `cEnd`) are unique in the database. A similar situation happens with cluster evolution paths. Therefore, the pairs (`clusterId`, `pStart`) and (`clusterId`, `pEnd`) are unique in the database. One additional restriction relates to the unique existence of the node of a spatial-temporal cluster for a given timestamp. Simply put, a spatial-temporal cluster should not happen more than once for each timestamp. Therefore, the pair (`clusterId`, `timestamp`) is unique. Lastly, some edges representing spatial-temporal cluster relationships are marked. Analysis methods that investigate clusters or cluster evolution paths usually examine a sequence of nodes. However, in some cases, several edges exist between two nodes. This is to represent a situation in which a spatial-temporal cluster has several relationships between two timestamps. For example, a T_ENTER and a T_LEAVE. This means that in a long cluster evolution path, there could be many ways from its start to end, which greatly impacts the performance of queries. Since analysis queries are usually interested in the path as a sequence of connected nodes, and for performance issues, every two nodes have at most one relationship marked with the property `pMain`, indicating that methods only need this relationship to connect these two nodes. Analysis methods that seek to investigate all relationships between two nodes can do so by not restricting results based on this property.

[11] https://opencypher.org

Chapter 4

Case Studies

This chapter contains a description of four case studies used to evaluate the proposed approach. The case studies are examples of graph-based cluster evolution. In the case studies, spatial-temporal clusters and their relationships are analyzed and the value of the results of these analyses is explained. The presented case studies demonstrate how graph-based spatial-temporal cluster evolution is critical in handling spatial-temporal problems, such as service supply and demand, traffic and travel flows, human mobility, and city planning, and how the proposed approach provides the needed support. The chapter starts with an explanation of the four datasets used in the case studies followed by sections for each case study.

In each case study, the workflow of the approach is performed, starting with preprocessing the chosen dataset, processing the resulting data, obtaining cluster lifetimes, generating a graph-based representation of cluster evolution, and finally analyzing the graph. When processing the data, one parameter (*rate*) specifies the size of the regular interval in which data is queried and another (*min_shared*) specifies the minimum percentage of spatial-temporal objects for the Jaccard-extended similarity function to identify clusters in consecutive timestamps. Additionally, DBSCAN uses two parameters for density calculation, namely ε and *min_cluster*. In all analysis tasks performed in the case studies, the regular interval has a size of 1 minute, because the dataset already has a 1-minute interval, and $min\_shared = 0.66$, following the discussion in Section 3.2.1. DBSCAN parameters have the values $\varepsilon = 50$ meters and $min\_cluster = 3$. This means that the radius of the neighborhood of data points is 50 meters. As explained in the next subsections, most of the data points are vehicles (e.g. taxis and trucks) and 50 meters is a reasonable distance between vehicles in the same cluster. Other values, such as 100 meters and 25 meters have been tested, but created strange clusters or excluded important clusters, respectively, from the analysis. This also means that the minimum number of data points to start a cluster is 3. A value of 2 means that two vehicles passing by each other form a cluster, which is not ideal. Tests with values of 4 or greater showed that some interesting clusters are not captured. In summary, these values are reached after failed attempts to find interesting phenomena using other values. One exception is analysis tasks that use the T-Drive dataset because of an irregular measurement interval. In this case, the size of the regular interval in which data is queried is five minutes.

The feasibility of the automation of the analysis of spatial-temporal clusters in the solution is assessed through four case studies. The execution of each case study resulted in the expected outcomes. In the end, analysis and visualization of spatial-temporal cluster evolution produced results, which are explained in each cases study. Furthermore, as the analysis of static clusters is demonstrated to be feasible [17, 101, 57], case studies presented here show the feasibility of the cluster evolution analysis workflow.

To illustrate the complexity of the algorithms used in the case studies, note that typically the algorithms take less than 300 milliseconds to finish. One notable exception is one execution during Case Study 3 that takes 16 hours to finish because of the amount of data available. Parallelism or concurrency is possible, helpful, and is attempted. However, the algorithm is executed in low concurrency mode, which explains the result. Some optimizations are discussed in the case studies. When querying a database, a query plan possibly with many steps is built and executed. In the query plans built for the case studies, most of the steps use less than 1 kilobyte (KB). Some steps in Case Study 3 use 17 KB and some steps in Case Study 4 use 59 KB.

4.1 Datasets

This section describes the datasets used in the case studies. Four datasets of spatial-temporal data are used: the Athens Trucks, the Rome Taxis, the Geolife, and the T-Drive datasets. All the datasets relate to vehicles moving, such as personal cars, taxis, and trucks. These datasets vary in size, number of spatial-temporal objects, and time of observation.

4.1.1 The Athens Trucks Dataset

The Athens Trucks dataset[1] contains spatial-temporal data about the movement of 50 trucks in Athens, Greece, collected during 33 distinct days. These trucks deliver concrete

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

to several construction sites around the Athens metropolitan area.

The spatial-temporal data about the trucks form 276 trajectories. The representation of the data is as follows: the spatial dimension, the location of the trucks, is represented by latitude and longitude coordinates and the temporal dimension is represented by timestamps in the local time. Measurements start on August 7, 2002 and end on September 16, 2002. The entire dataset has an uncompressed size of 7.7 megabytes (MB).

4.1.2 The Rome Taxis Dataset

The Rome Taxis dataset[2] [16] contains spatial-temporal data about the movement of approximately 320 taxis in Rome, Italy, collected over 30 days. These taxis serve passengers, many of whom are tourists, in the center of Rome.

The representation of the spatial-temporal data is as follows: the spatial-dimension, the location of the taxis, is represented by latitude and longitude coordinates and the temporal dimension is represented by timestamps in the local time. The sampling rate is every seven seconds. Measurements start on February 1, 2014 and end on March 2, 2014. The entire dataset has an uncompressed size of 1.61 gigabytes (GB).

4.1.3 The Geolife Dataset

The Geolife dataset[3] [153, 151, 152] contains spatial-temporal data about the movement of 182 users mostly in Beijing, China, collected over three years. These users moved around outdoors, including not only domestic routines such as go home and go to work but also entertainment and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. The Geolife dataset is the result of a project by Microsoft Research Asia[4].

The spatial-temporal data about the users form 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of more than 48,000 hours. The representation of spatial-temporal data is as follows: the spatial-dimension, the location of the users, is represented by latitude, longitude, and altitude coordinates and the temporal dimension is represented by timestamps recorded in the local time. The sampling rate varies from 5 seconds to a few minutes. Measurements start on April 12, 2007 and end on July, 27, 2012. The entire dataset has an uncompressed size of 1.15 GB.

[2] https://ieee-dataport.org/open-access/crawdad-romataxi

[3] https://www.microsoft.com/download/details.aspx?id=52367

[4] https://www.microsoft.com/research/lab/microsoft-research-asia/

4.1.4 The T-Drive Dataset

The T-Drive dataset[5] [146, 147] contains spatial-temporal data about the movement of 10,357 taxis in Beijing, China, collected during seven days. These taxis serve passengers in the center of Beijing.

The spatial-temporal data about the taxis is described by about 15 million points with a total distance of about 9 million kilometers. The representation of spatial-temporal data is as follows: the spatial-dimension, the location of the taxis, is represented by latitude and longitude coordinates and the temporal dimension is presented by timestamps in the local time. The sampling rate varies from a few seconds to a few minutes, averaging about 177 seconds. Measurements start on February 2, 2008 and end on February 8, 2008.

4.2 Case Study 1

Case Study 1 is an exploratory study in which several queries are executed to extract value from the graph-based cluster evolution present in the dataset. This is not an extensive list of possible queries, but a list of important and possible queries related to the representation, analysis methods, and implementation tools, or software support, used in the proposed approach. Each query is presented and its results are discussed. This case study uses the Athens Trucks and the Rome Taxis datasets.

Table 4.1 shows the 23 queries for Case Study 1 and is a guide for the next subsections. They are divided into queries about the *structure* of spatial-temporal clusters, their *relationships*, and the *graph*-based representation of spatial-temporal cluster evolution. These queries are written in Cypher, the graph query language from Neo4j, and their code is available in Appendix B.1.

4.2.1 Structure

This subsection describes queries executed on the *structure* of spatial-temporal clusters. The subsection is further divided into subsections in which each query is discussed.

[5] https://www.microsoft.com/research/publication/t-drive-driving-directions-based-on-taxi-trajectories/

Table 4.1: Queries of Case Study 1.

Type	Query
Structure	Show a specific cluster.
	How many clusters are there in the dataset?
	How many clusters are there around a specific location?
	What clusters start (end) around a specific location?
	What clusters exist for more than a number of minutes?
	What is the shortest (or longest) cluster with respect to the time it exists?
	What is the largest cluster with respect to the number of spatial-temporal objects?
	What clusters have more than a given number of spatial-temporal objects?
Relationships	What clusters start with a MERGE relationship? (Or What clusters end with a SPLIT relationship?)
	Show the locations where T_ENTER (or T_LEAVE, MERGE, SPLIT, etc.) relationships happen.
	What clusters have more than a given number of T_ENTER (or T_LEAVE) relationships?
	What clusters have a MERGE relationship before a given time?
Graph	Show the evolution of a given cluster, that is, all the cluster evolution paths starting at the given cluster.
	How many cluster evolution paths exist in the database?
	Show all cluster evolution paths and the location they start and end.
	What cluster evolution paths start (end) around a specific location?
	Show the location of every cluster change in all cluster evolution paths.
	What are the cluster evolution paths that start in the morning and end in the afternoon?
	Show all cluster evolution paths and the time they start and end.
	What is the cluster evolution path with the greatest number of relationships?
	What cluster evolution paths have more (or less) than a given number of relationships?
	What is the cluster evolution path with the greatest number of spatial-temporal objects during its existence?
	What is the cluster evolution path with the greatest average number of spatial-temporal objects during its existence?

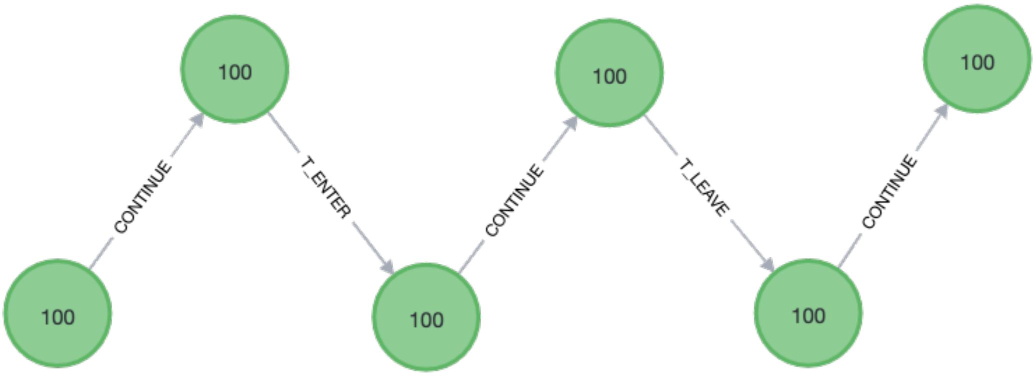


Figure 4.1: The result of the query: cluster c_{100} is shown from start to end.

Show a specific cluster

When analyzing spatial-temporal clusters, it is worthwhile to have a visualization of the cluster from start to end. Visualization is helpful in the observation of the properties of the cluster, such as size or location. This query is executed on the dataset Rome Taxis.

Figure 4.1 shows the result of this query. The figure shows cluster c_{100} from the time it starts to the time it ends. Note that the cluster c_{100} existed during several timestamps. Properties about cluster c_{100}, such as its size or location, are not shown but can be accessed when the occurrence of the cluster at a timestamp is accessed.

How many clusters are there in the dataset?

When analyzing spatial-temporal clusters, it is important to know the number of clusters that exist in the dataset, regardless of their spatial-temporal relationships. This result can be used in frequency or density calculations later. This query is executed on the dataset Geolife.

Table 4.2 shows the result of this query. The table shows that there exist 2 clusters of vehicles in Beijing during the time measurements were taken. This result does not take into consideration spatial-temporal relationships that these clusters might have or the time during which these clusters existed.

This query is also executed on the dataset Rome Taxis for a more expressive result and because most of the remaining queries are also executed in this dataset. Table 4.3

Table 4.2: The number of spatial-temporal clusters that exist in the Geolife dataset.

Number of spatial-temporal clusters
1,146

Table 4.3: The number of spatial-temporal clusters that exist in the Rome Taxis dataset.

Number of spatial-temporal clusters
18,699

shows the result of this query. The table shows that there exist 18,699 clusters of taxis in Rome during the time measurements were taken. As before, this result does not take into consideration spatial-temporal relationships that these clusters might have or the time during which these clusters existed.

How many clusters are there around a specific location?

Several analysis methods require the identification of spatial-temporal clusters that are near a specific location and the number of such clusters. The results of this query can be used, for example, to calculate interest, assuming a high number of clusters around a location means high interest and a low number of clusters around a location means low interest. In reality, however, other cluster properties, such as cluster density, can also be used to improve interest calculation, but the number of clusters can provide initial results.

This query is executed on the dataset Rome Taxis. Since Rome receives many tourists during the year, the query is executed for the location of one of the tourist sites in Rome, the Colosseum. The Colosseum (Italian: *Colosseo*) is an amphitheater built under the Flavian emperors of the Roman Empire. Its construction began under the Roman Emperor Vespasian between the years of 70 and 72. He intended the Colosseum to be an entertainment venue, hosting gladiator fights, animal hunts, and even mock naval battles. The Colosseum is located at the latitude and longitude coordinates `41.890278, 12.492222` in decimal format. In this query, “near” or “around” a specific location is defined to be at most 500 meters between the places being compared. The value of 500 meters is chosen after some tests to find a reasonable number of clusters.

Table 4.4 shows the result of this query. There are 61 clusters that, at some point during their existence, are near the Colosseum. These clusters can be further investigated for improved results. One interesting improvement is to observe the distribution of these clusters over time. To achieve this, time is divided into intervals of one hour and the

Table 4.4: The number of clusters and the clusters of taxis around the Colosseum.

Number	Clusters
61	c_{8954}, c_{4473}, c_{8021}, c_{12125}, c_{11515}, c_{17301}, c_{9208}, c_{16431}, c_{16487}, c_{10073}, c_{7999}, c_{1405}, c_{2733}, c_{16006}, c_{3393}, c_{9210}, c_{6741}, c_{5447}, c_{16900}, c_{17564}, c_{4978}, c_{1026}, c_{17289}, c_{15349}, c_{3301}, c_{17761}, c_{14685}, c_{518}, c_{17170}, c_{17529}, c_{6976}, c_{13338}, c_{18068}, c_{17742}, c_{7160}, c_{7751}, c_{7078}, c_{6945}, c_{7065}, c_{13345}, c_{3001}, c_{4662}, c_{13352}, c_{9236}, c_{13355}, c_{15220}, c_{6035}, c_{17534}, c_{11951}, c_{8020}, c_{18069}, c_{15841}, c_{4705}, c_{10593}, c_{17539}, c_{9238}, c_{13598}, c_{9775}, c_{15329}, c_{10068}, c_{18067}

timestamp of the clusters is used to pick the appropriate interval for the cluster. Table 4.5 shows the same 61 clusters distributed by the hours of the day. Notice the hours 10, 17, and 19, when the number of clusters exceed 8. This indicates that these are likely the peak times for tourists near the Colosseum. Results in Table 4.5 do not take into consideration the day when the cluster exists, showing an aggregation by the hour of the day. Another improvement is to consider the full date, including the day, month, and year, of the existence of clusters. Another way to visualize the data from Table 4.4 is in a distribution by day of the week.

What clusters start (end) around a specific location?

Inspecting the clusters that start at a specific location can give insights about how many clusters are being formed at that location. Extra information about the clusters, such as the time during which clusters exist, can expand on this investigation.

This query is executed on the dataset Rome Taxis. As the inspected dataset contains trajectories of taxis in Rome, it is interesting to investigate the clusters formed near popular tourist sites in Rome. One such place is Piazza Venezia, a square near some of the most important tourist sites in Rome, such as the Roman Forum and the Colosseum. The latitude and longitude coordinates of Piazza Venezia are `41.8964, 12.4825` in decimal format. In this query, "near" or "around" a specific location is defined to be at most 100 meters between the places being compared. The value of 100 meters is chosen after some tests to find a reasonable number of clusters.

There are 928 clusters that start around Piazza Venezia in the dataset. The map in Figure 4.2 shows the result of this query. In the figure, dots are the location where clusters start and end. Green dots represent the start location and black dots represent the end location. Each dot has the cluster ID below it. While most clusters end around the same

Table 4.5: The distribution of the 68 clusters that exists near the Colosseum per hour of the day.

Hour	Number	Clusters
1	3	c_{11951}, c_{15841}, c_{13598}
9	2	c_{3001}, c_{9775}
10	6	c_{12125}, c_{1405}, c_{16006}, c_{6741}, c_{15220}, c_{10593}
11	3	c_{17529}, c_{17534}, c_{17539}
12	3	c_{11515}, c_{17564}, c_{6035}
13	4	c_{15349}, c_{3301}, c_{6945}, c_{15329}
14	4	c_{16900}, c_{6976}, c_{7751}, c_{10068}
15	4	c_{10073}, c_{1026}, c_{518}, c_{17742}
16	5	c_{3393}, c_{5447}, c_{17761}, c_{7078}, c_{7065}
17	8	c_{9208}, c_{9210}, c_{14685}, c_{13338}, c_{7160}, c_{13345}, c_{4662}, c_{13352}
18	1	c_{13355}
19	9	c_{8021}, c_{16431}, c_{7999}, c_{2733}, c_{17170}, c_{9236}, c_{8020}, c_{4705}, c_{9238}
20	1	c_{16487}
21	1	c_{8954}
22	5	c_{4978}, c_{17289}, c_{18068}, c_{18069}, c_{18067}
23	2	c_{4473}, c_{17301}

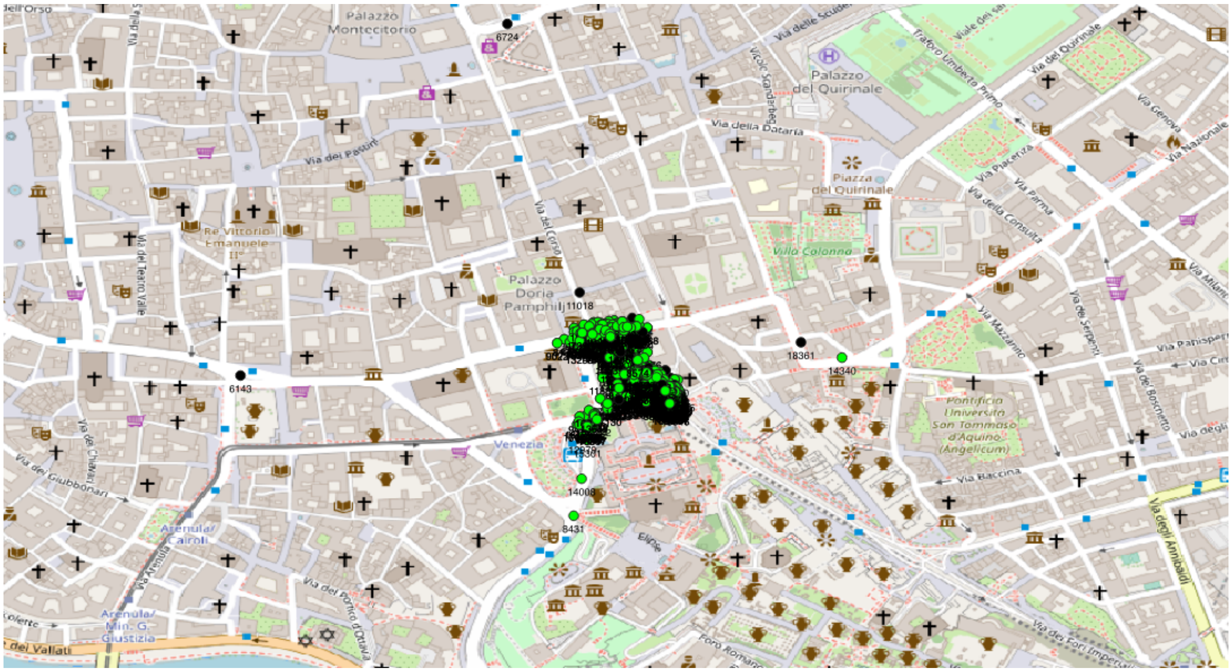


Figure 4.2: Spatial-temporal clusters that start around Piazza Venezia in Rome. Green dots represent their start location and black dots represent the end location.

place, a few clusters move to another, nearby, spot, namely clusters c_{6143}, c_{6724}, and c_{18361}. Further investigation on the reasons these clusters remained together may reveal interesting insights into the movement of taxis in this area.

What clusters exist for more than a number of minutes?

There are many reasons for spatial-temporal clusters to form. There could be an event in a city, or an accident, or a festival. Usually, these clusters last for some minutes and are the object of investigation. However, some clusters take a significant amount of time before they stop existing. Analyzing spatial-temporal clusters that exist for more than a number of minutes can give insights as to why they last for a greater amount of time or identify new opportunities for improvement. For example, spatial-temporal objects that are together, forming a cluster, for several timestamps are an opportunity for car-sharing or ridesharing.

This query is executed on the dataset Rome Taxis. In this query, the value of 60 minutes is chosen for investigation. The value of 60 minutes is chosen after some tests to

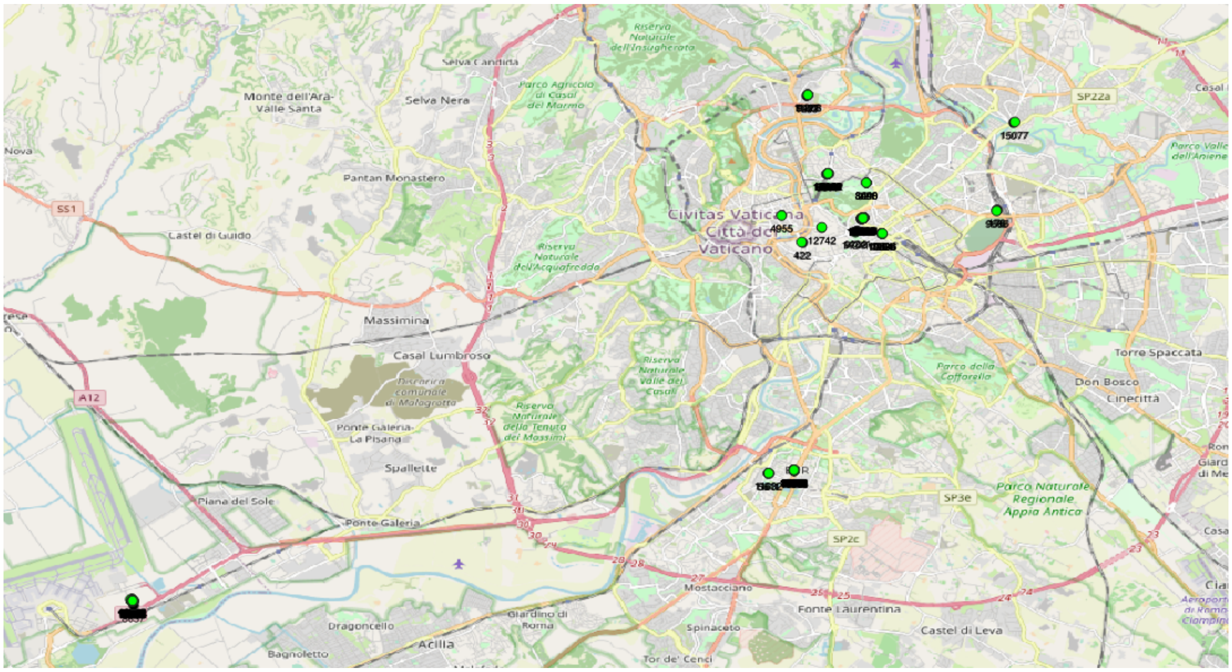


Figure 4.3: A total of 129 spatial-temporal clusters last more than 60 minutes.

find a reasonable number of clusters.

The map in Figure 4.3 shows the result of this query. A total of 129 spatial-temporal clusters exist for more than one hour. Some of these clusters are taxi stands, where several taxis arrive, wait for passengers, and go. The fact that clusters start and end at the same location strengthens this idea. The number of long-term spatial-temporal clusters formed at these locations provides insights about the importance of the location. Some locations have only one cluster, while other locations have more than 10.

What is the shortest (or longest) cluster with respect to the time it exists?

Analyzing the extremes is a good way to find outliers, edge cases, or problems in the dataset. This query finds the shortest and longest spatial-temporal clusters with respect to the time they exist. This query is similar to the one described earlier, but they differ in that the previous one assigns limits in the search, while this does not, because it searches for the extremes. This query is executed on the dataset Rome Taxis.

Table 4.6 shows the result of this query. A total of 11,695 spatial-temporal clusters

Table 4.6: Longest spatial-temporal clusters with respect to the time they existed in the Rome Taxis dataset.

Cluster	Duration
c_{9547}	259
c_{8657}	244
c_{4103}	230
c_{4921}	228
c_{12466}	225
c_{16282}	207
c_{1187}	207
c_{1280}	191
c_{9215}	171

last less than a minute. They are detected at a specific timestamp and are not detected at the next timestamp. A total of 1,690 clusters exist for exactly 1 minute, meaning that the difference between the timestamps at which they end and start is 1 minute. These are a significant number of spatial-temporal clusters that are short with respect to the time they existed considering that a total of 18,699 clusters are inspected. These clusters seem to represent cases in which taxi drivers test their GPS devices or taxis pass by each other. Table 4.6 shows the 10 longest spatial-temporal clusters in the dataset and the time they last in minutes. Cluster c_{9547} exists for more than 4 hours. It represents a parking lot near the airport of Rome where taxi drivers wait for passengers. The duration of a cluster of taxis in this parking lot relates to the demand for taxis in the airport at that time of the day.

What is the largest cluster with respect to the number of spatial-temporal objects?

A large spatial-temporal cluster grows or shrinks depending on the surrounding conditions. Analyzing the size of the cluster, that is the number of spatial-temporal objects it contains, indicates whether the demand for services is high and helps interested parties to consider reallocating spatial-temporal objects accordingly. This query is executed on the dataset Rome Taxis.

Table 4.7 shows the result of this query. The table shows the 10 largest clusters with respect to the number of spatial-temporal objects they contain. Spatial-temporal cluster c_{8657} is present in the previous table, as one of the longest existing clusters in the dataset,

Table 4.7: The largest spatial-temporal clusters with respect to their size in the Rome Taxis dataset.

Cluster	Number of Spatial-Temporal Objects
c_{8657}	12
c_{8904}	12
c_{9507}	11
c_{9547}	11
c_{12466}	11
c_{15118}	11
c_{1475}	10
c_{14228}	10
c_{10408}	10

and now as one of the largest as well. The cluster similar to cluster c_{9547} also in Table 4.7 exists next to the airport of Rome where taxis wait for passengers. Spatial-temporal cluster c_{8904} exists in the center of Rome, next to the Piazza Navona. Further inspection shows that this cluster existed from 8:52 PM to 9:35 PM on February 14, 2014, which is Valentine's Day. It is then likely that cluster c_{8904} was formed expecting those who had celebrated Valentine's Day to return home.

What clusters have more than a given number of spatial-temporal objects?

Analyzing properties of spatial-temporal clusters, such as the number of spatial-temporal objects in a cluster, is the way to examine the structure of these clusters. Sometimes, limiting the range of these properties improves the quality of the results or is a technical requirement. This query searches for the spatial-temporal clusters that contain more than a given number of spatial-temporal objects. This query is similar to the one described earlier, but they differ in that the previous one searches for extremes, while this query assigns limits to the inspected clusters.

This query is executed on the dataset Rome Taxis. The query searches for spatial-temporal clusters that contain more than 10 spatial-temporal objects. The value of 10 spatial-temporal objects is chosen to demonstrate the query and can be changed to any desired value.

Table 4.8: Spatial-temporal clusters that contain more than 10 spatial-temporal objects in the Rome Taxis dataset (a) and spatial-temporal clusters that contain more than five spatial-temporal objects on the Athens Trucks dataset (b).

Cluster
c_{8657}
c_{8904}
c_{9507}
c_{9547}
c_{12466}
c_{15118}

(a)

Cluster
c_{34}
c_{55}
c_{109}
c_{122}

(b)

Table 4.8a shows the result of this query. As expected, Table 4.8a is very similar to Table 4.7. Clusters c_{1475}, c_{14228} and c_{10408} contain exactly 10 spatial-temporal clusters and are not included in Table 4.8a because of the restriction on the wording of the query that means "strictly greater than".

As the results in Table 4.8a are very similar to those in Table 4.7, the query is then executed on the dataset Athens Trucks as well. The query is updated to search for spatial-temporal clusters that contain more than five spatial-temporal objects.

Table 4.8b shows the result of this query. Only one spatial-temporal cluster, namely cluster c_{55}, contains more than five trucks. Further inspection reveals that this cluster exists at a garage of a cement company with loading docks and machinery to load trucks. Three other spatial-temporal clusters, namely c_{34}, c_{109}, and c_{122} contain exactly five trucks and are not shown as the result of the query because of the limitation of the wording of the query that means "strictly greater than". These clusters also exist at garages of a truck company. It is reasonable to expect that larger clusters happen at start or end points of a trip, in the case of a truck company, because these are the places where trucks naturally cluster.

4.2.2 Relationships

This subsection describes queries executed on the *relationships* of spatial-temporal clusters. The subsection is divided into subsections in which each query is discussed.

What clusters start with a MERGE relationship? (Or What clusters end with a SPLIT relationship?)

Analysis on the spatial-temporal clusters that start with a MERGE relationship provides more information about the existence of these clusters. For example, traffic jams start with clusters merging, which are represented by several MERGE relationships. Similarly, the end of events, for example, are represented by large spatial-temporal clusters splitting multiple times as spatial-temporal objects leave in groups to other parts of a city. This produces many SPLIT relationships. This query is executed on the Rome Taxis dataset.

Table 4.9 shows the result of this query. There are 66 spatial-temporal clusters that start with a MERGE relationship. Figure 4.4 shows the locations where these clusters exist. Note that most of the clusters are grouped around two locations. The top location is a region in Rome with many hotels. Taxis at this location might be waiting for passengers. The bottom location is a region near a bus, taxi, and train terminal. Taxis at this location are waiting for passengers to continue their commute. Spatial-temporal clusters start with a MERGE relationship when two or more clusters that are initially apart group together, forming a large cluster. The main reason for this happening in the dataset can be the parameter ε used by the approach. A small ε means a stricter interpretation of a distance between near points in a cluster, tending to break clusters into many smaller ones. A larger ε admits larger distances between points in a cluster, tending to produce larger clusters both in size and shape. The value of $\varepsilon = 50$ is used for this analysis in this dataset. This value can break what should be a large spatial-temporal cluster into small clusters that eventually approach each other and effectively create the larger spatial-temporal cluster, in these regions. A larger ε can filter out these cases, allowing other situations of MERGE relationships to appear, but it also makes the clustering algorithm accept spatial-temporal clusters where spatial-temporal objects are fairly distant.

Table 4.10 also shows the result of this query. There are 75 spatial-temporal clusters that end with a SPLIT relationship. All of them split into exactly two other clusters. No split into three or more clusters is observed. The map in Figure 4.5 shows the locations where these clusters exist. Among the 75 spatial-temporal clusters, 12 have a SPLIT relationship in the taxi parking lot near the airport, not shown in the figure because of space limitation, other 23 have the SPLIT relationship near the terminal for taxi, train, and bus, and 34 have the SPLIT relationship in a street called Via Boncompagni where many luxury hotels are located.

Table 4.9: Spatial-temporal clusters that start with a MERGE relationship in the Rome Taxis dataset.

Clusters					
c_{41}	c_{3067}	c_{4851}	c_{8327}	c_{12869}	c_{14926}
c_{755}	c_{3196}	c_{4960}	c_{8372}	c_{13187}	c_{14929}
c_{759}	c_{3434}	c_{5060}	c_{8384}	c_{13277}	c_{14941}
c_{807}	c_{3451}	c_{5064}	c_{8454}	c_{14253}	c_{15137}
c_{898}	c_{3453}	c_{5899}	c_{9532}	c_{14260}	c_{15141}
c_{1080}	c_{3458}	c_{6019}	c_{9544}	c_{14265}	c_{15145}
c_{1169}	c_{3870}	c_{6033}	c_{9547}	c_{14270}	c_{15259}
c_{1499}	c_{4013}	c_{6326}	c_{10055}	c_{14316}	c_{15865}
c_{2129}	c_{4018}	c_{6923}	c_{10145}	c_{14546}	c_{16618}
c_{2132}	c_{4172}	c_{8273}	c_{12186}	c_{14808}	c_{16836}
c_{2199}	c_{4177}	c_{8275}	c_{12233}	c_{14920}	c_{17442}

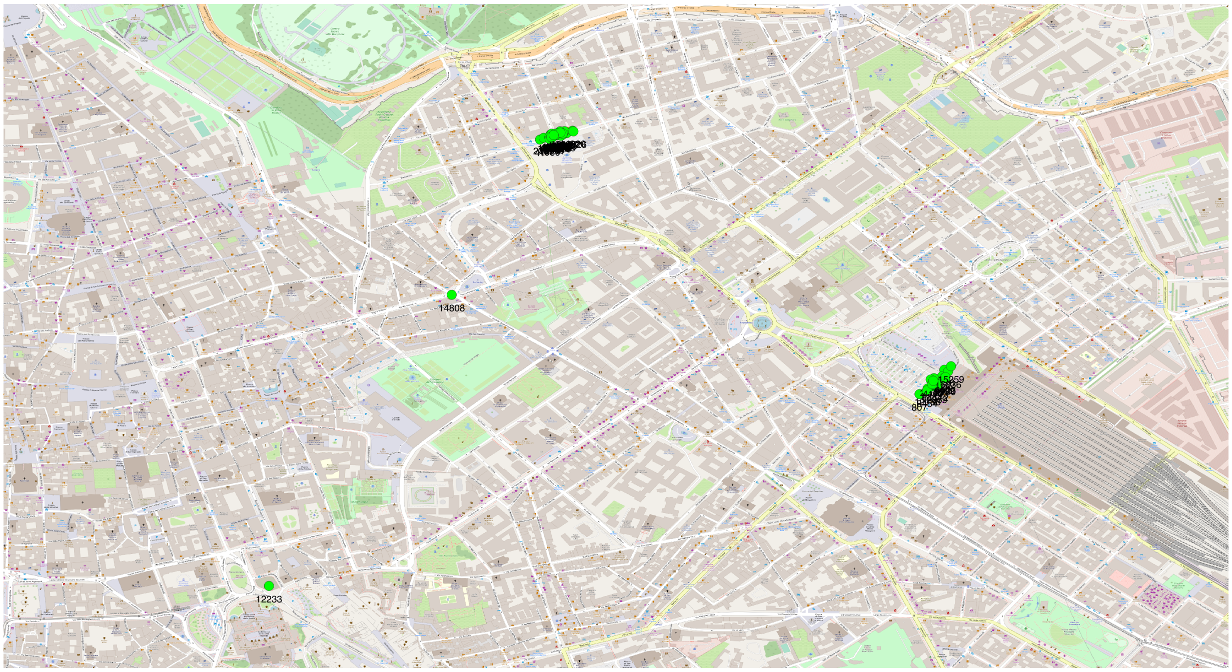


Figure 4.4: The location of spatial-temporal clusters that start with a MERGE relationship in the Rome Taxis dataset.

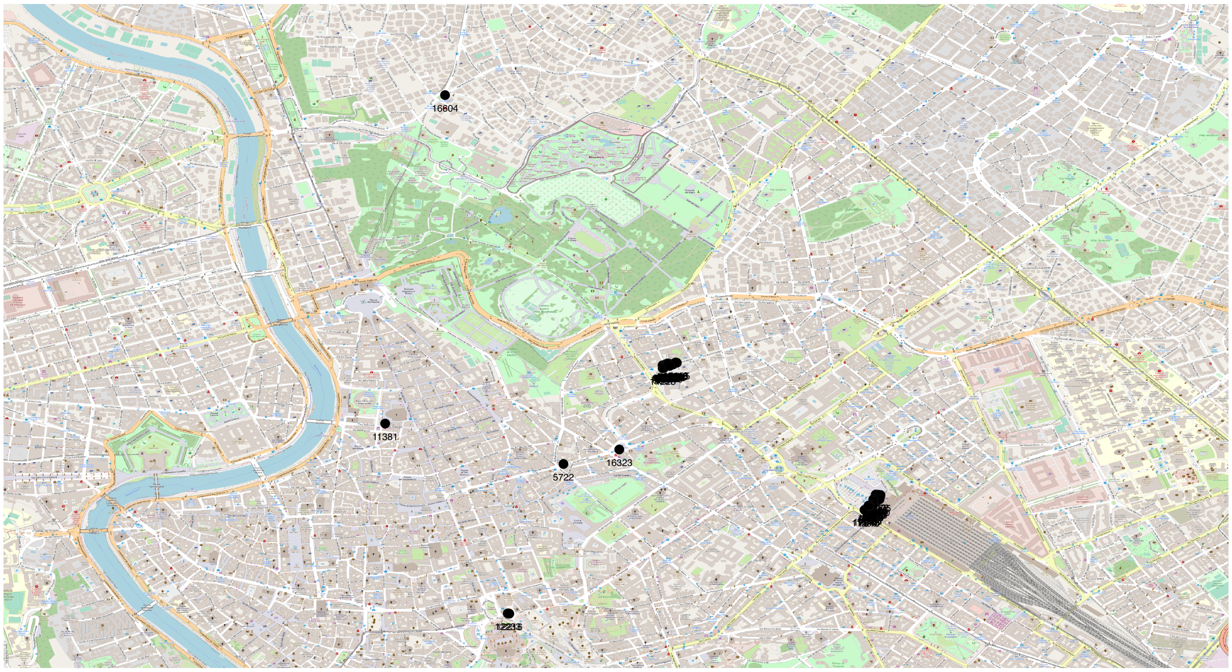


Figure 4.5: The location of spatial-temporal clusters that end with a SPLIT relationship in the Rome Taxis dataset.

Table 4.10: Spatial-temporal clusters that end with a SPLIT relationship in the Rome Taxis dataset.

Clusters				
c_{794}	c_{4172}	c_{8358}	c_{14223}	c_{15078}
c_{798}	c_{4219}	c_{8372}	c_{14228}	c_{15118}
c_{885}	c_{4845}	c_{8444}	c_{14250}	c_{15137}
c_{1463}	c_{4960}	c_{8454}	c_{14253}	c_{15141}
c_{1475}	c_{5048}	c_{9507}	c_{14260}	c_{15262}
c_{1499}	c_{5060}	c_{9544}	c_{14265}	c_{15854}
c_{2129}	c_{5064}	c_{10032}	c_{14298}	c_{15857}
c_{3047}	c_{5090}	c_{10553}	c_{14316}	c_{16323}
c_{3067}	c_{5722}	c_{11290}	c_{14319}	c_{16407}
c_{3390}	c_{5805}	c_{11381}	c_{14381}	c_{16447}
c_{3429}	c_{5830}	c_{11967}	c_{14525}	c_{16604}
c_{3434}	c_{5837}	c_{12213}	c_{14896}	c_{16755}
c_{3453}	c_{5969}	c_{12216}	c_{14920}	c_{16836}
c_{3927}	c_{6923}	c_{12233}	c_{14926}	c_{17076}
c_{4013}	c_{7910}	c_{13231}	c_{14929}	c_{17437}

Show the locations where T_ENTER (or T_LEAVE, MERGE, SPLIT, etc.) relationships happen.

Analysis of the location of spatial-temporal cluster relationships uncover important information about how spatial-temporal objects behave during observation. For example, several T_ENTER relationships happening at a specific part of a highway may indicate the location where congestion starts.

This query is executed on the Rome Taxis dataset. The query searches for the location of every T_ENTER relationship in the dataset.

Figure 4.6 shows the result of this query. There exist 8,439 occurrences of the T_ENTER relationship. The figure shows some of these locations and marks them with a dot. Not all locations are shown in Figure 4.6 because of space restrictions. Some locations are distant from the center of Rome, such as the Rome airport, and showing them would reduce the level of detail in the figure. By omitting these dots, the figure prioritizes details in the center of Rome. Notice that the T_ENTER relationships happen at somewhat well-defined locations. These are taxi stands where taxis wait for passengers, but they can now be ranked by importance. Taxi stands where many T_ENTER relationships happen indicate

Table 4.11: Spatial-temporal clusters that have more than 15 T_ENTER relationships in the Rome Taxis dataset.

Clusters			
c_{3210}	c_{4921}	c_{9547}	c_{16282}
c_{3390}	c_{5810}	c_{9679}	c_{16601}
c_{3458}	c_{8280}	c_{10408}	c_{17373}
c_{4103}	c_{8657}	c_{12466}	c_{17797}
c_{4908}	c_{9507}	c_{13237}	

a greater interest in the location.

What clusters have more than a given number of T_ENTER (or T_LEAVE) relationships?

The analysis of the number of spatial-temporal cluster relationships during the existence of spatial-temporal cluster is important for frequency or rates calculation. For example, a cluster where several spatial-temporal objects entered, generating T_ENTER relationships, that grows, indicates an important cluster or a cluster in an important location, and give insights about the movement of the spatial-temporal objects.

This query is executed on the Rome Taxis dataset. The query searches for spatial-temporal clusters that have more than 15 T_ENTER relationships during their existence. The value of 15 T_ENTER relationships is chosen to demonstrate the query and can be changed to any desired value.

Table 4.11 shows the result of this query. There are 19 spatial-temporal clusters that have more than 15 T_ENTER relationships. Further inspection shows that these clusters end around the same location where they start. It also shows that some of these clusters happen at the taxi parking stop near the Rome airport and the street Via Boncompagni where luxury hotels are located. However, three of these clusters happened at two other locations and this is worth noting. The locations are Piazzale Don Giovanni Minzoni to the north of Rome and Obelisco di Marconi to the south of Rome. Both locations are near museums of Art or History, which results in a higher interest in the regions and explains in part the increased number of T_ENTER relationships.

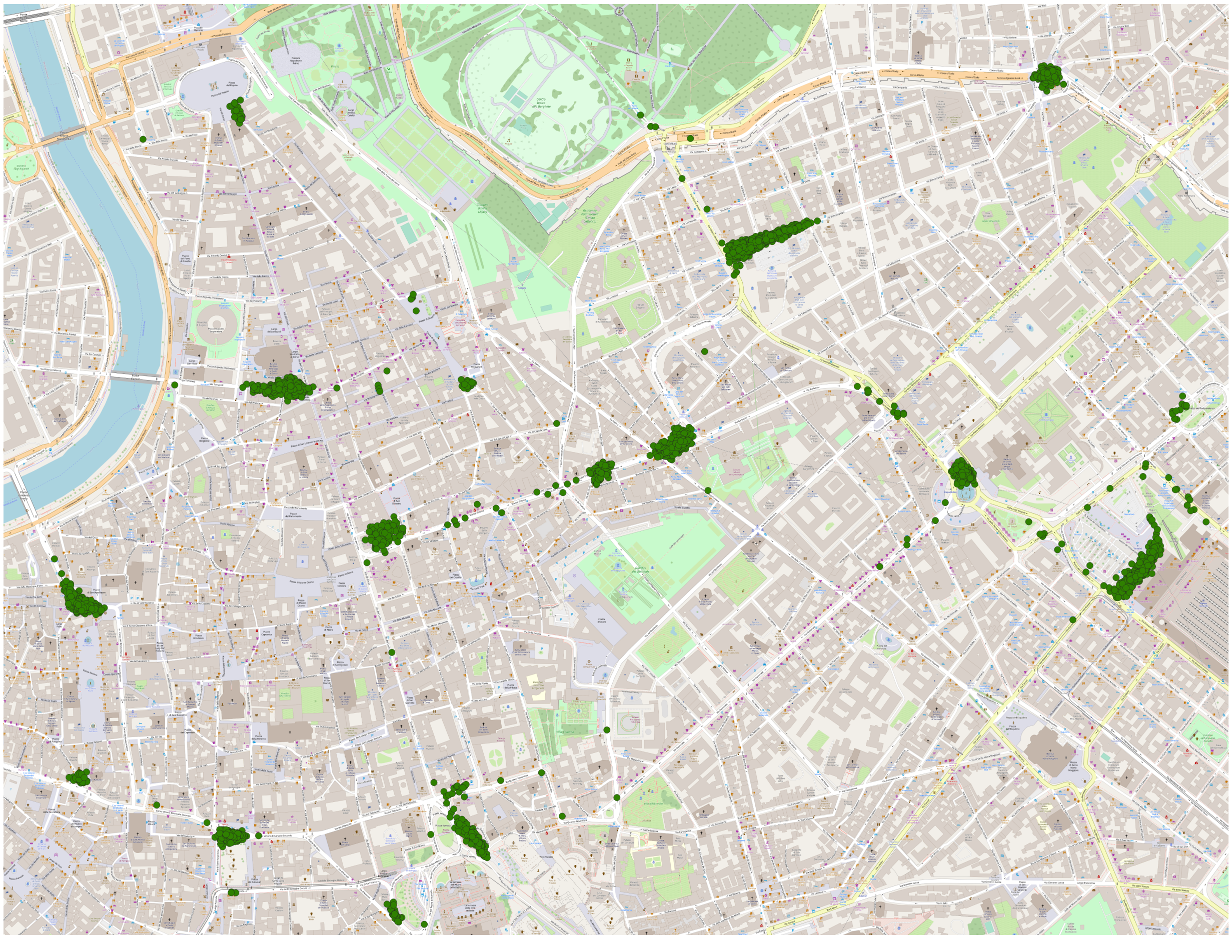


Figure 4.6: The locations where T_ENTER relationships happened in the Rome Taxis dataset.

Table 4.12: Spatial-temporal clusters that have a MERGE relationship before 8 AM in the Rome Taxis dataset.

Clusters	
c_{41}	c_{8275}
c_{5060}	c_{14253}
c_{5064}	c_{15865}
c_{8273}	c_{16618}

What clusters have a MERGE relationship before a given time?

Analysis on the occurrence of spatial-temporal cluster relationships limited by some temporal restriction allows for the observation of interesting phenomena during specific times of the day. For example, interesting taxi stands in the morning may not be the same in the afternoon, or at night. Limiting the analysis to some times during the day can lead to more in-depth results.

This query is executed on the Rome Taxis dataset. The query searches for spatial-temporal clusters that have a MERGE relationship before 8 AM. The value of 8 AM is chosen to demonstrate the query and can be changed to any desired value.

Table 4.12 shows the result of this query. Eight spatial-temporal clusters are created as a result of a MERGE relationship before 8 AM in the Rome Taxis dataset. Table 4.12 shows the resulting cluster and not the two or more clusters that merged. Further inspection shows that most of these clusters are formed after 7 AM, indicating that this is the time where taxis cluster around taxi stands waiting for passengers.

4.2.3 Graph

This subsection describes queries executed on the *graph*-based representation of spatial-temporal cluster evolution. The structure of spatial-temporal clusters as well as the relationships these clusters have are considered. Moreover, analyses are not limited to one cluster, as the connections between these clusters are also considered in the analyses. The subsection is divided into subsections in which each query is discussed.

Show the evolution of a given cluster, that is, all the cluster evolution paths starting at the given cluster.

Spatial-temporal clusters have different relationships with other clusters or spatial-temporal objects. Visualizing these relationships helps in the identification of patterns or outliers in these relationships. For example, a spatial-temporal cluster that has multiple T_ENTER and T_LEAVE relationships happening at the same time may indicate a full parking lot, where one car enters only when another leaves.

This query is executed on the Rome Taxis dataset. The query results in the spatial-temporal cluster evolution of a cluster, that is, a graph representing all cluster evolution paths starting from this cluster. Spatial-temporal cluster c_{5805} is chosen for this query

Figure 4.7 shows the evolution of spatial-temporal cluster c_{5805}. The cluster has several changes in the number of participants with many T_ENTER or T_LEAVE relationships and then splits into clusters c_{5829} and c_{5830}. The first of these two has changes until it eventually ends. The second of these two has other cluster relationships until it splits again into cluster c_{5837} and c_{5838}. Cluster c_{5838} ends a few timestamps later, but cluster c_{5837} continues to exist, changes its size, and splits again into clusters c_{5841} and c_{5842}, which stops existing moments later. It is interesting to observe that one spatial-temporal cluster, namely cluster c_{5805}, resulted in four other clusters. It is also interesting to note that Figure 4.7 provides a graph-based representation of the complete cluster evolution of cluster c_{5805}.

How many cluster evolution paths exist in the database?

Visualizing cluster evolution from beginning to the many possible ends is a great tool for generating insights, even if the evolution is limited to that of a single spatial-temporal cluster. However, since the analysis of cluster evolution is graph-based, most analysis methods investigate paths from the root node to the leaf node, *i.e.* from the start of a spatial-temporal cluster to one end of the evolution of this cluster. Results can be used to understand the behavior of the cluster among the many transformations it has undergone during its existence. This query is executed on the Rome Taxis dataset.

Table 4.13 shows the result of this query. There are 18,639 cluster evolution paths in the Rome Taxis dataset. This means that, given the starting point of spatial-temporal clusters formed by the GROUP relationship, there exist 18,639 paths starting at these clusters. Note that although there are 18,699 clusters in the dataset, not all of them are the start of a path. Many are in the middle of a path. In fact, there are 18,483 clusters that start a path. This results in an average of approximately 1.0084 cluster evolution

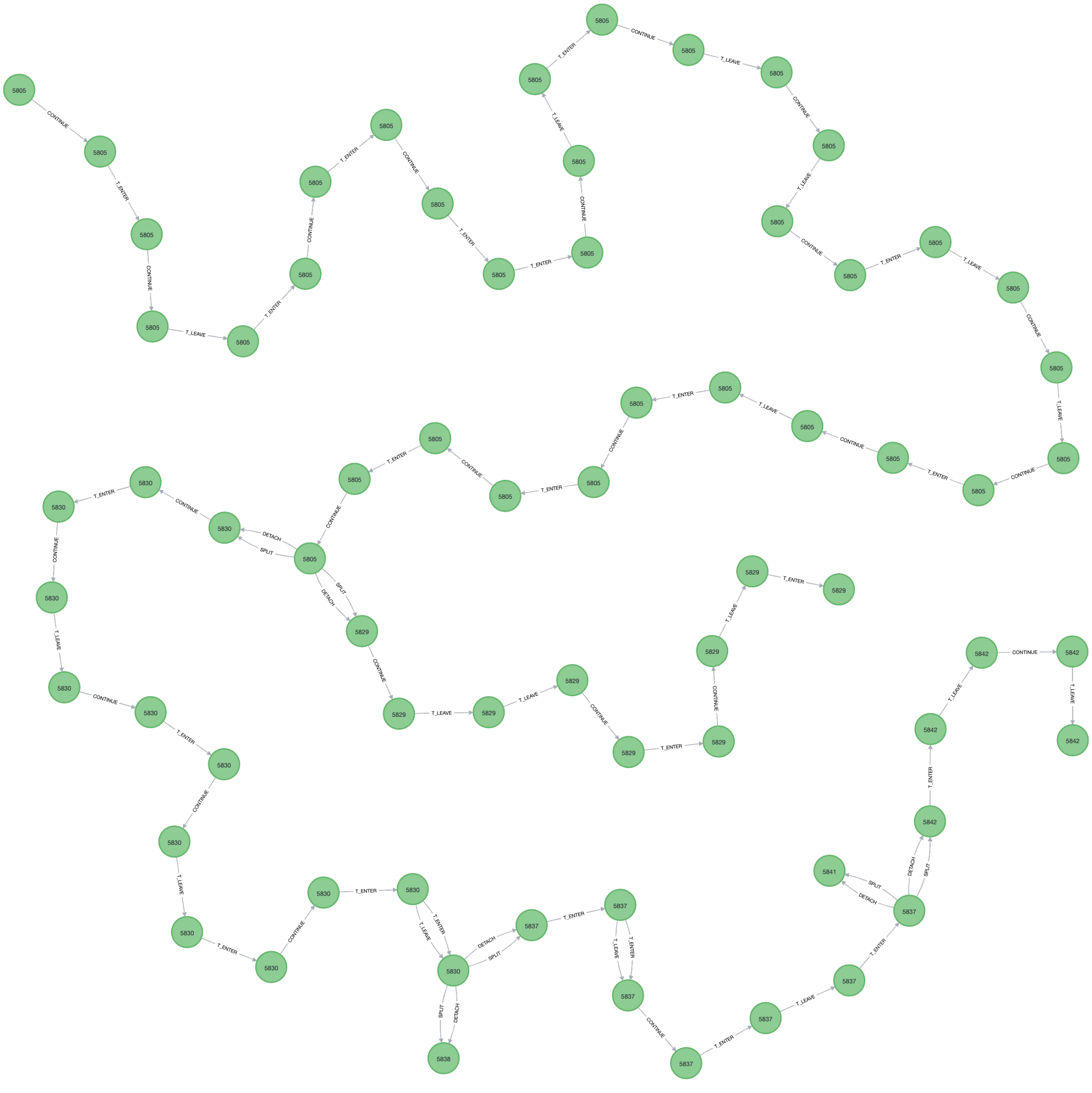


Figure 4.7: The evolution of spatial-temporal cluster c_{5805} in the dataset Rome Taxis.

Table 4.13: Number of cluster evolution paths in the Rome Taxis dataset.

Number of Cluster Evolution Paths
18,639

paths per clusters that start a path. One important consideration is that, sometimes, a cluster c_i has both a T_ENTER and a T_LEAVE relationship between timestamps. This is represented by two arrows from c_{i,t_j} to $c_{i,t_{j+1}}$ in the graph. When calculating a path, the two arrows are not considered different paths since they connect the same nodes of the graph.

Show all cluster evolution paths and the location they start and end.

When performing exploratory analysis, it is worthwhile to have a tabular view of the data of interest with additional data alongside. For example, in ML, training data is usually the data of interest in a tabular format having the labels as the last column. In the context of spatial-temporal clusters, additional data can give insights about why some clusters are selected by the analysis method.

This query is executed on the Rome Taxis dataset. Note that, although cluster evolution paths are represented visually, in a graph, by several nodes at many different timestamps, here, in text, this representation would not be appropriate because many paths would be very long. Instead, cluster evolution paths are represented by the different clusters they contain. So, for example, if cluster c_1 exists for three timestamps and becomes cluster c_2 for another three timestamps, this is represented here and in the remainder of this chapter as a list of two elements, with both clusters.

Table 4.14 shows the result of this query. First, from the result of the previous query, there exist 18,639 cluster evolution paths in the Rome Taxis dataset. It is not possible to represent all paths here because of space limitations, although the analysis does show results for all paths. Instead, Table 4.14 shows results for some cluster evolution paths. Second, note the cluster evolution path in the first row of the table. It shows that the path starts with cluster c_{758}, which has several relationships and eventually merges into cluster c_{759}. More complex cluster evolution paths can be found at the bottom of the table. Third, note the second and third rows of the table. It shows a MERGE relationship between clusters c_{14796} and c_{14807} that creates the spatial-temporal cluster c_{14808}, which eventually disperses. It is possible to observe the latitude and longitude coordinates of the start and end of all cluster evolution paths. This information then can be used for visualization or other types of analyses.

What cluster evolution paths start (end) around a specific location?

Analysis on cluster evolution may benefit significantly when it is restricted to a location. Cluster evolution paths that started at a specific place may or may not develop in similar ways. For example, migration patterns from a specific city to another or a commuting trip from a specific residential area to a city center are restrictions that, when applied, increase the detail of the results. Their cluster evolution paths can be analyzed and compared to draw novel conclusions.

This query is executed on the Rome Taxis dataset. The query searches for cluster evolution paths that start around the Aeroporto di Roma Fiumicino Leonardo da Vinci. Like many airports, the Rome airport has more than one terminal. This means that proper coordinates and a large enough radius must be chosen to be able to capture relevant cluster evolution paths. One of the airport terminals is located at the latitude and longitude coordinates `41.794700, 12.250700` in decimal format. In this query, "near" or "around" a specific location is defined to be distance of at most 500 meters between the places being compared. The value of 500 meters is chosen after some tests to find a reasonable number of cluster evolution paths.

Table 4.15 shows the result of the query. There are 150 cluster evolution paths starting near the airport in Rome. Not all paths are shown in Table 4.15 because of space limitations. One interesting thing to note is that all 113 cluster evolution paths include only one cluster. This means that they start and end on the same cluster, and no MERGE or SPLIT relationship happens. The map in Figure 4.8 shows the airport and the start, represented as green dots, and end, represented as black dots, points of all 150 cluster paths. Cluster evolution paths start at the start of the first cluster and end at the end of the last cluster. Since all cluster evolution paths are made of only one spatial-temporal cluster, they are represented by the start and end of the cluster. All cluster evolution paths end around the airport and none reach the Rome city center. However, if long-lasting cluster evolution paths are encountered, then the analysis would indicate the need for improvements in the transportation system, such as a new bus line.

Table 4.14: Some of the cluster evolution paths in the Rome Taxi dataset and the location they start and end.

Cluster Evolution Paths	Start Latitude	Start Longitude	End Latitude	End Longitude
$[c_{758}, c_{759}]$	41.907832	12.491429	41.907483	12.49021
$[c_{14796}, c_{14808}]$	41.903484	12.488224	41.903643	12.48877
$[c_{14807}, c_{14808}]$	41.90322	12.487057	41.903643	12.48877
$[c_{15247}, c_{15259}]$	41.901673	12.501347	41.901047	12.500748
$[c_{15256}, c_{15259}]$	41.901906	12.501304	41.901047	12.500748
$[c_{752}, c_{755}, c_{759}]$	41.907535	12.490407	41.907483	12.49021
$[c_{754}, c_{755}, c_{759}]$	41.907878	12.491641	41.907483	12.49021
$[c_{798}, c_{805}, c_{807}]$	41.901363	12.501137	41.900838	12.500388
$[c_{3390}, c_{3423}, c_{3434}, c_{3439}, c_{3453}, c_{3455}, c_{3458}]$	41.795414	12.276264	41.795203	12.276342
$[c_{3390}, c_{3424}, c_{3434}, c_{3439}, c_{3453}, c_{3455}, c_{3458}]$	41.795414	12.276264	41.795203	12.276342
$[c_{15118}, c_{15132}, c_{15137}, c_{15138}, c_{15141}, c_{15143}, c_{15145}]$	41.907971	12.491983	41.907512	12.49004
$[c_{15118}, c_{15131}, c_{15137}, c_{15139}, c_{15141}, c_{15144}, c_{15145}]$	41.907971	12.491983	41.907512	12.49004
$[c_{14896}, c_{14917}, c_{14920}, c_{14922}, c_{14926}, c_{14927}, c_{14929}, c_{14939}, c_{14941}]$	41.907921	12.491503	41.907502	12.490215
$[c_{14896}, c_{14917}, c_{14920}, c_{14922}, c_{14926}, c_{14927}, c_{14929}, c_{14940}, c_{14941}]$	41.907921	12.491503	41.907502	12.490215

Table 4.15: Some of the cluster evolution paths that start near the Rome airport in the Rome Taxis dataset.

Cluster Evolution Paths	
$[c_{10071}]$	$[c_{11105}]$
$[c_{10390}]$	$[c_{11187}]$
$[c_{10391}]$	$[c_{11195}]$
$[c_{10872}]$	$[c_{11196}]$
$[c_{11102}]$	$[c_{11197}]$

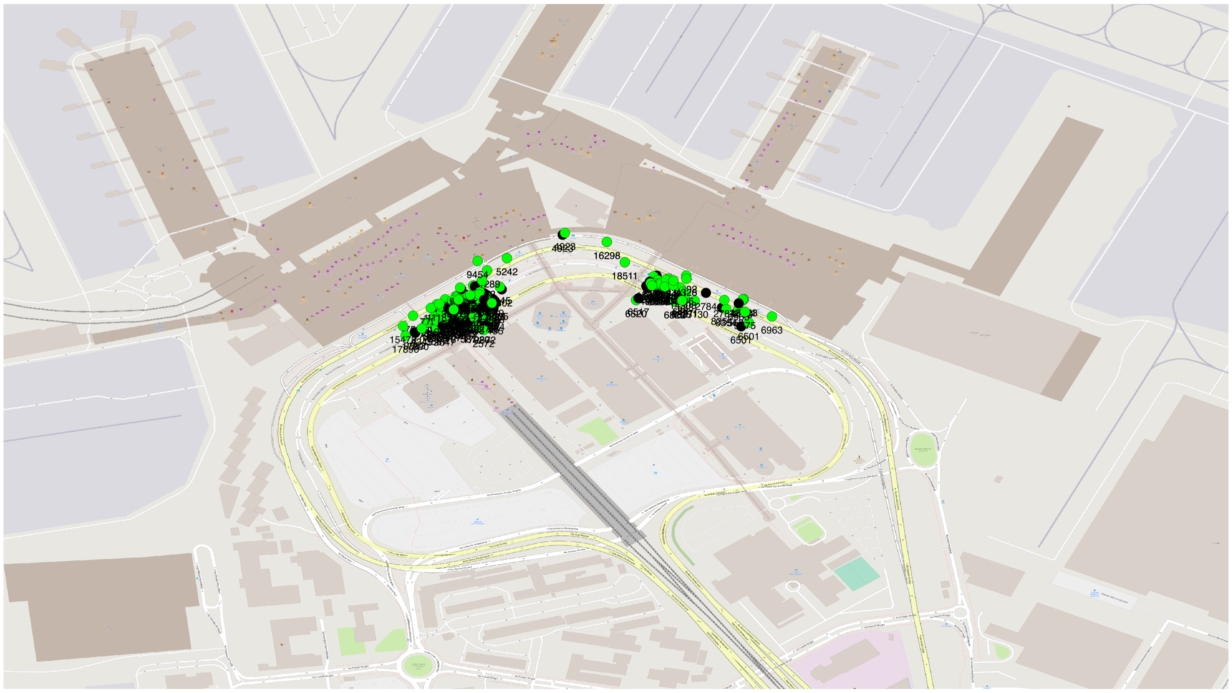


Figure 4.8: The start and end location of cluster paths near the Rome airport in the Rome Taxis dataset.

Show the location of every cluster change in all cluster evolution paths.

A cluster evolution path is a sequence of clusters c_{i,t_j}. At some point in this sequence, a cluster change happens, that is, the index i changes from, say, 1 to 4. This cluster change indicates a spatial-temporal cluster relationship, such as MERGE or C_ENTER. Analysis on cluster change in cluster evolution paths show the moment when or location where spatial-temporal objects decide to follow a different direction, which may include other clusters. When analyzing human behavior, this may represent a change of opinion or choice, when a group of individuals have their classification, and thus the clusters they belong to, changed. In the context, of spatial-temporal clusters, this means a better route, or the arrival or departure places. This query is executed on the Rome Taxis dataset.

Table 4.16 shows the result of this query. There are 621 locations where cluster changes happen. Note that not all locations are shown in the table because of space limitations. The first two lines of Table 4.16 show a split from spatial-temporal cluster c_{10032} into clusters c_{10043} and c_{10047}. Taking each cluster evolution path individually, there exists only one cluster change. For instance, in the path of the first line of the table, there is a cluster change from cluster c_{10032} to c_{10047}. The location where this cluster change happened is shown in the other columns of the table. Note that a cluster evolution path can have several cluster changes, as shown in line 3 of the table. Spatial-temporal cluster c_{16834} merges with c_{16835} to form cluster c_{16836}, which then eventually splits forming clusters c_{16840} and c_{16841}. The location of the cluster changes from c_{16834} to c_{16836} and from c_{16836} to c_{16840} of the cluster evolution path in line 3 are shown in the other columns of the table.

What are the cluster evolution paths that start in the morning and end in the afternoon?

Analyzing the start and end times of cluster evolution paths provides insights into their nature. For example, cluster evolution paths starting and ending at the commute time indicate a group of vehicles trying to find the best route to work.

This query is executed on the Rome Taxis dataset. The definitions of morning and afternoon are not precise. For this query, morning is defined as the time between 6 AM (inclusive) to 12 PM (exclusive), and afternoon is defined as the time between 12 PM (inclusive) to 6 PM (exclusive). The tool uses a 24-hour clock, so results for 6 PM will be shown as hour 18.

Table 4.17 shows the result of this query. There are 74 cluster evolution paths that start in the morning and end in the afternoon. The table does not show all 74 cluster

Table 4.16: Some cluster changes of all cluster evolution paths in the Rome Taxis dataset.

Cluster Evolution Paths	Latitude	Longitude
$[c_{10032}, c_{10043}]$	41.907799	12.4913470
$[c_{10032}, c_{10047}]$	41.9076365	12.4908065
$[c_{16834}, c_{16836}, c_{16840}]$	41.901389	12.501104
	41.901306	12.5009395
$[c_{16834}, c_{16836}, c_{16841}]$	41.901389	12.501104
	41.901432	12.4998215
$[c_{16835}, c_{16836}, c_{16840}]$	41.901814	12.501151
	41.901306	12.5009395
$[c_{16835}, c_{16836}, c_{16841}]$	41.901814	12.501151
	41.901432	12.4998215
$[c_{5805}, c_{5829}]$	41.907695	12.4906565
$[c_{5805}, c_{5830}, c_{5838}]$	41.9078325	12.491307
	41.907769	12.4912135
$[c_{5805}, c_{5830}, c_{5837}, c_{5841}]$	41.9078325	12.491307
	41.907578	12.490627
	41.907467	12.4904055
$[c_{5805}, c_{5830}, c_{5837}, c_{5842}]$	41.9078325	12.491307
	41.907578	12.490627
	41.9076515	12.491019

Table 4.17: Some of the cluster evolution paths in the Rome Taxis dataset that start in the morning and end in the afternoon.

Cluster Evolution Paths	Start Time	End Time
$[c_{848}]$	2014-02-03 11:08:01	2014-02-03 12:17:01
$[c_{12187}]$	2014-02-20 11:30:01	2014-02-20 12:12:01
$[c_{129}]$	2014-02-01 11:56:01	2014-02-01 12:03:01
$[c_{17561}]$	2014-02-28 11:58:01	2014-02-28 13:13:01
$[c_{3184}, c_{3196}]$	2014-02-06 11:46:01	2014-02-06 12:16:01
$[c_{3186}, c_{3196}]$	2014-02-06 11:47:01	2014-02-06 12:16:01
$[c_{5969}, c_{6018}, c_{6019}, c_{6033}]$	2014-02-11 11:34:01	2014-02-11 12:43:01
$[c_{5969}, c_{6017}, c_{6019}, c_{6033}]$	2014-02-11 11:34:01	2014-02-11 12:43:01
$[c_{1463}, c_{1475}, c_{1493}, c_{1499}, c_{1502}]$	2014-02-04 11:34:01	2014-02-04 12:02:01
$[c_{1463}, c_{1475}, c_{1493}, c_{1499}, c_{1503}]$	2014-02-04 11:34:01	2014-02-04 12:05:01
$[c_{1463}, c_{1475}, c_{1494}, c_{1499}, c_{1502}]$	2014-02-04 11:34:01	2014-02-04 12:02:01
$[c_{1463}, c_{1475}, c_{1494}, c_{1499}, c_{1503}]$	2014-02-04 11:34:01	2014-02-04 12:05:01

evolution paths because of space limitations. The table includes the start and end time for further analysis. Note that the cluster evolution paths in lines 1 and 4 of the table existed for more than one hour. The same can be said for the cluster evolution paths in lines 7 and 8 of the table in which a SPLIT and a MERGE happens. All clusters start between the hours 11 and 12 of the day and most end within an hour after the start.

Show all cluster evolution paths and the time they start and end.

A previous query searched for the start and end location of cluster evolution paths to give insights on the movement of clusters. This query searches for the start and end times of all cluster evolution paths in the dataset. While the previous query focuses on the spatial dimension of data, this query focuses on the temporal dimension of data. As before, the goal of the query is to include important information alongside the cluster evolution paths for further analysis. This query is executed on the Rome Taxi dataset.

Table 4.18 show the result of this query. A previous query showed that there exist 18,639 cluster evolution paths in the Rome Taxis dataset. It is not possible to represent all paths here because of space limitations, although the analysis does show results for all paths. Instead, Table 4.18 shows results for some cluster evolution paths, which are the same as a previously similar query, for comparison. The start and end timestamp can be integrated for an improved analysis.

What is the cluster evolution path with the greatest number of relationships?

In exploratory analysis of graph-based cluster evolution, it is important to query the dataset looking for outliers or special cases because they may hold important information of the phenomenon being analyzed. For example, in the case of cluster evolution, a path with a great number of relationships may indicate an interesting route in the city, or a common way to tour a city. This query is executed on the Rome Taxis dataset.

Table 4.19 shows the result of the query. The greatest number of relationships that any cluster evolution path has in the Rome Taxis dataset is 215 relationships. There are two cluster evolution paths that have these many relationships. These paths are very similar, except for a SPLIT into and a MERGE of clusters c_{9530} and c_{9531}. Further inspection reveals that the two cluster evolution paths exist in the parking lot of taxis near the airport of Rome. It is expected that many relationships happen in such places as taxis arrive and leave.

What cluster evolution paths have more (or less) than a given number of relationships?

The result of an analysis method becomes more specific when some restrictions are applied to the queries. This query is similar to the one explained earlier, but it expands the previous results to all cluster evolution paths that have at least (or at most) a given number of spatial-temporal cluster relationships.

This query is executed on the Rome Taxis dataset. This query searches for cluster evolution paths that have more than 100 relationships. The value of 100 relationships is chosen to demonstrate the query and can be changed to any desired value.

Table 4.20 shows the result of this query. There are 21 cluster evolution paths that have more than 100 spatial-temporal cluster relationships. The table also shows the number of relationships. Note that the cluster evolution path in the first line of this table has 102 relationships without ever having any cluster change.

Table 4.18: Some of the cluster evolution paths in the Rome Taxi dataset and the time they start and end.

Cluster Evolution Paths	Start Timestamp	End Timestamp
[c_{758},c_{759}]	2014-02-03 09:29:01	2014-02-03 09:33:01
[c_{14796},c_{14808}]	2014-02-24 18:38:01	2014-02-24 19:13:01
[c_{14807},c_{14808}]	2014-02-24 18:50:01	2014-02-24 19:13:01
[c_{15247},c_{15259}]	2014-02-25 11:07:01	2014-02-25 11:22:01
[c_{15256},c_{15259}]	2014-02-25 11:13:01	2014-02-25 11:22:01
[c_{752},c_{755},c_{759}]	2014-02-03 09:21:01	2014-02-03 09:33:01
[c_{754},c_{755},c_{759}]	2014-02-03 09:26:01	2014-02-03 09:33:01
[c_{798},c_{805},c_{807}]	2014-02-03 10:07:01	2014-02-03 10:15:01
[c_{3390},c_{3423},c_{3434}, c_{3439},c_{3453},c_{3455},c_{3458}]	2014-02-06 16:08:01	2014-02-06 20:14:01
[c_{3390},c_{3424},c_{3434}, c_{3439},c_{3453},c_{3455},c_{3458}]	2014-02-06 16:08:01	2014-02-06 20:14:01
[c_{15118},c_{15132},c_{15137}, c_{15138},c_{15141},c_{15143},c_{15145}]	2014-02-25 08:04:01	2014-02-25 09:00:01
[c_{15118},c_{15131},c_{15137}, c_{15139},c_{15141},c_{15144},c_{15145}]	2014-02-25 08:04:01	2014-02-25 09:00:01
[c_{14896},c_{14917},c_{14920},c_{14922}, c_{14926},c_{14927},c_{14929},c_{14939},c_{14941}]	2014-02-24 20:10:01	2014-02-24 21:31:01
[c_{14896},c_{14917},c_{14920},c_{14922}, c_{14926},c_{14927},c_{14929},c_{14940},c_{14941}]	2014-02-24 20:10:01	2014-02-24 21:31:01

Table 4.19: The cluster evolution path with the greatest number of relationships in the Rome Taxis dataset.

Cluster Evolution Path	Number of Relationships
[c_{9507},c_{9530},c_{9532},c_{9544},c_{9545},c_{9547}]	215
[c_{9507},c_{9531},c_{9532},c_{9544},c_{9545},c_{9547}]	215

Table 4.20: Cluster evolution paths that have more than 100 spatial-temporal cluster relationships in the Rome Taxis dataset.

Cluster Evolution Path	Number of Relationships
$[c_{4103}]$	102
$[c_{4921}]$	107
$[c_{8657}]$	109
$[c_{12466}]$	109
$[c_{9539}, c_{9544}, c_{9545}, c_{9547}]$	123
$[c_{9539}, c_{9544}, c_{9546}, c_{9547}]$	123
$[c_{9539}, c_{9544}, c_{9545}, c_{9547}]$	124
$[c_{3390}, c_{3423}, c_{3434}, c_{3438}, c_{3453}, c_{3456}, c_{3458}]$	125
$[c_{3390}, c_{3423}, c_{3434}, c_{3438}, c_{3453}, c_{3455}, c_{3458}]$	126
$[c_{3390}, c_{3423}, c_{3434}, c_{3439}, c_{3453}, c_{3456}, c_{3458}]$	127
$[c_{3390}, c_{3423}, c_{3434}, c_{3439}, c_{3453}, c_{3455}, c_{3458}]$	128
$[c_{3390}, c_{3424}, c_{3434}, c_{3438}, c_{3453}, c_{3456}, c_{3458}]$	128
$[c_{3390}, c_{3424}, c_{3434}, c_{3438}, c_{3453}, c_{3455}, c_{3458}]$	129
$[c_{3390}, c_{3424}, c_{3434}, c_{3439}, c_{3453}, c_{3456}, c_{3458}]$	130
$[c_{3390}, c_{3424}, c_{3434}, c_{3439}, c_{3453}, c_{3455}, c_{3458}]$	131
$[c_{9507}, c_{9531}, c_{9532}, c_{9544}, c_{9545}, c_{9547}]$	214
$[c_{9507}, c_{9531}, c_{9532}, c_{9544}, c_{9546}, c_{9547}]$	214
$[c_{9507}, c_{9530}, c_{9532}, c_{9544}, c_{9545}, c_{9547}]$	214
$[c_{9507}, c_{9530}, c_{9532}, c_{9544}, c_{9546}, c_{9547}]$	214
$[c_{9507}, c_{9531}, c_{9532}, c_{9544}, c_{9545}, c_{9547}]$	215
$[c_{9507}, c_{9530}, c_{9532}, c_{9544}, c_{9545}, c_{9547}]$	215

Table 4.21: The cluster evolution path with the greatest number of spatial-temporal objects in the Rome Taxis dataset.

Cluster Evolution Path	Number of Spatial-Temporal Objects
$[c_{8657}]$	12
$[c_{8904}]$	12

What is the cluster evolution path with the greatest number of spatial-temporal objects during its existence?

The analysis of the number of spatial-temporal objects contained in a cluster evolution path provides insights about concepts such as spatial-temporal service demand. An earlier query searched for the path with the greatest number of relationships. This query searches for the cluster evolution path with the greatest number of spatial-temporal objects.

This query is executed on the Rome Taxis dataset. The query ranks all cluster evolution paths based on the greatest number of spatial-temporal objects paths have at any point during their existence, not limited to the start or the end. The query then filters the results to show the cluster evolution path sought.

Table 4.21 shows the result of this query. The greatest number of spatial-temporal objects any cluster path in the Rome Taxis dataset has is 12. Two cluster evolution paths have these many spatial-temporal objects. Note that these cluster evolution paths do not include more than one cluster. Such a number of taxis clustered together in a place may not necessarily be a good indicator of demand. For example, a high number of taxis clustered in a parking lot at times different than rush hours may indicate low demand. Inspections on the time may help confirm this statement.

What is the cluster evolution path with the greatest average number of spatial-temporal objects during its existence?

Analyzing the number of spatial-temporal objects in cluster evolution paths is valuable and results such as the one in the previous query are a great starting point for more investigations. However, analyses methods that identify the greatest or lowest number of some property may be affected by peaks in the values being monitored. For example, in the previous query, cluster c_{8657} could have undergone a sudden and rapid increase of the

Table 4.22: The cluster evolution path with the greatest average number of spatial-temporal objects in the Rome Taxis dataset.

Cluster Evolution Path	Average Number of Spatial-Temporal Objects
$[c_{8657}]$	9
$[c_{8904}]$	9

number spatial-temporal objects followed by a quick decrease of the number. Although the investigation is still important, sometimes it is required to remove such outlier cases from the observations. One solution is to search for average values, which is the function of this query.

This query is executed on the Rome Taxis dataset. There are several ways to calculate the average number of spatial-temporal objects in cluster evolution paths. They can be averaged by time, by cluster, or by relationship. This query chooses to average spatial-temporal objects by the number of clusters in the cluster evolution path. In this case, the previous result, in Table 4.21 is also the result of this query, since the cluster evolution path with the greatest number of spatial-temporal objects has only one cluster. However, this query attempts to avoid peaks of growth or decay in the numbers being observed and find cluster evolution paths that keep a consistent number of spatial temporal objects during existence. Therefore, this query does not use the greatest number of spatial-temporal objects, but instead the difference between the greatest and the smallest number of spatial-temporal objects in a cluster evolution path.

Table 4.22 shows the result of this query. The cluster evolution paths in the result of the previous query appear as ones with the greatest average number of spatial-temporal objects per cluster in the Rome Taxis dataset. The two cluster evolution paths are further inspected, which consists of inspecting the only cluster they have. Cluster c_{8657} starts and ends at the taxi parking lot near the airport of Rome. Such elevated number of spatial-temporal objects is consistent with expectations. Cluster c_{8904} exists near Piazza Navona, a square in Italy famous for several art masterpieces by well-known Baroque artists. Both clusters exist on February 14, but the first exists from around 5 PM to almost 9 PM and the second one exists from 9 PM to 9:30 PM. Notice the importance of this square during Valentine's Day. Although interesting, the clusters do not share any spatial-temporal objects.

4.3 Case Study 2

This section details Case Study 2, and the ideas of ever-increasing or ever-decreasing regions in graph-based cluster evolution, average rate of change (AROC), and the connections with offer and demand of spatial-temporal services.

Graph-based cluster evolution can be analyzed by considering the cluster evolution paths that start from a spatial-temporal cluster. Recall that a cluster evolution path is a sequence of clusters c_{i,t_j} or a path in a graph. Each cluster c_{i,t_j} has several properties including the number of spatial-temporal objects in it, called the size of the cluster. Given these definitions, the ever-increasing region of a cluster evolution path is a subpath where the size of the clusters c_{i,t_j} either increases or stays the same when compared to the size of the cluster at the previous timestamp $c_{i,t_{j-1}}$. An ever-decreasing region is similar but having the size of clusters decreasing or staying the same. Note that these regions do not need to start at the first cluster of a cluster evolution path, and that a cluster-evolution path may have several ever-increasing or ever-decreasing regions. These regions can be represented by a pair of clusters, the one at the start of the region and another at the end. Alternatively, ever-increasing or decreasing regions can be represented by the timestamps at their start and end.

Ever-increasing or ever-decreasing regions can provide insights about the behavior of spatial-temporal clusters or the contained spatial-temporal objects especially when analyzed with other contextual information. For example, a cluster path with an ever-decreasing region indicates the place or time where spatial-temporal objects disperse in the road network, leaving the city center.

The identification of ever-increasing or ever-decreasing regions allows for the calculation of AROC in cluster size. This rate quantifies how fast a spatial-temporal cluster grows or shrinks. If c_{i,t_j} and c_{k,t_ℓ} are the clusters at the start and end of an ever-increasing or ever-decreasing region, respectively, and $|\cdot|$ calculates the size of a cluster, then AROC can be calculated as follows, which simply divides the difference in size by the difference in time.

$$AROC = \frac{|c_{k,t_\ell}| - |c_{i,t_j}|}{t_\ell - t_j}$$

Positive values for AROC mean that the cluster grows and negative values for AROC mean that the cluster shrinks.

Case Study 2 identifies the lengthiest ever-increasing region of all cluster evolution paths in the Rome Taxis dataset. Then, two cluster evolution paths, one with a short ever-increasing region and another with a long ever-increasing region, are selected for further analysis. Their growth pattern is plotted and analyzed. The query to identify the longest ever-increasing regions of all cluster evolution paths in the dataset is written in Cypher, the graph query language from Neo4j, and its code is available in Appendix B.2.

Table 4.23 shows the result of this query. There are 7,666 longest ever-increasing regions in the Rome Taxis dataset. The table does not include all 7,666 regions because of space limitations, but some regions are shown for a discussion of the results. The number of regions is smaller than the total of 18,639 cluster evolution paths in the dataset found in a query in the previous section for two reasons. First, if a cluster evolution path has more than one ever-increasing region, then the query selects the longest one with respect to the number of spatial-temporal cluster relationships. If there is more than one, then the query selects all of them. Second, some cluster evolution paths are short, ending at the timestamp where they started. This does not produce any subpath for the query to examine. Table 4.23 contains the cluster evolution path in the first column, followed by information about the identified ever-increasing region: the start and end cluster, the start and end timestamp, and the start and end size.

The map in Figure 4.9 shows the locations in green where the ever-increasing regions start. The map also shows the locations in black where these regions end, but some are below the symbols for start locations. The overall map shows that these regions start in several places in Rome, including the Rome international airport, at the bottom left part of the figure. The overall map does not include details. For this reason, part of the overall map is highlighted. The detailed map shows the center of Rome and several of the locations where ever-increasing regions start and end. Note that there is no particular pattern to the start or end locations of the regions, as they start or end in several different places, including tourism sites, terminals, or regular taxi stands in the city.

Consider the cluster evolution path in the first line of Table 4.23. It contains only the spatial-temporal cluster c_{15995}. Its start and end cluster, therefore, are the cluster c_{15995} as shown in the second and third columns of the table. The identified ever-increasing region starts at 10:30 AM and ends at 11:18 AM. At the start, the cluster contains three spatial-temporal objects and it ends with seven.

It is interesting to note that the largest end size observed is 12. This does not mean that the largest cluster in the dataset has at most 12 spatial-temporal objects. It does mean that, among all ever-increasing regions in this dataset, the largest one has at most 12 spatial-temporal objects. A cluster formed by 20 spatial-temporal objects grouped

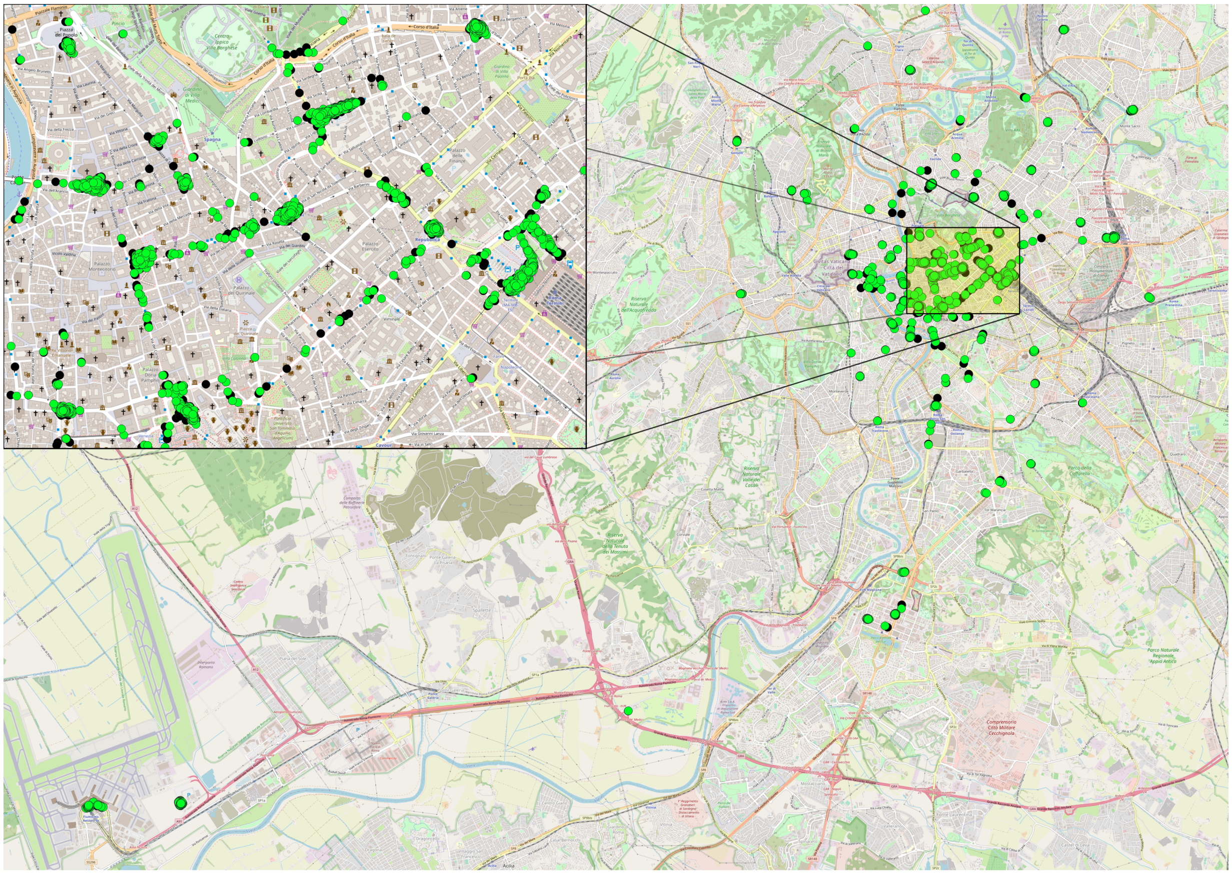


Figure 4.9: The locations where ever-increasing regions of all cluster evolution paths start, in green, and end, in black, for Case Study 2.

Table 4.23: The ever-increasing regions of every cluster evolution path in the Rome Taxis dataset.

Cluster Evolution Path	Start Cluster	End Cluster	Start Timestamp	End Timestamp	Start Size	End Size
$[c_{15995}]$	c_{15995}	c_{15995}	2014-02-26 10:30:01	2014-02-26 11:18:01	3	7
$[c_{7442}]$	c_{7442}	c_{7442}	2014-02-13 06:26:01	2014-02-13 06:34:01	4	8
$[c_{15854}, c_{15871}]$	c_{15854}	c_{15854}	2014-02-26 06:33:01	2014-02-26 06:39:01	5	8
$[c_{6919}, c_{6923}, c_{6928}]$	c_{6919}	c_{6919}	2014-02-12 13:31:01	2014-02-12 13:32:01	3	4
$[c_{6919}, c_{6923}, c_{6928}]$	c_{6923}	c_{6923}	2014-02-12 13:33:01	2014-02-12 13:34:01	3	7
$[c_{6921}, c_{6923}, c_{6928}]$	c_{6921}	c_{6923}	2014-02-12 13:32:01	2014-02-12 13:34:01	3	7
$[c_{1280}]$	c_{1280}	c_{1280}	2014-02-04 05:49:01	2014-02-04 07:07:01	3	9
$[c_{14228}, c_{14231}]$	c_{14228}	c_{14228}	2014-02-24 06:15:01	2014-02-24 06:23:01	3	10

together and decreasing in size as time passes would not be listed in the table because there is no ever-increasing region, despite having more spatial-temporal clusters. However, it happens that the largest size of all spatial-temporal clusters in the dataset is 12. This indicates that taxi drivers are aware of the size of the clusters and avoid clusters composed of more than 12 taxis. In the analyses below, two cluster evolution paths are selected. The largest size of these cluster evolution paths is not greater than 10 as the figures show. This number is not chosen, but it may be a natural consequence of the reason explained here.

Further inspection of the results shows that the cluster evolution path containing the spatial-temporal cluster c_{1280} and the other path containing the clusters c_{14228} and c_{14231} are one of the shortest and one of the longest paths in the dataset. They are selected for a detailed analysis. The first cluster evolution path happens at the taxi parking spot near the Rome airport and the second path happens at the street Via Boncompagni, where many luxury hotels are located. Both end near the locations where they started.

Figure 4.10 shows a plot of the size per minute of cluster c_{1280} in the ever-increasing region. In the figure, the x-axis starts at the time the ever-increasing region starts, which is the same time cluster c_{1280} starts, which is 2014-02-24 05:49:01. Other points on the axis show the amount of time, in minutes, that has elapsed since this start time. The plot shows one small increase in the number of spatial-temporal objects contained in cluster c_{1280}, followed by a steady moment, which is followed by a high increase. The plot is divided into these three sections and the AROC in cluster size is calculated for each section. Results are shown in the plot. The first section, with a small increase, has an AROC of 1 spatial-temporal object per minute, represented as 1 sto/min, while the second section has an AROC of 0 sto/min. These are trivial results. The third section yields more interesting

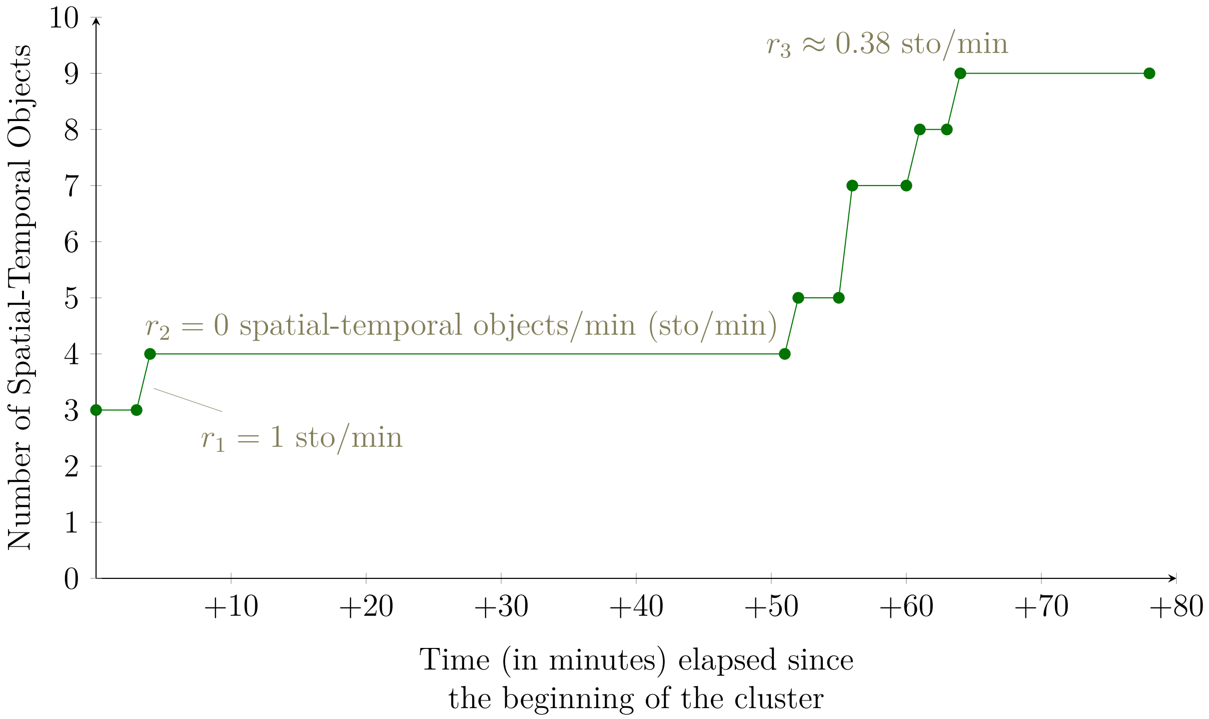


Figure 4.10: A plot of the size of cluster c_{1280} per minute in the ever-increasing region.

calculations. In this section, the AROC is calculated from the x value of +51 to +64 and is 0.38 *sto/min*. This confirms a slow growth of the cluster and of the cluster evolution path containing it.

When analyzing the plot and the spatial and the temporal dimension of the cluster, it is possible to conclude that cluster c_{1280} is formed by taxis waiting for passengers to take them to their destinations. However, no passenger needed a taxi at that moment. For example, since no relationship related to spatial-temporal objects leaving the cluster is detected, the three taxis that formed the cluster wait at least 1 hour and 20 minutes for the first passenger. It can be concluded that this is a situation of low demand and high offer. Analyses such as this can have a great impact on the price of spatial-temporal services at certain locations.

Figure 4.11 shows a plot of the size of cluster c_{14228} per minute in the ever-increasing region. In the figure, the x-axis starts at the time the ever-increasing region starts, which is the same time at which cluster c_{14228} starts, namely 2014-02-24 06:15:01. Other points on the axis show the amount of time, in minutes, elapsed since this start time. The plot also shows a small increase, a steady moment, and a longer increase, and there can be a comparison with the plot in Figure 4.10. However, in general, the plot shows a steady increase in the number of spatial-temporal objects and dividing the plot into sections does not seem to add value. Instead, the graph is analyzed as a whole. An AROC of 0.875 sto/min is calculated for the entire graph and it is shown in the plot. If the small steady region at the top of the plot is excluded from the analysis, that is if only the data from +0 to +7 is considered, then the AROC is exactly 1 sto/min. Considering that this ever-increasing region takes less than 10 minutes, this result shows and confirms a rapid growth of the cluster and the cluster evolution path that contains it.

When analyzing the plot and the spatial and temporal information of the data about the cluster, that is its location and timestamp, it is possible to conclude that cluster c_{14228} is formed by several taxi drivers turning on their GPS devices at the moment they start to work, beginning to record spatial-temporal measurements and become ready for passengers. The fact that they share a common time to start working explains the rapid growth of the cluster. It is possible therefore, to identify the start time of the work shift. Analyses such as this one have impacts on the readiness to provide spatial-temporal services as they quantify it, leading to other types of analysis, such as those in the fields of economy or health, that can assess, for example, the wages or health of taxi drivers.

4.4 Case Study 3

This section details Case Study 3. It describes what cluster evolution path similarities are, how they can be calculated, and some analysis opportunities that can be considered.

Similarity is a fundamental concept in ML, as analysis methods compare features of a new data point under examination with features of other data points for classification or clustering tasks. However, in some cases, defining similarity can be challenging. For instance, it can be trivial to define a similarity metric for house prices, but defining a similarity metric for faces requires more studies. In general, though, a similarity value is calculated based on quantifiable measurements about the nature or property of the data points. One such property is the sequential nature of data. Words can be viewed as a sequence of letters and a list of student grades can be interpreted as a sequence of numbers. A simple way to calculate similarity between two sequences is to sum the differences between the values on each position in the sequence, which produces a distance value, and take its reciprocal. In the case of two words, or sequences of letters, the distance can be the difference between the position of the letters in the alphabet. This simple method, however, assumes that the sequences have the same length. In the context of

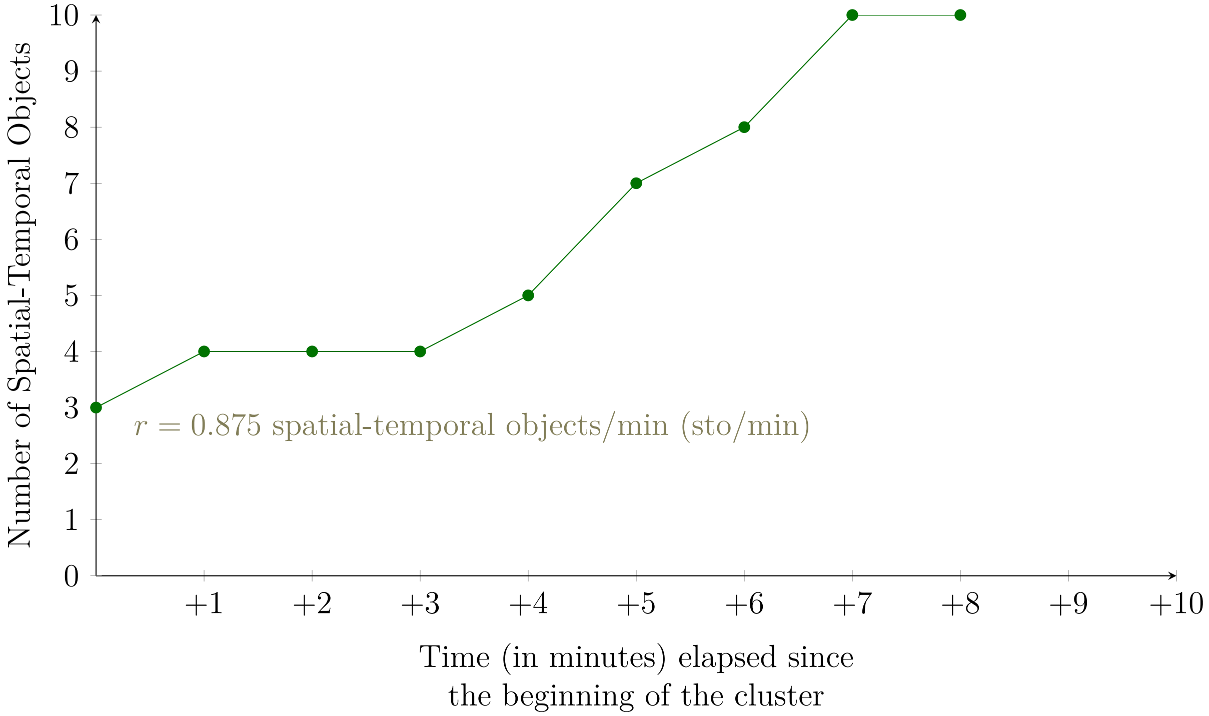


Figure 4.11: A plot of the size of cluster c_{14228} at every minute.

graph-based cluster evolution, sequences are derived from cluster evolution paths and their lengths may differ.

Dynamic Time Warping (DTW) [9, 117, 32] is an algorithm to calculate the similarity between two temporal sequences of possibly different lengths. One common application of DTW is speech recognition. The DTW algorithm proceeds by iterating over the two sequences and choosing the minimum difference between the value stored current or next position in the first sequence and the value stored in the current or next position in the second sequence. For DTW to be used in the context of graph-based cluster evolution, some adaptations are required. First, cluster evolution paths are formed by clusters of spatial-temporal data c_{i,t_j}. Previous sections have described a way to compare two such clusters using an extended version of Jaccard similarity, but it assumed that the clusters are at consecutive timestamps. When comparing clusters in two cluster evolution paths, this assumption does not hold. In fact, the clusters may not even share a single spatial-temporal object. Calculating a "difference" between these clusters becomes a challenging task. The solution is to derive two new sequences based on the properties of the clusters in the cluster evolution paths. For example, a new sequence can contain the number of spatial-temporal objects at the timestamp of the cluster, that is the size of the cluster, at each position. Comparisons between these new sequences then become the calculation of difference between these sizes, allowing DTW to be used.

Case Study 3 analyzes cluster evolution paths by calculating their similarity, and then shows useful results that can be derived from this calculation, before suggesting novel analysis methods that can use the similarity value. Data for this case study comes from the T-Drive dataset, more specifically of cluster evolution paths that start near the Beijing Capital International Airport, around Terminals 1 and 2. To calculate the similarity value between two cluster evolution paths, two new sequences are derived containing the number of spatial-temporal objects that each cluster in the path contains and, then, the DTW algorithm is used to compare these two new sequences. Time is divided into 24 slots, each representing one hour, and comparison happens between cluster evolution paths within the time slots, but regardless of the day. The query to identify cluster evolution paths in the dataset and to create part of the results table are written in Cypher, the graph query language from Neo4j, and their code are available in Appendix B.3. The query for the calculation of the similarity is written in Python using libraries for pandas[6], a Python Data Analysis library, and fastdtw[7][8], an implementation of FastDTW [111, 112], which is a fast implementation of the DTW algorithm with optimal linear time and memory

[6] https://pandas.pydata.org

[7] https://pypi.org/project/fastdtw/

[8] https://github.com/slaypni/fastdtw

complexity.

Some considerations are discussed. In the entire T-Drive dataset, there are 244,737 clusters that can start cluster evolution paths. In an attempt to reduce the number of these clusters, cluster evolution paths that exists for just one timestamp, and those that start in a timestamp and end on the next one, are filtered out of the analysis. This reduces the number of clusters that can start cluster evolution paths to 5404. These clusters now should be filtered to capture only those which are near the Beijing airport. The map in Figure 4.12 shows the airport building, its terminals, and the locations of clusters that start cluster evolution paths near the airport. In the middle of the figure, the road network forms a rectangle where cars can access terminals 1 and 2. These terminals are located at the top and left sides of this rectangle. The locations of clusters that start cluster evolution paths are represented with dots on the map. The difference in colors of the dots is explained shortly. Selecting the relevant locations can be done by using the latitude and longitude coordinates of the airport and using a radius that spans from the location represented by these coordinates, creating a circular region whose contained dots are selected. However, based on the geometry of the roads near the airport, a circular selection would include dots that are relatively far from the airport terminals, that is the clusters around the right or bottom sides of the rectangle. Instead, selection of the relevant dots, representing the location of clusters that start cluster evolution paths, is done manually, based on the visualization of these dots. In Figure 4.12, the dots in yellow, and the respective locations they represent, are selected for the Case Study 3.

There are 175 different locations selected. The locations in the Figure 4.12 come from the spatial-temporal clusters that start cluster evolution paths. Therefore, selecting locations effectively selects the clusters for the analysis. Cluster evolution paths from these clusters are identified and a new sequence containing the size of the cluster at the timestamps they have spatial-temporal cluster relationships is created for each path. Cluster evolution paths are grouped based on their starting hour of the day, their new sequences are compared using the FastDTW algorithm. Results are as follows.

Tables 4.24 and 4.25 show the results of this analysis. Data in the tables is grouped by the hour of the day as shown in the first column. It uses a 24-hour clock. For instance, the row for hour 8 effectively relates to the time from 8:00:00 AM to 8:59:59 AM. In Table 4.24, the second column shows the list of clusters that can start cluster evolution paths near the Beijing airport for a specific hour. Note that this table does not show any cluster evolution path. Table 4.25 shows the number of clusters that can start cluster evolution paths in that hour in the second column. The third column relates to an aggregated value over all the cluster evolution paths starting on the hour for that row. Specifically, it shows the number of different spatial-temporal clusters that exist in all cluster evolution paths

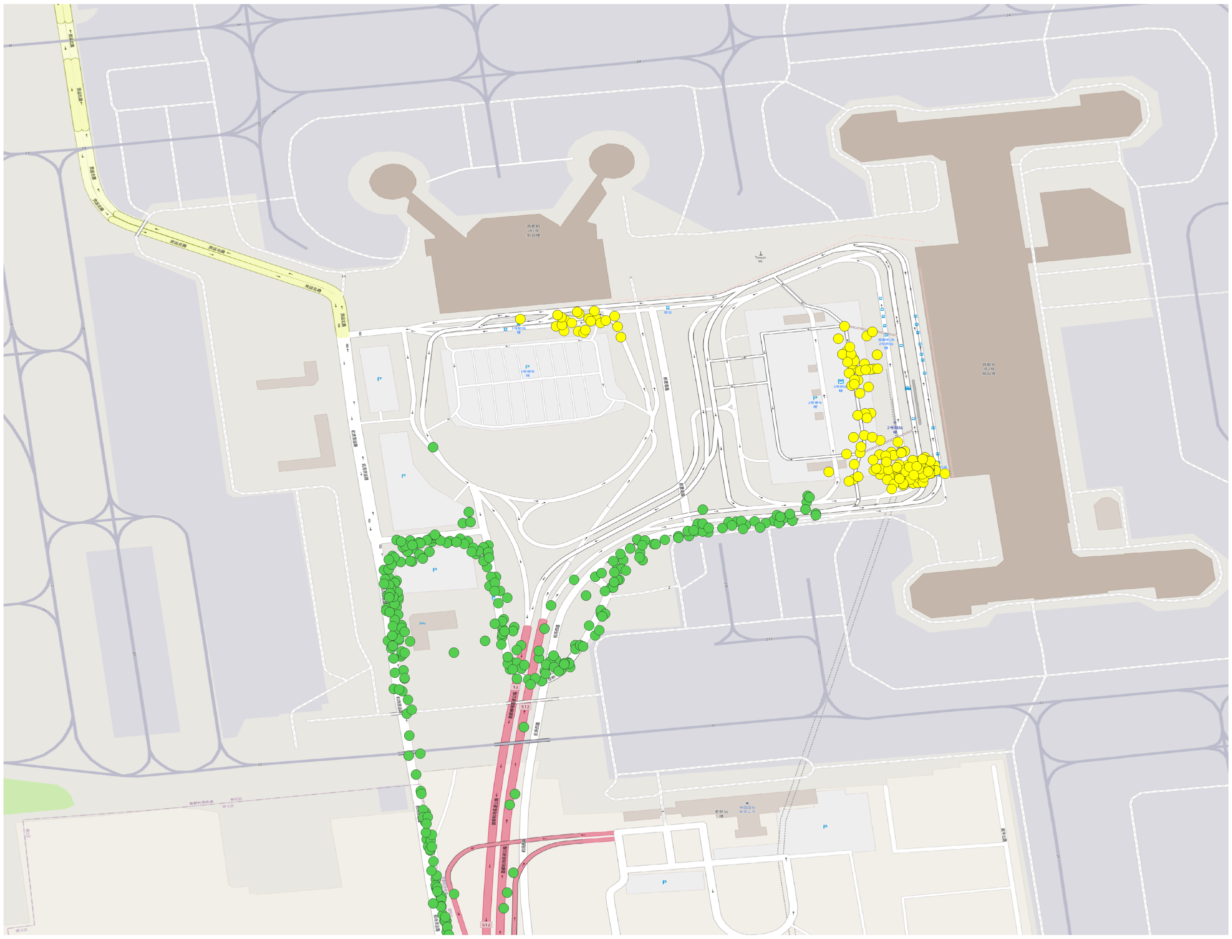


Figure 4.12: The many locations where cluster evolution paths start near the Beijing airport. The locations in yellow are the selected ones for Case Study 3.

for the hour of a specific row. The fourth column has the number of cluster evolution paths that start in the hour of a specific row. The fifth column contains the number of cluster evolution paths performed. Notice that this column can be calculated from the value in the fourth column since the number of comparisons is the combination of the number of cluster evolution paths taken in pairs, or $C(n, k)$, where n is the number of cluster evolution paths and $k = 2$. FastDTW calculates a distance, or dissimilarity, value. As comparisons happen, the sum of distance values is updated. The sixth column shows this sum of distance values. Finally, the seventh value shows the effective similarity value for that hour of the day. However, some considerations are discussed. In general, a similarity value is the reciprocal of the distance value. However, calculating the reciprocal of the value in the sixth column would create similarity values for different hours that are not comparable because the number of cluster evolution paths, and therefore the number of comparisons, is different for each hour. To solve this, the similarity value is averaged by dividing it by the number of cluster evolution path comparisons. This normalizes the similarity value and allows comparison between rows.

Among the results in Table 4.25, note that no cluster evolution paths are detected at the hours 5 or 6 of the day and, therefore, no similarity value can be calculated. A similar situation happens at hour 7, where only one cluster evolution path is detected. The lack of cluster evolution paths starting near the airport at hours 5 or 6 can be explained by the work hours. Naturally, taxi drivers stop working in the evening and restart working in the morning. This impacts the number of cluster paths formed. Additionally, as explained previously, cluster evolution paths whose existence is short are filtered out. Some of these clusters can be listed on hours 5 or 6, but are not analyzed because of this restriction. Note also that, based on the number of clusters in all cluster evolution paths for an hour, as in the fourth column, hours 8 and 16 rank at the top with 46 and 57 clusters respectively. Lastly, note the number of cluster evolution paths detected starting at hours 15 and 16 and the consequent number of comparisons. Results for these two rows are challenging to calculate and computations are divided into parts and then aggregated.

The results in Table 4.25 are plotted in Figure 4.13 for an improved visualization. The figure shows six plots, each for one of spatial-temporal clusters that start cluster evolution paths, the number of spatial-temporal clusters in all cluster evolution paths, the number of cluster evolution paths, the number of cluster evolution path comparisons, the sum of distance values, and the similarity value. The plots are aligned based on the hour of the day, in the x-axis, so that they can be compared. Note also that some values are very high and showing them would reduce the details of other values that are not so high. In the interest of displaying more information in the plots, the plots of these values are not shown, but their exact numerical value and a mark on the hour they happen are shown in

Table 4.24: The spatial-temporal clusters that start cluster evolution paths around the Beijing airport alongside the hour they are formed.

Hour	Spatial-Temporal Clusters that Start Cluster Evolution Paths
0	c_{41834},c_{104054},c_{166213},c_{213306},c_{239066}
1	c_{43160},c_{43316},c_{44241},c_{44625},c_{105586},c_{106508},c_{167169},c_{167566},c_{213856}
2	c_{44881},c_{106984},c_{107009},c_{107329},c_{213989}
3	c_{107860},c_{108156},c_{168581},c_{214257}
4	c_{46295},c_{46435},c_{214581}
5	-
6	-
7	c_{109437}
8	c_{47459},c_{47779},c_{47879},c_{48080},c_{109801},c_{170675},c_{170824},c_{215011},c_{215139},c_{215140},c_{215239},c_{215327},c_{229419}
9	c_{48662},c_{49244},c_{110580},c_{110710},c_{112032},c_{170979},c_{171800},c_{171945},c_{215511},c_{215693},c_{240080}
10	c_{50393},c_{50417},c_{51045},c_{52042},c_{115074},c_{229788}
11	c_{55351},c_{116911},c_{117628},c_{176553},c_{176734},c_{177109},c_{217116},c_{217658},c_{230205}
12	c_{57052},c_{58723},c_{120405},c_{181357},c_{218880},c_{230739},c_{242385}
13	c_{702},c_{61029},c_{62159},c_{63638},c_{124175},c_{126225},c_{127797},c_{184405},c_{184512},c_{220424},c_{231146},c_{242977},c_{243746}
14	c_{4131},c_{4430},c_{4463},c_{65752},c_{186010},c_{187017},c_{187409},c_{221012},c_{244908}
15	c_{7177},c_{7366},c_{7530},c_{8414},c_{69171},c_{69474},c_{71594},c_{189344}
16	c_{11846},c_{72463},c_{72505},c_{74161},c_{75991},c_{136500},c_{138982},c_{192196},c_{223119},c_{234148},c_{246231},c_{246938}
17	c_{14441},c_{76250},c_{76346},c_{77187},c_{77523},c_{78860},c_{80113},c_{144171},c_{196190},c_{197621},c_{224179},c_{234621},c_{247757}
18	c_{18263},c_{145633},c_{200096},c_{225739},c_{225973}
19	c_{24226},c_{25812},c_{85714},c_{86310},c_{87096},c_{88388},c_{150049},c_{152640},c_{202530},c_{204574},c_{226418},c_{236838}
20	c_{27097},c_{27494},c_{29246},c_{89645},c_{90704},c_{154717},c_{205729},c_{227611}
21	c_{33652},c_{34745},c_{95667},c_{158180},c_{158708},c_{208777},c_{227707},c_{237883},c_{238119}
22	c_{36534},c_{98326},c_{98479},c_{98964},c_{161382},c_{210929},c_{238543}
23	c_{40931},c_{162750},c_{163471},c_{164324},c_{211505},c_{211924}

Table 4.25: A summary of the results of Case Study 3.

Hour	Number of Spatial-Temporal Clusters that Start Cluster Evolution Paths	Number of Spatial-Temporal Clusters in All Cluster Evolution Paths	Number of Cluster Evolution Paths	Number of Cluster Evolution Path Comparisons	Sum of Distance Values	Similarity Values
0	5	5	5	10	96	0.10417
1	9	13	16	120	1,352	0.08876
2	5	9	19	171	1,309	0.13063
3	4	5	5	10	269	0.03717
4	3	3	3	3	14	0.21429
5	0	0	0	0	0	0.00000
6	0	0	0	0	0	0.00000
7	1	1	1	0	0	0.00000
8	13	46	188	17,578	451,674	0.03892
9	11	39	136	9,180	365,521	0.02511
10	6	23	583	169,653	1,989,121	0.08529
11	9	14	16	120	1,497	0.08016
12	7	32	718	257,403	10,184,999	0.02527
13	13	19	38	703	14,859	0.04731
14	9	20	28	378	9,119	0.04145
15	8	42	38,241	731,167,920	26,497,775,571	0.02759
16	12	57	9,503	45,148,753	1,934,382,580	0.02334
17	13	35	334	55,611	1,384,936	0.04015
18	5	7	6	15	219	0.06849
19	12	13	12	66	801	0.08240
20	8	13	19	171	2,232	0.07661
21	9	13	16	120	5,357	0.02240
22	7	9	9	36	483	0.07453
23	6	9	8	28	195	0.14359

the plots.

Based on the number of clusters formed at cluster evolution paths, shown in the second plot in Figure 4.13, it is possible to conclude that the hours 8 and 16 are the busiest hours and hours 5, 6, and 7 are the least busy hours at the Beijing airport. This is consistent with what is expected in most airports, whose busiest hours happen sometime from 8am to 6pm. The reason why hours 8 and 16 are the busiest hours in this airport can be explained by the preference of passengers for morning or early evening flights. The price of flight tickets is the main reason as early morning or late afternoon flights tend to be cheaper, but this is not the only one. Long international flights happen at night when possible, and passengers are advised to arrive early for these flights. This may explain part of the high number of clusters in hour 16. Moreover, several of these flights arrive at their destinations in the morning of the following day, which explains part of the results for hour 8.

The third plot in Figure 4.13 shows the elevated number of cluster evolution paths that start at hours 15 and 16. The plot also shows that hours 10 and 12 have a high number of cluster evolution paths. This usually happens when spatial-temporal clusters have several relationships with other clusters around them. Depending on when these relationships happen, several other cluster evolution paths can be formed. Moreover, when spatial-temporal clusters have several relationships with clusters around them, it means that there are many clusters nearby and that clusters exist for a long period of time. The main conclusion of this plot is that, at hours 10 and 12, but specially at hours 15 and 16, there are many spatial-temporal clusters of taxis around the Beijing airport, these clusters are close enough such that many relationships happen, and they form longer cluster evolution paths. Further analysis of the data about spatial-temporal objects of these paths can indicate, for example, the most profitable terminal or position to wait for passengers.

Figure 4.13 shows, in its sixth plot, the average similarity value of cluster evolution paths for each hour of the day. Note that hours 4 and 23 have the highest similarity. The comparison between the average similarity value, in the sixth plot, with the number of cluster evolution paths, in the third plot, yields interesting results. Note that data is inversely proportional. In other words, the greater the number of cluster evolution paths in an hour, the lower the overall similarity between these paths. This indicates that cluster evolution paths are developing different, at least with respect to the size of their spatial-temporal clusters. The correlation between these two sets of data is approximately -0.182. When comparing the average similarity of cluster evolution paths in an hour of the day and the number of clusters in all cluster evolution paths in that hour, the inverse correlation is even more explicit, with a value of approximately -0.297. Comparisons between different times of the day are to be carried out and may uncover additional insights.

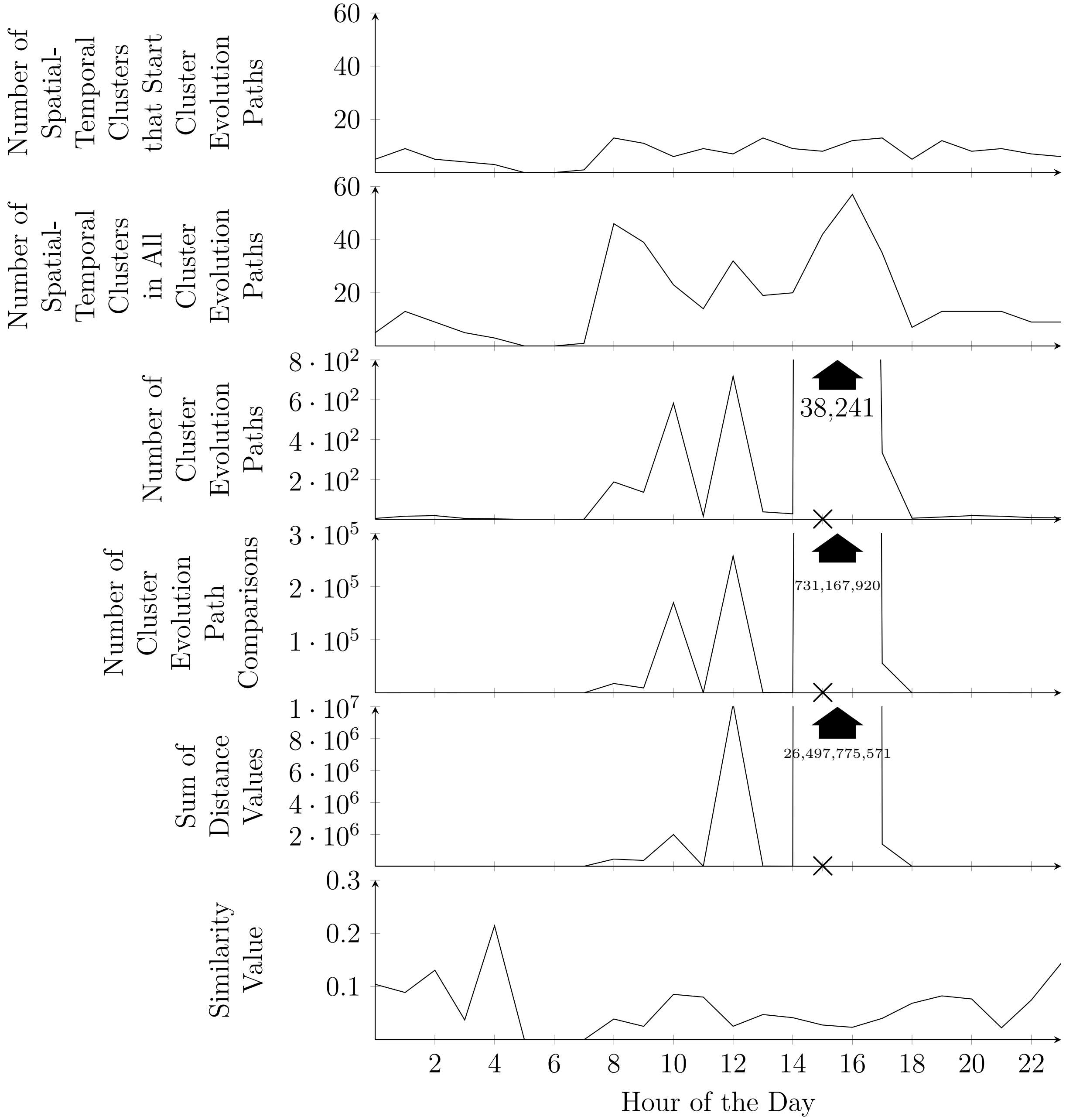


Figure 4.13: The plot of six metrics about cluster evolution paths near the Beijing airport in the T-Drive dataset.

As stated at the beginning of this section, a similarity value is essential for many ML analysis methods. The calculation of a similarity value between cluster evolution paths allows for several other analysis tasks. For example, given that one driver knows how to avoid traffic jams, or bad roads, or any other traffic problems in a city, the cluster evolution path, that is the sequence of spatial-temporal clusters, in which this driver participates is deemed favorable. Similar cluster evolution paths can then be found, perhaps in other cities, or be discovered. ML tasks, such as classification or clustering, are enabled for cluster evolution as well as other similarity-based analysis methods.

4.5 Case Study 4

This section describes the Case Study 4. The section discusses important analysis points related to graph-based cluster evolution, such as the direction and distance of movement, the distribution of cluster evolution paths per region or day, and temporal aspects of such paths.

Clusters can be used for several tasks, including classification or outlier detection. Therefore, the presence and identification of these clusters can bring many benefits to interested parties, depending on the domain. In the spatial-temporal domain, clusters can be used to identify people or vehicles that move together, thus sharing some characteristics, which assists in the classification of outlier detection tasks. However, some spatial-temporal clusters, such as those of taxis, are not always desirable, whether stopped or moving. For instance, taxis clustered stopped together in a cluster may indicate idle taxis or a traffic jam. This could produce higher costs for the taxi driver, the taxi company, and eventually the passenger. In addition, taxis moving together in a cluster may indicate car-sharing or ridesharing opportunities, especially if the cluster exists for a long time. Ways to analyze the movement of spatial-temporal clusters and, in a broader sense, the cluster evolution paths that they eventually form has the potential to bring important improvements to a city.

In graph-based cluster evolution, cluster evolution paths are formed when spatial-temporal objects group together, creating clusters, and these clusters have relationships with other clusters. Cluster evolution paths have well-defined start and end points as well as other characteristics, such as the number of spatial-temporal clusters or objects in a cluster, ever-increasing or ever-decreasing regions, and a similarity with other cluster evolution paths. Cluster evolution paths contain spatial-temporal clusters, and these clusters move. It is then possible to analyze the movement that a cluster evolution path describes. Analyzing features such as the direction, distance, time, region of the movement are novel aspects of cluster evolution paths that contain valuable data to be explored. In addition, analyzing a single cluster evolution path can uncover novel results, but graph-based cluster evolution allows for an aggregated analysis of all cluster evolution paths either in a day, region, or in the entire data set. This enables novel perspectives on spatial-temporal data and allows for many opportunities for improvements, such as improvements to the road network leading to crowded regions of a city or improvements to the public transportation system at given days or hours when demand is high.

Case Study 4 analyzes the movement of cluster evolution paths starting at specific points. The analysis takes into consideration the direction, distance, time, and region of the movement and produces aggregated results that can be used for improvements in several aspects of a city, such as city planning, its traffic, or its road network.

The data used in the analysis of Case Study 4 comes from the T-Drive dataset, which contains spatial-temporal data about taxis in Beijing, China, in 2008, the year in which the city hosted the Olympic Games. The city of Beijing has four main train stations, one on each side of the main square. Because of their locations, and to simplify the names, they are called here Beijing North, Beijing South, Beijing West, and Beijing East train stations. The analysis in Case Study 4 considers cluster evolution paths that start near the Beijing West and Beijing East stations.

Some limitations are considered and discussed. Unfortunately, there are not enough cluster evolution paths starting near the Beijing North and Beijing South train stations for analysis, thereby impacting aggregated results. For this reason, only cluster evolution paths that start near the Beijing West and Beijing East stations are considered. In addition, in an attempt to identify relevant cluster evolution paths, some paths are filtered out. First, cluster evolution paths that start and end at the same timestamp or that start at some timestamp t_j and end at the next timestamp t_{j+1} are not considered for analysis because of their brief existence. Second, the analysis seeks cluster evolution paths that moved at least 500 meters from their start location and does not consider other cluster evolution paths. The number 500 is arbitrary and is chosen to make sure that the movement of cluster evolution paths are observed and that interesting phenomena are analyzed.

In summary, Case Study 4 analyzes the movement of cluster evolution paths that start near the Beijing West and Beijing East stations that moved for at least 500 meters in the T-Drive dataset. More specifically, the direction of movement, based on cardinal points of the compass and a distribution of the movement by day and region are calculated and discussed. In addition, analysis on the time of cluster evolution path formation and the distance traveled is performed. The four analysis questions below guide the results and the conclusions.

1. What is the general destination of the cluster evolution paths?
2. What is the distribution of cluster evolution paths by region and by day?
3. What time of the day are clusters evolution paths forming?
4. What is the largest distance traveled in the movements represented by cluster evolution paths formed near each train station?

There are 244,737 locations where cluster evolution paths start in the entire T-Drive dataset, but this value reduces to 5,404 after some of the limitations discussed. Figures 4.14 and 4.15 show the two train stations in Beijing and some locations where cluster evolution paths start. In this analysis "near" or "around" a specific location is not precise. Cluster evolution paths near the two train stations are manually selected using subjective best judgment. In the maps in Figures 4.14 and 4.15, the locations marked in yellow are selected for analysis. The data about the location where cluster evolution paths start comes from the first cluster of each path. Thus, selecting locations effectively means selecting the first cluster of each cluster evolution path.

A total of 116 clusters that start cluster evolution paths exist near the Beijing West train station, but only 8 of these clusters developed 26 cluster evolution paths that describe movements of at least 500 meters. The situation is different at the other train station. A total of 152 clusters that start cluster evolution paths exist near the Beijing East train station, and 59 of these clusters developed an expressive number of 35,002 cluster evolution paths that describe movements of at least 500 meters.

A conclusion can be drawn from these figures. In the case for the Beijing West train station, very few cluster evolution paths describe movements of at least 500 meters, indicating that spatial-temporal clusters tend to disperse as soon as they move away from the train station. In the case for the Beijing East train station, almost half of the clusters that start cluster evolution paths generated a significant number of paths that represent movements of at least 500 meters, which indicates that, for example, car-sharing or ride sharing should be considered.

The remainder of this section discusses the results and conclusions for each of the four analysis questions described.

What is the general destination of the cluster evolution paths?

The direction of the movement described by cluster evolution paths can be described based on the cardinal points: north (N), south (S), west (W), and east (E). To express the

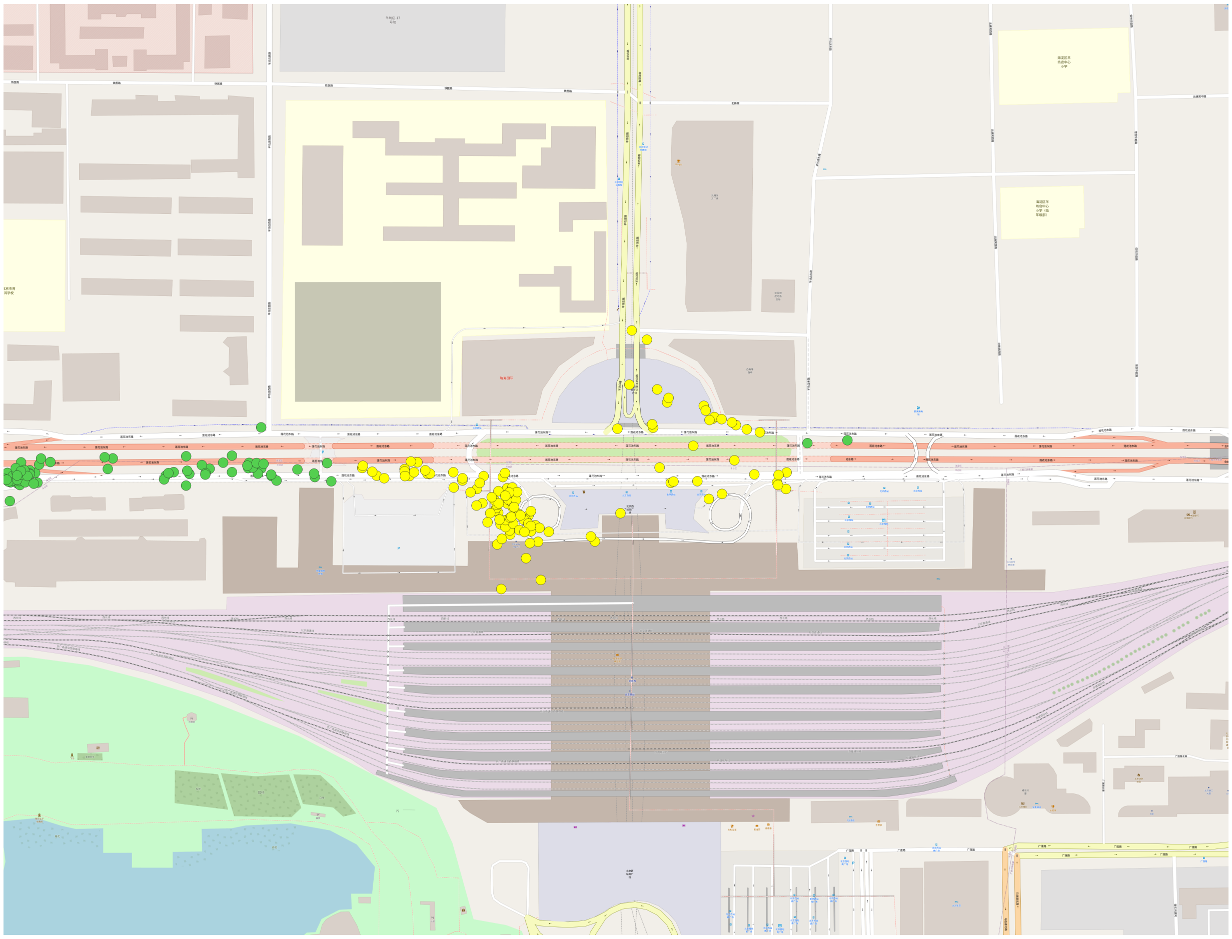


Figure 4.14: The location where cluster evolution paths start near the Beijing West train station in the T-Drive dataset. Locations marked in yellow are considered for analysis.

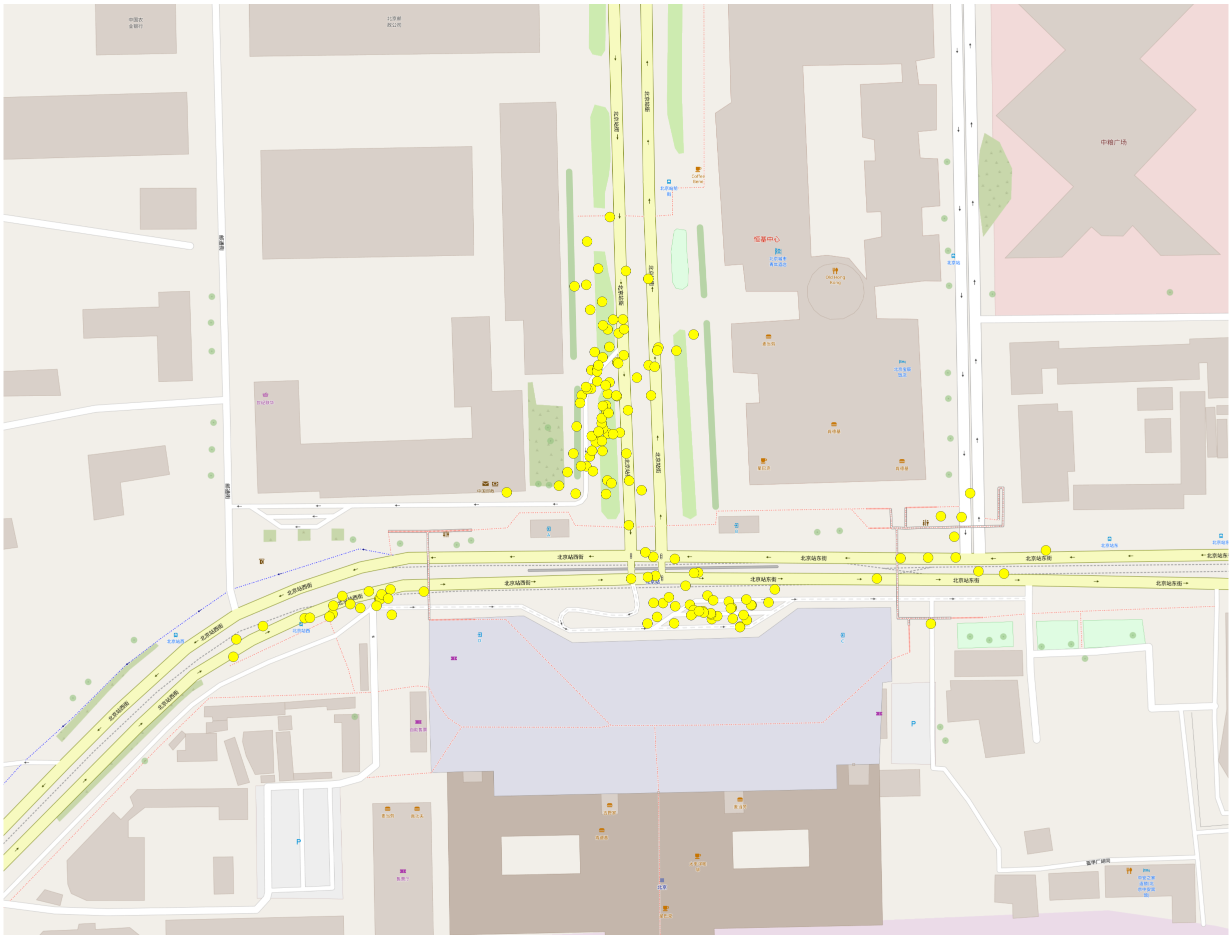


Figure 4.15: The location where cluster evolution paths start near the Beijing East train station in the T-Drive dataset. Locations marked in yellow are considered for analysis.

Table 4.26: The direction of movement represented by cluster evolution paths near the Beijing West and Beijing East train stations in the T-Drive dataset.

Cardinal Direction	Beijing West Train Station	Beijing East Train Station
E	61.54%	4.58%
NE	11.54%	95.13%
N	26.92%	0.29%
NW	0.00%	0.00%
W	0.00%	0.00%
SW	0.00%	0.00%
S	0.00%	0.00%
SE	0.00%	0.00%

direction of movement in an improved way, the intercardinal points can also be included: northwest (NW), northeast (NE), southwest (SW), and southeast (SE). All eight directions are used in this analysis.

Identifying the direction of the movement represented by the cluster evolution paths happens as follows. Since cluster evolution paths are a sequence of occurrences of spatial-temporal clusters at timestamps, the location of the cluster at the first and last timestamps are captured. Since the curvature of Earth does not have significant effects in relatively small distances, the region of the movement can be seen as a plane, such as a map on a table. The two locations then are two points that form a line connecting them. The angle that this line forms with a horizontal line in this map is calculated based on the differences between latitude and longitude coordinates at the end and start points and using either the *arcsine* or *arccosine* functions. This angle is then compared with the angles in the 8-point compass rose, finally defining the direction. The code for this operation is written in Cypher, the graph query language from Neo4j, and is available in Appendix B.4.

Table 4.26 shows the results for this analysis. Note that the table has columns of results for each train station analyzed and values in percentages are relative to the respective train station. In the case of the Beijing West train station, there are 16 (61.54%) movements represented by cluster evolution paths in the East direction, while in the case of Beijing East train station, there are 33,298 (95.13%) movements represented by cluster evolution paths in the northeast direction. Overall, the results tend to show a movement of spatial-temporal clusters towards the Eastern region of Beijing. Improvements to the road network should focus on this region.

What is the distribution of cluster evolution paths by region and by day?

The T-Drive dataset contains spatial-temporal data about several taxis serving passengers in the city of Beijing that was captured in a week, from February 2, 2008 to February 8, 2008. This short amount of time allows for analysis of any distribution by day and a compact view of the results. The distribution by region is the same as observed in the previous analysis, based on cardinal points. In this analysis, results are aggregated and distributed over two dimensions, instead of just one as in the previous result.

As explained in the previous analysis, the identification of the direction of the movements represented by the cluster evolution paths is calculated based on the angle that the line passing through the start and end points of the movement has with a horizontal line in a plan, like a map on a table. Results are then aggregated based on the 8-points compass rose. This time, results are also aggregated by day, using both the spatial and temporal dimensions of the data. The code for this operation is written in Cypher, the graph query language from Neo4j, and is available in Appendix B.4.

Table 4.27 shows the results for this analysis. Note that the table has columns of results for each train station analyzed and values in percentages are relative to the respective train station. The first column of the table has rows for each day of measurements. The other columns are described in pairs, one column for the case in the Beijing West train station and another column for the case in the Beijing East train station. The first pair of columns shows the percentage of cluster evolution paths that started at the date specified by the row. The second pair of columns shows the region where the greatest number of movements represented by cluster evolution paths is directed at the date specified by the row. Finally, the third pair of columns show percentage of movements represented by cluster evolution paths that moved in the direction of the second pair of columns in the date specified by the row.

Overall, the same trend in movement observed in the previous analysis can be observed in the results of this analysis. The second pair of columns of Table 4.27, the fourth and fifth columns, shows a movement towards the east and northeast direction. It is interesting to note a trend in movement towards the north. Results for February 3, for example, show equal movement between the regions of North and Northeast. In addition, note that, according to the first pair of columns, the second and the third columns, February 4 (Monday) and February 5 (Tuesday) are the days when the greatest number of cluster evolution paths are detected. This is independent of the train station considered in the analysis. This indicates that most cluster evolution paths are formed at the beginning of the week and not at the end of it. This result is confirmed by the values at the bottom of the table, showing that no cluster evolution path exists at the end of the week. This

Table 4.27: The distribution of the region and date of the movement represented by cluster evolution paths near the Beijing West and Beijing East train stations in the T-Drive dataset.

	Percentage		Region		Percentage Following Region	
Date	Beijing West Train Station	Beijing East Train Station	Beijing West Train Station	Beijing East Train Station	Beijing West Train Station	Beijing East Train Station
February 2nd, 2008	7.69%	0.05%	NE	NE	100.00%	100.00%
February 3rd, 2008	7.69%	1.09%	N/NE	NE	50.00%	50.13%
February 4th, 2008	23.08%	94.66%	N	NE	100.00%	99.19%
February 5th, 2008	61.54%	4.19%	E	E	100.00%	84.57%
February 6th, 2008	0.00%	0.01%	-	E	-	100.00%
February 7th, 2008	0.00%	0.00%	-	-	-	-
February 8th, 2008	0.00%	0.00%	-	-	-	-

result indicates the region and the day where improvements in the city infrastructure can be directed. In addition, since clusters that move for a long time represent car-sharing or ridesharing opportunities, changes in the price of such services can be considered based on the spatial and temporal dimensions of data.

What time of the day are clusters evolution paths forming?

Identifying and analyzing the time of the day cluster evolution paths start assists in the decision-making process related to improvements in the road network or traffic system controls in a city. For example, during long trips, vehicles tend to enter or leave several clusters of vehicles, which may form cluster evolution paths. If several paths tend to start at a specific time, or time of the day, then additional measures could be taken at the specific time of the day to avoid congestion or other problems.

In this analysis, time is divided into four times of the day, namely, early morning, morning, afternoon, and evening. Early morning comprises the time from 12 AM (midnight) to 6 AM (exclusive). Morning represents the time from 6 AM to 12 PM (noon) (exclusive) and afternoon is the time from 12 PM (noon) to 6 PM (exclusive). Lastly, evening is defined as the time from 6 PM to 12 AM (exclusive). The starting point of every cluster evolution path is queried from the dataset, their times are captured and classified into one

Table 4.28: A distribution of the movement represented by cluster evolution paths near the Beijing West and Beijing East train stations in the T-Drive dataset by the time of the day.

Time of the Day	Beijing West Train Station	Beijing East Train Station
Early Morning	0.00%	2.98%
Morning	69.23%	3.28%
Afternoon	23.08%	1.95%
Evening	7.69%	91.79%

of the four times of the day. The code for this operation is written in Cypher, the graph query language from Neo4j, and is available in Appendix B.4.

Table 4.28 shows the results for this analysis. Note that the table has columns of results for each train station analyzed and values in percentages are relative to the respective train station. The first column of the table contains the four times of the day discussed, followed by two columns with results for each train station in the analysis. Note that, for the case of the Beijing West train station, 18 (69.23%) cluster evolution paths exist in the morning, showing that this station is chosen by passengers arriving in Beijing in the morning. Note also that, for the case of the Beijing East train station, 32,127 (91.79%) cluster evolution paths start in the evening, showing that this station is chosen by passengers returning to Beijing in the evening. It is an interesting phenomenon that indicates that passengers may start the day in one station and end it in another. One possible reason is traffic. These results assist in the allocation of time and resources when investing in improvements in a city.

What is the largest distance traveled in the movements represented by cluster evolution paths formed near each train station?

Cluster evolution paths are not a movement, but rather a sequence of clusters. Since these clusters have a location, it is possible to identify the movement that spatial-temporal objects have within the path. This query investigates cluster evolution paths by calculating the distance of the movement that cluster evolution paths represent and identifying the largest distance for cluster evolution paths starting at each train station. Depending on the domain, such as taxis, long-distance clusters may not be desirable because they may lead to redundant trips, which increase costs of operation, and can be solved with car-sharing or ridesharing opportunities.

Table 4.29: The longest movements represented by cluster evolution paths with respect to distance near the Beijing West and Beijing East train stations in the T-Drive dataset.

Train Station	Cluster Evolution Path	Distance
Beijing West Train Station	$[c_{97057}]$	1598.43 m
Beijing East Train Station	$[c_{61462}, c_{61791}, c_{66349}, c_{67692}, c_{68740}]$	2414.08 m

In this analysis, the movement represented cluster evolution paths starting near each train station are examined for their distance. In other words, the difference in location between the first and last cluster occurrence in all cluster paths is calculated, ranked, and the top results are presented. In Neo4j, the distance between two locations represented in latitude and longitude coordinates are calculated using the Haversine formula [30, 128] over a spherical Earth approximation. The results of the calculation are in meters. The code for this operation is written in Cypher, the graph query language from Neo4j, and is available in Appendix B.4.

Table 4.28 shows the results for this analysis. Note that the table has rows of results for each train station analyzed and distance values are in meters and relative to the respective train station. The first column of the table shows each train station. The second column shows the cluster evolution path in the respect train station whose represented movement is the most distant. Finally, the third column shows the distance value of this most distance movement in meters.

The most distant movement starting from near the Beijing West train station is represented by the cluster evolution path containing only the spatial-temporal cluster c_{97057} and has moved 1,598.43 meters. As it can be seen from the table, the most distant movement starting from near the Beijing East train station is represented by the cluster evolution path containing the spatial temporal clusters c_{61462}, c_{61791}, c_{66349}, c_{67692}, and c_{68740} and has moved 2,414.08 meters. Such long cluster evolution paths indicate that there are opportunities for car-sharing or ridesharing services or improvements in the public bus system in the region.

Chapter 5

Conclusion

This chapter concludes this thesis. The chapter starts with a section containing a summary of the context of this thesis, followed by a description of the problems and the proposed solution. The description of the solution is divided into three perspectives: representation, analysis, and implementation. Then, the four case studies are summarized. Later, the chapter discusses the limitations of the proposed approach. The chapter ends presenting some opportunities for future work.

5.1 Conclusion

Spatial-temporal objects are real-world entities that move, such as vehicles on a road. Their movement can be described by their location, thus the spatial dimension, and the time, thus the temporal dimension. During their movement, they may group and stay as a group for some time. A spatial-temporal cluster is a group of spatial-temporal objects that stay together for some time. During this time, if a spatial-temporal cluster approaches another cluster, it may enter the second cluster, or they can merge and form a larger cluster. Similarly, a larger cluster may split into two or more spatial-temporal clusters. These interactions between clusters happen based on the movement of spatial-temporal objects a cluster contains. Spatial-temporal relationships are the interactions that spatial-temporal clusters have with other clusters or spatial-temporal objects. The structure of spatial-temporal objects and the occurrence of spatial-temporal relationships happen throughout time. Spatial-temporal cluster evolution refers to the development of all clusters and their interactions in space and time. Spatial-temporal relationships bind spatial-temporal clusters such that the evolution of a cluster, from its start, going through several relationships, to its end, can be visualized in a connected representation using a graph. Graph-based cluster evolution refers to the representation of the structure and relationships of all spatial-temporal clusters in a graph format.

However, there is a lack of studies in three major areas: the representation, the analysis, and the implementation of graph-based cluster evolution.

There is a lack of studies in the representation of the *structure* of spatial-temporal clusters. These clusters exist throughout time and several changes happen to their shape or number of spatial-temporal objects they contain. There is also a lack of studies in the representation of the *relationships* these clusters have with other clusters or spatial-temporal objects. Identifying these relationships becomes challenging when clusters are very close to each other. Moreover, there is a lack of studies in the representation of spatial-temporal cluster evolution in a connected structure, as in a *graph.* Representing the structure and the relationships of spatial-temporal clusters in such a connected structure requires a conversion of representations, from spatial-temporal to a graph notation.

There is also a lack of studies in the analysis methods for the *structure* of spatial-temporal clusters. Analysis on properties of clusters such as their location or size are not available. There is a lack of studies in the analysis methods for the *relationships* of spatial-temporal clusters. Analysis on the frequency, or density of these relationships during the existence of a spatial-temporal cluster are not performed and important conclusions are not drawn. Moreover, there is a lack of studies in the analysis methods for the *graph*-based representation of spatial-temporal cluster evolution. Analysis of the location, time, or movement of spatial-temporal clusters, including aggregated results among all clusters in a dataset, are not performed and significant value is missed.

Moreover, there is a lack of studies in the implementation tools, or software support, for the *representation* of graph-based cluster evolution. The tools and libraries available for clustering tasks do not take into consideration the nature of spatial-temporal clusters, which move in time, or their relationships. There is a lack of studies in the implementation tools, or software support, for the *analysis* of graph-based cluster evolution do not exist, as most tools are available for just general analysis tasks.

The proposed solution in this thesis is an approach for graph-based cluster evolution. The approach contributes in three main areas: the representation, the analysis, and the implementation of graph-based cluster evolution.

The solution makes contributions in the representation of the *structure* of spatial-temporal clusters. The identification and representation of spatial-temporal clusters follow a two-step process. First, as spatial-temporal clusters exist at several timestamps, the approach identifies the slice of a cluster at each timestamp. Second, after each slice of a spatial-temporal cluster has been identified, the approach then links them. The approach uses DBSCAN, a clustering technique, on each timestamp to identify slices of spatial-temporal clusters and links each slice using a similarity value that is an extension of Jaccard similarity. The approach makes contributions in the representation of the *relationships* spatial-temporal clusters have with other clusters or spatial-temporal objects. The approach identifies 14 spatial-temporal relationships that describe the spatial-temporal lifetime of a spatial-temporal cluster. The approach makes contributions in the construction of a *graph*-based representation for spatial-temporal cluster evolution. Once the structure and the relationships of spatial-temporal clusters are identified, the approach has the information to represent cluster lifetimes describing information about the evolution of a cluster, such as its location, relationships, and size. The lifetime is then converted to a graph format where graph vertices, or graph nodes, are occurrences of this cluster in each timestamp and graph edges are spatial-temporal relationships.

The solution makes contributions in the analysis of the *structure* of spatial-temporal clusters. Several analysis methods can be performed on the location or size of spatial-temporal clusters to identify size changes, most frequent locations, or the location of interesting phenomena, such as opportunities for car sharing and city improvements. The solution makes contributions in the analysis of the *relationships* spatial-temporal clusters have with other clusters or spatial-temporal objects. Several analysis methods can be performed on the frequency of these relationships or the density at some interval of time, facilitating discoveries such as clusters that grow or shrink rapidly. The approach makes contributions in the analysis of a *graph*-based representation of spatial-temporal cluster evolution. Several analysis methods can be performed on the movement or evolution of spatial-temporal clusters, leading to the representation of spatial-temporal cluster paths and several analysis results in an aggregated format, such as the longest cluster path or a similarity between cluster paths.

The solution makes contributions in the implementation of the *representation* of graph-based cluster evolution. The identification of spatial-temporal clusters uses clustering techniques present in the scikit-learn library for Python. The linking of slices of spatial-temporal clusters per timestamp is done in Python, as well as the identification of spatial-temporal cluster relationships. To have a graph-based representation of spatial-temporal cluster evolution, the approach converts spatial-temporal cluster lifetimes to a graph format, using Python, to generate a list of graph nodes and edges. These nodes and edges are then imported into Neo4j, a graph data platform. The solution has contributions in the implementation of the *analysis* of graph-based cluster evolution. Analysis on the structure and relationships of spatial-temporal clusters or in a connected, graph-based representation of cluster evolution happens mostly in Neo4j, with several queries to the data imported.

These queries are written in Cypher, a graph query language, and are saved for reuse.

Four case studies show applications of the approach and its contributions to real spatial-temporal data in popular datasets. The datasets are Athens Trucks, containing the trajectory of cement trucks in Athens, Rome Taxis, containing the trajectory of taxis in Rome, Geolife, containing the spatial-temporal data about the movement of people and cars in everyday activities mostly in Beijing, and T-Drive, containing the trajectory of taxis in Beijing.

The first case study is exploratory and shows some queries that can be executed on the graph containing the graph-based cluster evolution. It describes 23 queries divided among queries about the representation, analysis, and implementation of graph-based cluster evolution. Each query is executed, explained, and their results are discussed, such as the identification of spatial-temporal clusters that start at a location, start by a merge of other clusters, or that produce cluster evolution paths containing a certain number of spatial-temporal cluster relationships. Results help identify interesting phenomena in the dataset, such as the time when the demand for taxis is highest or a common destination during Valentine's day, or outliers, that can be further investigated for novel results.

The second case study investigates moments in the existence of a spatial-temporal cluster during which the number of spatial-temporal objects contained only increase or decrease. These moments are called ever-increasing or ever-decreasing regions. The average rate of change (AROC) of cluster size is discussed and calculated. Two cluster evolution paths, one with a long ever-increasing region and another with a short one, are discussed, their growth rates are calculated, and some conclusions about the offer and demand of spatial-temporal services are presented.

The third case study introduces the similarity between cluster evolution paths in graph-based cluster evolution. The case study discusses the process to calculate such similarity value and shows how it can be calculated. It then proceeds to divide time into 24 slots, corresponding to hours of the day, and calculates the similarity of cluster evolution paths starting in each slot. Comparisons between the similarity value and the number of cluster paths at each hour of the day are made, and a correlation is discussed. It ends with a discussion on the importance of a similarity value for ML tasks such as classification and clustering.

The fourth case study analyzes important aspects of the movement represented by cluster evolution paths, such as its distance and direction, and provides aggregated results on the distribution of cluster evolution paths by hour, day, and region. The case study contains a list of four analysis questions related to the analysis aspects just presented that guides the discussion of the results. These results can be used for identifying phenomena such as car-sharing opportunities and improvements in traffic planning, the traffic system, and the road network of cities.

5.2 Limitations

The proposed approach is a solution for the representation, analysis, and implementation of graph-based spatial-temporal cluster analysis. Although the approach can manipulate spatial-temporal data, transform it, represent it in a graph, analyze it and extract its value, this solution still has some limitations. This section discusses these limitations and divides them into three groups: representation, analysis, and implementation.

In the representation of graph-based spatial-temporal cluster evolution, spatial-temporal data is described by latitude and longitude coordinates. This is because most spatial-temporal objects are vehicles or people. However, some studies consider the movement of airplanes, in which the altitude becomes a third spatial dimension [104]. The current version of the approach does not consider a third spatial dimension, such as altitude, and does not consider other theoretical spatial dimensions. Therefore, one limitation of the proposed approach is the description of data in two dimensions.

The proposed approach identifies 14 spatial-temporal relationships that describe the interactions a cluster has during its existence. Although effort is put to describe several interactions that cluster may have, it is understood that these 14 relationships are a starting point in the identification of spatial-temporal cluster relationships. Other relationships can be discovered or extended from this initial set. One such relationship is the GO_THROUGH relationship [69]. The current version of the approach does not consider extended relationships. Therefore, one limitation of the proposed approach is the description of only 14 primitive spatial-temporal cluster relationships.

Spatial-temporal clusters can assume any shape and are not limited to a circular one. This is the main reason for the choice of a density-based clustering technique in the identification of the clusters. However, in some situations, spatial-temporal objects inside a cluster can be represented as other clusters. For example, a parking lot can be divided into VIP parking and regular parking. Another example is a group of tourists in a city that can be divided in smaller groups based on the travel agency. In both examples, spatial-temporal objects can be represented as smaller clusters within the larger cluster. This raises the concept of hierarchical clustering [110]. The current version of the approach does not capture hierarchical spatial-temporal clusters. Therefore, one limitation of the proposed approach is the description of cluster in a “flat”, not a nested, representation.

In the analysis of graph-based spatial-temporal cluster evolution, spatial-temporal data is obtained from a single repository. The reason for this is partly because spatial-temporal datasets are usually centralized and partly because the "Preprocess" step of the approach adopts the strategy to centralize data in one repository. However, because the amount of data that can be manipulated can be very high, it is best to leave spatial-temporal data in different datasets and query them at analysis time. The current version of the approach does not support the representation and analysis of spatial-temporal data in several repositories. Therefore, one limitation of the proposed approach is the description of spatial-temporal data in a single repository.

The analysis of graph-based cluster evolution happens in Neo4j because of the use of Cypher, a graph query language, that becomes the standard for graph analysis. Cypher is a declarative language similar to Structured Query Language (SQL) for relational databases. Because Cypher is optimized for query and retrieval of graph data, new libraries simplify other manipulations of the data. One such library is Awesome Procedures on Cypher (APOC)[1][2]. However, some queries would be more understandable and intuitive if a procedural language was used. For example, checking properties of objects on a list or subqueries that return data could be simpler in Python. For this reason, sometimes, the result of a query is further processed in Python or on the rows and columns of a spreadsheet application. The current version of the approach does offer support for a procedural query and retrieval of spatial-temporal data from the graph format. Therefore, one limitation of the proposed approach is a declarative, SQL-like, syntax to query and retrieve data.

In the implementation of graph-based spatial-temporal cluster evolution, spatial-temporal data is transformed from a tabular format into a graph format and imported in a graph data platform called Neo4j. The choice of Neo4j is based on its graph query language Cypher. However, other graph data platforms are available to process the graph format of cluster evolution. Some adaptations are required, especially in the conversion between representations. The current version of the approach does not integrate with other graph data platforms. Therefore, one limitation of the proposed approach is the integration with only Neo4j for graph data processing.

When analyzing spatial-temporal data in Neo4j, data is queried from a centralized repository and processed in a single processing unit, called a computation node or simply a node. This is not a graph node, but a computer with processing capabilities. The research area of distributed systems [67] studies the systems that use computation capabilities of several machines, or computation nodes, so that calculations can be performed in parallel,

[1] https://neo4j.com/apoc/

[2] https://github.com/neo4j/apoc

delivering results more efficiently. The current version of the approach does not provide support for distributed analysis of spatial-temporal data. Therefore, one limitation of the proposed approach is the analysis of graph-based cluster evolution in a centralized way.

5.3 Future Work

Some opportunities for future work relate to the use of contextual information in the analysis of graph-based cluster evolution, the introduction of real-time analysis to graph-based cluster evolution, the use of Artificial Intelligence (AI) or ML methods for prediction tasks in graph-based cluster evolution, and the use of more advanced graph analysis methods, such as graph comparison, in graph-based cluster evolution. Each opportunity is further discussed.

Spatial-temporal clusters are identified by performing a clustering technique on temporal-spatial data present at every timestamp and linking the resulting clusters based on their similarity. This process is completely based on the spatial-temporal data that describe spatial-temporal objects, namely their location, the time, and an identification. However, other types of information about spatial-temporal objects are not used, such as the velocity, the number of passengers, color, weather, lane, or purpose of the trip. These types of information relate to the environment in which the spatial-temporal object resides and are usually referred to as contextual information [121, 119]. The use of contextual information can enrich the data about that cluster, assist in the calculation of similarity between timestamps, help identify novel spatial-temporal relationships, and yield a more accurate representation of spatial-temporal clusters in graph-based cluster evolution.

Analysis on graph-based cluster evolution in the proposed approach happens after the spatial-temporal data has been measured and made available in a dataset. Although important phenomena are captured and identified, the decision-making process is impacted because no immediate action can be taken in a city to improve conditions if necessary. For example, if a major accident has been identified in a road, traffic cannot be redirected to other roads, or if it is identified that the demand for spatial-temporal services is high in a region, no spatial-temporal objects can be directed to that region in a timely manner. Real-time analytics [129] is defined as the process of using analytics tools on data to provide better results for decision-making quickly. Analysis in real-time happens in several domains, including the spatial-temporal and the graph domain. For example, the study in [63] discusses the software support and the analysis methods for real-time analysis of congestion in a city. The studies on [72, 71] describe Graphone, which performs real-time analysis on evolving graphs. Real-time analysis of graph-based cluster evolution delivers quick results for immediate decision-making, assisting in the handling of changes to the traffic flow or in offer and demand values, which result in better spatial-temporal services.

The current analysis methods discussed when proposing this approach relate to identifying changes to the structure of spatial-temporal clusters, the occurrences of spatial-temporal cluster relationships, ranking clusters based on their spatial or temporal features, and calculating a similarity for classification or clustering tasks. However, these analysis methods have results that are within the spatial and temporal domain in which the data was measured, although the impacts of such results can go beyond this domain. For example, classifying spatial-temporal cluster paths based on the number of relationships it contains produces valuable results that are within the measurements taken. A prediction is a statement, made in advance, about an event or situation. Several AI and ML methods are dedicated to predicting events [8, 42], and some of them are focused on the spatial-temporal domain [108]. For example, the study in [143] uses neural networks to predict traffic conditions. Analysis methods for predictions produce results that are beyond the spatial and temporal domains of the data because they refer to situations in a different location or in the future. In graph-based cluster evolution, ML methods for prediction can have results in the prediction of the location of spatial-temporal clusters, the relationships they might have, the flow of spatial-temporal objects in a city, and the traffic conditions based on the number of spatial-temporal objects, the start or end of events, or the weather.

The connected representation of cluster evolution is based on a graph. Spatial-temporal clusters are represented as several vertices, or nodes, of the graph and their relationships are represented by the edges. Relationships that involve more than one spatial-temporal cluster bind these clusters in the graph representation, creating a connected, graph-based cluster evolution. Because of this connected representation, many graph analysis methods can be executed, which produce novel results and uncover value in the spatial-temporal domain. For example, a long cluster evolution path may indicate opportunities for car-sharing. However, several other analysis methods are still not used because of the adaptations required to perform in a graph-based cluster evolution environment. Graph analysis is an area of research focused on the use of analysis tools for graphs [87]. Usually, graph analysis methods are used for social network analysis, fraud detection, or recommendation engines. Some graph analysis methods are partitioning, which divides a graph in many parts to find weak points, shortest path, which find the shortest path between two vertices, or nodes, and can be used in optimizations in transportation logistics, and page rank, which measures the popularity of websites and ranks them for efficient results in online searches [62]. Graph analysis methods can be used in graph-based cluster evolution to identify, for example, points of interest based on the relationship between the movement of clusters and locations. One interesting approach is graph comparison, in which the traffic flow of different cities can be compared to produce novel results and conclusions.

References

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings - 29th International Conference on Very Large Data Bases, VLDB 2003*, pages 81–92. Morgan Kaufmann, 2003.

[2] Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5):1110–1121, 2012. DOI: 10.1137/110848712.

[3] Mohammed Eunus Ali, Shadman Saqib Eusuf, Kaysar Abdullah, Farhana M. Choudhury, J. Shane Culpepper, and Timos Sellis. The maximum trajectory coverage query in spatial databases. In volume 12 of number 3, pages 197–209. Association for Computing Machinery, 2018. DOI: 10.14778/3291264.3291266.

[4] Federico Amato, Luigi Lombardo, Marj Tonini, Antonino Marvuglia, Daniela Castro-Camilo, and Fabian Guignard. Spatiotemporal data science: theoretical advances and applications. *Stochastic Environmental Research and Risk Assessment*, 36(8):2027–2029, 2022. DOI: 10.1007/s00477-022-02281-4.

[5] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60. Association for Computing Machinery, 1999. DOI: 10.1145/304182.304187.

[6] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2):49–60, 1999. DOI: 10.1145/304181.304187.

[7] Mohd Yousuf Ansari, Amir Ahmad, Shehroz S. Khan, Gopal Bhushan, and Mainuddin. Spatiotemporal clustering: a review. *Artificial Intelligence Review*, 53(4):2381–2423, 2020. DOI: 10.1007/s10462-019-09736-1.

[8] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Elsevier Inc., 2014. DOI: 10.1016/C2012-0-00234-7.

[9] Richard Bellmain and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1958. DOI: 10.1109/TAC.1959.1104847.

[10] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry: Theory and Applications*, 41(3):111–125, 2008. DOI: 10.1016/j.comgeo.2007.10.003.

[11] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, 1981, pages 1–272.

[12] Panthadeep Bhattacharjee and Pinaki Mitra. A survey of density based clustering algorithms. *Frontiers of Computer Science*, 15(1), 2021. DOI: 10.1007/s11704-019-9059-3.

[13] Sonja Biedermann. *Evolutionary Graph Clustering*. Bachelor's thesis, Universität Wien, 2017.

[14] Derya Birant and Alp Kut. St-dbscan: an algorithm for clustering spatial-temporal data. *Data and Knowledge Engineering*, 60(1):208–221, 2007. DOI: 10.1016/j.datak.2006.01.013.

[15] Veselka Boeva, Milena Angelova, and Elena Tsiporkova. A split-merge evolutionary clustering algorithm. In volume 2, pages 337–346. SciTePress, 2019. DOI: 10.5220/0007573103370346.

[16] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. The rome taxis dataset, 2014. DOI: 10.15783/C7QC7M.

[17] Ludwig M. Busse, Peter Orbanz, and Joachim M. Buhmann. Cluster analysis of heterogeneous rank data. In volume 227, pages 113–120, 2007. DOI: 10.1145/1273496.1273511.

[18] Xu Chen, Weijun Li, and Li Yan. A uml-based representation of fuzzy spatiotemporal relations. In *ICNC-FSKD 2017 - 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 1090–1098. Institute of Electrical and Electronics Engineers Inc., 2018. DOI: 10.1109/FSKD.2017.8392915.

[19] Miyoung Cho, Chang Choi, Junho Choi, Hongryoul Yi, and Pankoo Kim. Trajectory annotation and retrieval based on semantics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4918 LNCS:251–264, 2008. DOI: 10.1007/978-3-540-79860-6_20.

[20] Miyoung Cho, Dan Song, Chang Choi, Junho Choi, Jongan Park, and Pankoo Kim. Comparison between motion verbs using similarity measure for the semantic representation of moving object. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4071 LNCS:281–290, 2006. DOI: 10.1007/11788034_29.

[21] Miyoung Cho, Dan Song, Chang Choi, and Pankoo Kim. Measuring similarity in the semantic representation of moving objects in video. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4092 LNAI:78–87, 2006. DOI: 10.1007/11811220_8.

[22] Chang Choi, Miyoung Cho, and Pankoo Kim. The new modeling for semantic representation of moving objects in video. In *International Conference on Advanced Communication Technology, ICACT*, volume 1, pages 363–367, 2007. DOI: 10.1109/ICACT.2007.358373.

[23] William G. Cochran. The χ^2 Test of Goodness of Fit. *The Annals of Mathematical Statistics*, 23(3):315–345, 1952. DOI: 10.1214/aoms/1177729380.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. DOI: 10.1023/A:1022627411411.

[25] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. DOI: 10.1109/TIT.1967.1053964.

[26] Jesus Cuenca-Jara, Fernando Terroso-Saenz, Ramon Sanchez-Iborra, and Antonio F. Skarmeta-Gomez. Classification of spatio-temporal trajectories based on support vector machines. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10978 LNAI:140–151, 2018. DOI: 10.1007/978-3-319-94580-4_11.

[27] Ticiana L. Coelho Da Silva, José A. F. De Macêdo, and Marco A. Casanova. Discovering frequent mobility patterns on moving object data. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2014 - In Conjunction with the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM*

SIGSPATIAL 2014, pages 60–67. Association for Computing Machinery, 2014. DOI: 10.1145/2675316.2675325.

[28] Abraham De Moivre. Approximatio ad summam terminorum binomii $\overline{a+b}\backslash^n$ in seriem expansi. 1733.

[29] Abraham De Moivre. *The Doctrine of Chances: A Method of Calculating the Probabilities of Events in Play*. H. Woodfall, 2nd edition, 1738.

[30] Don Joseph de Mendoza y Rios. *Memoria sobre algunos métodos nuevos de calcular la longitud por las distancias lunares: Y applicacion de su teórica á la solucion de otros problemas de navegacion*. Imprenta Real, 1795.

[31] Reinhard Diestel. *Graph Theory*. Springer Berlin Heidelberg, 2017. DOI: 10.1007/978-3-662-53622-3.

[32] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008. DOI: 10.14778/1454159.1454226.

[33] Christos Doulkeridis, Akrivi Vlachou, Nikos Pelekis, and Yannis Theodoridis. A survey on big data processing frameworks for mobility analytics. *SIGMOD Record*, 50(2):18–29, 2021. DOI: 10.1145/3484622.3484626.

[34] Moez Draief and Ayalvadi Ganesh. A random walk model for infection on graphs: spread of epidemics & rumours with mobile agents. *Discrete Event Dynamic Systems: Theory and Applications*, 21(1):41–61, 2011. DOI: 10.1007/s10626-010-0092-5.

[35] Gérard Dreyfus. *Neural Networks: Methodology and Applications*. Springer Berlin Heidelberg, 2005. DOI: 10.1007/3-540-28847-3.

[36] Joseph C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973. DOI: 10.1080/01969727308546046.

[37] Ramakrishan Reddy Eamani, N. Vinodh Kumar, and Ganga Ramesh Jakkamsetti. K-means clustering algorithm and architecture: a brief survey. *International Journal of Advanced Science and Technology*, 29(6):2955–2967, 2020.

[38] Bradley Efron. *Large-Scale Inference Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, 2012. DOI: 10.1017/CBO9780511761362.

[39] Gidon Eshel. *Spatiotemporal Data Analysis*. Princeton University Press, 2011.

[40] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press, 1996.

[41] Shimon Even and Guy Even. *Graph Algorithms, 2nd Edition*. Cambridge University Press, 2011, pages 1–189. DOI: 10.1017/CBO9781139015165.

[42] Matteo Fontana, Gianluca Zeni, and Simone Vantini. Conformal prediction: a unified review of theory and new challenges. *Bernoulli*, 29(1):1–23, 2023. DOI: 10.3150/21-BEJ1447.

[43] Edward Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21(3):768–769, 1965.

[44] Ove Frank. A survey of statistical methods for graph analysis. *Sociological Methodology*, 12:110–155, 1981.

[45] David A. Freedman. *Statistical Models: Theory and Practice*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511815867.

[46] Carl Friedrich Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. F. Perthes & I. H. Besser, 1809.

[47] Sarah Gerster, Ermir Qeli, Christian H. Ahrens, and Peter Bühlmann. Protein and gene model inference based on statistical modeling in k-partite graphs. *Proceedings of the National Academy of Sciences of the United States of America*, 107(27):12101–12106, 2010. DOI: 10.1073/pnas.0907654107.

[48] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. State-of-the-art on clustering data streams. *Big Data Analytics*, 1(13):1–27, 2016. DOI: 10.1186/s41044-016-0011-3.

[49] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 330–339, 2007. DOI: 10.1145/1281192.1281230.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[51] Derek Greene and Pádraig Cunningham. Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th Annual ACM Web Science Conference, WebSci'13*, pages 118–121. Association for Computing Machinery, 2013. DOI: 10.1145/2464464.2464471.

[52] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003. DOI: 10.1109/TKDE.2003.1198387.

[53] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 359–366. IEEE, 2000.

[54] Jiawei Ha, Micheline Kambe, and Jian Pe. *Data Mining: Concepts and Techniques.* Elsevier, 2011. DOI: 10.1016/C2009-0-61819-5.

[55] Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci. *Handbook of cluster analysis.* CRC Press, 2015, pages 1–731. DOI: 10.1201/b19706.

[56] Clemens Holzmann. Rule-based reasoning about qualitative spatiotemporal relations. In *Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-hoc Computing, MPAC 2007 held at the ACM/IFIP/USENIX 8th International Middleware Conference*, pages 49–54, 2007. DOI: 10.1145/1376866.1376875.

[57] Chowdhury Md. Intisar and Yutaka Watanobe. Cluster analysis to estimate the difficulty of programming problems. In pages 23–28. Association for Computing Machinery, 2018. DOI: 10.1145/3274856.3274862.

[58] Paul Jaccard. Distribution de la flore alpine dans le bassin des dranes et dans quelques regions voisines. *Bull. Soc. Vaud. Sci. Nat.*, 37(140):241–272, 1901.

[59] Anil Kumar Jain, Musti Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. DOI: 10.1145/331499.331504.

[60] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008. DOI: 10.14778/1453856.1453971.

[61] Ioannis Kakoudakis and Babis Theodoulidis. The TAU Time Model. Technical report, Timelab, UMIST, 1996.

[62] Ananth Kalyanaraman and Partha Pratim Pande. A brief survey of algorithms, architectures, and challenges toward extreme-scale graph analytics. In *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, pages 1307–1312. Institute of Electrical and Electronics Engineers Inc., 2019. DOI: 10.23919/DATE.2019.8715024.

[63] Lamia Karim, Azedine Boulmakoul, and Ahmed Lbath. Real time analytics of urban congestion trajectories on hadoop-mongodb cloud ecosystem. In *ACM International Conference Proceeding Series*. Association for Computing Machinery, 2017. DOI: 10.1145/3018896.3018923.

[64] Leonard Kaufman and Peter J. Rousseeuw. *Partitioning around medoids (program pam)*. In *Finding Groups in Data*. John Wiley & Sons, Ltd, 1990. Chapter 2, pages 68–125. DOI: 10.1002/9780470316801.ch2.

[65] Ross Kindermann and James Laurie Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980. DOI: https://doi.org/10.1090/conm/001.

[66] Slava Kisilevich, Florian Mansmann, Mirco Nanni, and Rinzivillo Salvatore. Spatio-Temporal Clustering: a Survey. Technical report, Italian National Research Council, 2015.

[67] Martin Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly, 2017.

[68] Sotiris B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013. DOI: 10.1007/s10462-011-9272-4.

[69] Christian Kray and Anselm Blocher. Modeling the basic meanings of path relations. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 1, pages 384–389, 1999.

[70] Dirk P. Kroese, Thomas Taimre, and Zdravko I. Botev. *Handbook of Monte Carlo Methods*. Wiley Blackwell, 2011. DOI: 10.1002/9781118014967.

[71] Pradeep Kumar and H. Howie Huang. Graphone: a data store for real-time analytics on evolving graphs. In *Proceedings of the 17th USENIX Conference on File and Storage Technologies, FAST 2019*, pages 249–263. USENIX Association, 2019.

[72] Pradeep Kumar and H. Howie Huang. Graphone: a data store for real-time analytics on evolving graphs. *ACM Transactions on Storage*, 15(4), 2020. DOI: 10.1145/3364180.

[73] Seoyun J. Kwon. *Artificial Neural Networks*. Nova Science Publishers, Inc., 2011.

[74] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. Dynamic log file analysis: an unsupervised cluster evolution approach for anomaly detection. *Computers and Security*, 79:94–116, 2018. DOI: 10.1016/j.cose.2018.08.009.

[75] Hang T. Lau. *A java library of graph algorithms and optimization*. CRC Press, 2006, pages 1–386. DOI: 10.1201/9781584887195.

[76] Patrick Laube, Stephan Imfeld, and Robert Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005. DOI: 10.1080/13658810500105572.

[77] Patrick Laube, Marc van Kreveld, and Stephan Imfeld. Finding remo — detecting relative motion patterns in geospatial lifelines. In *Developments in Spatial Data Handling*, pages 201–215. Springer Berlin Heidelberg, 2005. DOI: 10.1007/3-540-26772-7_16.

[78] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007. DOI: 10.1145/1247480.1247546.

[79] Bonan Li and Frederico Fonseca. Tdd: a comprehensive model for qualitative spatial similarity assessment. *Spatial Cognition and Computation*, 6(1):31–62, 2006. DOI: 10.1207/s15427633scc0601_2.

[80] Xiaohui Li, Vaida Čeikute, Christian S. Jensen, and Kian-Lee Tan. Effective online group discovery in trajectory databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2752–2766, 2013. DOI: 10.1109/TKDE.2012.193.

[81] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1):723–734, 2010. DOI: 10.14778/1920841.1920934.

[82] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[83] Woong-Kee Loh and Young-Ho Park. A survey on density-based clustering algorithms. *Lecture Notes in Electrical Engineering*, 280 LNEE:775–780, 2014. DOI: 10.1007/978-3-642-41671-2_98.

[84] Zongmin Ma, Luyi Bai, and Li Yan. Spatiotemporal data and spatiotemporal data models. *Studies in Computational Intelligence*, 894:1–18, 2020. DOI: 10.1007/978-3-030-41999-8_1.

[85] Nehal Magdy, Mahmoud A. Sakr, Tamer Mostafa, and Khaled El-Bahnasy. Review on trajectory similarity measures. In *2015 IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS 2015*, pages 613–619. Institute of Electrical and Electronics Engineers Inc., 2016. DOI: 10.1109/IntelCIS.2015.7397286.

[86] Johannes S. Maritz and Thaung Lwin. *Empirical Bayes Methods, Second Edition.* CRC Press, 2018. DOI: 10.1201/9781351071666.

[87] Claudio Martella, Roman Shaposhnik, and Dionysios Logothetis. *Practical Graph Analytics with Apache Giraph.* Apress Media LLC, 2015. DOI: 10.1007/978-1-4842-1251-6.

[88] Jean Damascène Mazimpaka and Sabine Timpf. Trajectory data mining: a review of methods and applications. *Journal of Spatial Information Science*, 13(2016):61–99, 2016. DOI: 10.5311/josis.2016.13.263.

[89] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions: Second Edition.* Wiley Blackwell, 2007, pages 1–369. DOI: 10.1002/9780470191613.

[90] O. Mekenyan, D. Bonchev, and N. Trinajstić. Chemical graph theory: modeling the thermodynamic properties of molecules. *International Journal of Quantum Chemistry*, 18(2):369–380, 1980. DOI: 10.1002/qua.560180206.

[91] Xiao-Li Meng and David Van Dyk. The EM Algorithm—an Old Folk-song Sung to a Fast New Tune. *Journal of the Royal Statistical Society: Series B (Methodological)*, 59(3):511–567, 1997. DOI: 10.1111/1467-9868.00082.

[92] Michael Mills, Adamantia Psarologou, and Nikolaos Bourbakis. Modeling natural language sentences into spn graphs. In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, pages 889–896, 2013. DOI: 10.1109/ICTAI.2013.135.

[93] Cho Miyoung, Choi Chang, and Kim Pankoo. Measuring similarity between trajectories using motion verbs in semantic level. In *International Conference on Advanced Communication Technology, ICACT*, volume 1, pages 511–515, 2007. DOI: 10.1109/ICACT.2007.358406.

[94] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26(4):354–359, 1983. DOI: 10.1093/comjnl/26.4.354.

[95] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. DOI: 10.1002/widm.53.

[96] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006. DOI: 10.1007/s10844-006-9953-7.

[97] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001. DOI: 10.1145/375360.375365.

[98] Mark E.J. Newman, Duncan J. Watts, and Steven H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(SUPPL. 1):2566–2572, 2002. DOI: 10.1073/pnas.012582999.

[99] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamouliso, Lei Chen, and HansPeter Kriegel. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *Proceedings of the VLDB Endowment*, 7(3):205–216, 2013. DOI: 10.14778/2732232.2732239.

[100] Christian Nordahl, Veselka Boeva, Håkan Grahn, and Marie Persson Netz. Evolvecluster: an evolutionary clustering algorithm for streaming data. *Evolving Systems*, 13(4):603–623, 2022. DOI: 10.1007/s12530-021-09408-y.

[101] Ratchata Peachavanish. Stock selection and trading based on cluster analysis of trend and momentum indicators. In volume 1, pages 317–326. Newswood Limited, 2016.

[102] Karl 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. *Philosophical Magazine*, 50(5):157–175, 1900.

[103] Nikos Pelekis, Babis Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. Literature review of spatio-temporal database models. *Knowledge Engineering Review*, 19(3):235–274, 2004. DOI: 10.1017/S026988890400013X.

[104] Rémi Perrichon, Xavier Gendre, and Thierry Klein. A geometric approach to study aircraft trajectories: the benefits of opensky network ads-b data. *Engineering Proceedings*, 28(1), 2022. DOI: 10.3390/engproc2022028006.

[105] Alain Polguère. Lexical systems: graph models of natural language lexicons. *Language Resources and Evaluation*, 43(1):41–55, 2009. DOI: 10.1007/s10579-008-9078-4.

[106] Ivens Portugal, Paulo Alencar, and Donald Cowan. Spatial-temporal cluster relations: a foundation for trajectory cluster lifetime analysis, 2019. DOI: 10.48550/arXiv.1911.02105.

[107] Satrio Adi Priyambada, Mahendrawathi Er, Bernardo Nugroho Yahya, and Tsuyoshi Usagawa. Profile-based cluster evolution analysis: identification of migration patterns for understanding student learning behavior. *IEEE Access*, 9:101718–101728, 2021. DOI: 10.1109/ACCESS.2021.3095958.

[108] Milan P. Ptotić, Miloš B. Stojanović, and Predrag M. Popović. A review of machine learning methods for long-term time series prediction. In *2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST 2022*. Institute of Electrical and Electronics Engineers Inc., 2022. DOI: 10.1109/ICEST55168.2022.9828618.

[109] Roni Ramon-Gonen and Roy Gelbard. Cluster evolution analysis: identification and detection of similar clusters and migration patterns. *Expert Systems with Applications*, 83:363–378, 2017. DOI: 10.1016/j.eswa.2017.04.007.

[110] Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 2022. DOI: 10.1007/s10462-022-10366-3.

[111] Stan Salvador and Philip Chan. Fastdtw: toward accurate dynamic time warping in linear time and space. In *KDD Workshop on Mining Temporal and Sequential Data*, pages 70–80. Association for Computing Machinery, 2004.

[112] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007. DOI: 10.3233/ida-2007-11508.

[113] Joerg Sander. *Density-based clustering*. In *Encyclopedia of Machine Learning*. Springer, 2010, pages 270–273. DOI: 10.1007/978-0-387-30164-8_211.

[114] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. DOI: 10.1016/j.cosrev.2007.05.001.

[115] Jens Schrodt, Aleksei Dudchenko, Petra Knaup-Gregori, and Matthias Ganzinger. Graph-representation of patient data: a systematic literature review. *Journal of Medical Systems*, 44(4), 2020. DOI: 10.1007/s10916-020-1538-4.

[116] Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems*, 42(3), 2017. DOI: 10.1145/3068335.

[117] Pavel Senin. Dynamic Time Warping Algorithm Review. Technical report, University of Hawai'i at Mānoa, 2008.

[118] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781107298019.

[119] Mohammad Sharif and Ali Asghar Alesheikh. Multi-dimensional pattern discovery of trajectories using contextual information. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4W7):31–36, 2017. DOI: 10.5194/isprs-archives-XLII-4-W7-31-2017.

[120] Zhicheng Shi and Lilian S.C. Pun-Cheng. Spatiotemporal data clustering: a survey of methods. *ISPRS International Journal of Geo-Information*, 8(3), 2019. DOI: 10.3390/ijgi8030112.

[121] Katarzyna Siła-Nowicka, Jan Vandrol, Taylor Oshan, Jed A. Long, Urška Demšar, and A. Stewart Fotheringham. Analysis of human mobility patterns from gps trajectories and contextual information. *International Journal of Geographical Information Science*, 30(5):881–906, 2016. DOI: 10.1080/13658816.2015.1100731.

[122] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C.P.L.F. De Carvalho, and Joã Gama. Data stream clustering: a survey. *ACM Computing Surveys*, 46(1), 2013. DOI: 10.1145/2522968.2522981.

[123] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: generalizations and performance improvements. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1057 LNCS:3–17, 1996. DOI: 10.1007/bfb0014140.

[124] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. The move-split-merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2013. DOI: 10.1109/TKDE.2012.88.

[125] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908. DOI: 10.1093/biomet/6.1.1.

[126] Richard Swinburne. *Bayes's Theorem*. Oxford University Press, 2012. DOI: 10.5871/bacad/9780197263419.001.0001.

[127] Chandimal D. Tilakaratne and Liwan Liyanage-Hansen. A review of strengths and weaknesses of spatiotemporal data analysis techniques. In *Proceedings - International Conference on Machine Learning and Data Engineering, iCMLDE 2018*, pages 61–66. Institute of Electrical and Electronics Engineers Inc., 2019. DOI: 10.1109/iCMLDE.2018.00020.

[128] Glen Van Brummelen. *Heavenly mathematics: The forgotten art of spherical trigonometry*. Princeton University Press, 2012.

[129] Shikhar Verma, Yuichi Kawamoto, Zubair Md. Fadlullah, Hiroki Nishiyama, and Nei Kato. A survey on network methodologies for real-time analytics of massive iot data and open research issues. *IEEE Communications Surveys and Tutorials*, 19(3):1457–1477, 2017. DOI: 10.1109/COMST.2017.2694469.

[130] Shuang Wang and Hakan Ferhatosmanoglu. Ppq-trajectory: spatio-temporal quantization for querying in large trajectory repositories. *Proceedings of the VLDB Endowment*, 14(2):215–227, 2020. DOI: 10.14778/3425879.3425891.

[131] Xiaoyu Wang, Xiaofang Zhou, and Sanglu Lu. Spatiotemporal data modelling and management: a survey. In *Proceedings of the Conference on Technology of Object-Oriented Languages and Systems, TOOLS*, pages 202–211. IEEE Comp Soc, 2000.

[132] Thunshun Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005. DOI: 10.1016/j.patcog.2005.01.025.

[133] Christopher J.C.H. Watkins and Peter Dayan. Technical note: q-learning. *Machine Learning*, 8(3):279–292, 1992. DOI: 10.1023/A:1022676722315.

[134] Darrell Whitley. An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, 43(14):817–831, 2001. DOI: 10.1016/S0950-5849(01)00188-4.

[135] Darrell Whitley and Andrew M. Sutton. *Genetic algorithms - a survey of models and methods*, volume 2-4. Springer Berlin Heidelberg, 2012, pages 637–671. DOI: 10.1007/978-3-540-92910-9_21.

[136] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier Inc., 2016.

[137] Jing Wu, Christophe Claramunt, Lamia Belouaer, and Min Deng. A qualitative modelling approach for the representation of trajectories: application to the analysis of flight patterns. *Annals of GIS*, 21(4):275–285, 2015. DOI: 10.1080/19475683.2015.1085439.

[138] Jing Wu, Christophe Claramunt, and Min Deng. Towards a qualitative representation of movement. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8823:191–200, 2014. DOI: 10.1007/978-3-319-12256-4_20.

[139] Rui Xu and Donald C. Wunsch. *Clustering*. John Wiley and Sons, 2008. DOI: 10.1002/9780470382776.

[140] Rui Xu and Donald Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. DOI: 10.1109/TNN.2005.845141.

[141] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Joerg Sander. Distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings - International Conference on Data Engineering*, pages 324–331. IEEE Comp Soc, 1998.

[142] Xin Xu, Guilin Zhang, and Wei Wu. A fast distribution-based clustering algorithm for massive data. *Lecture Notes in Electrical Engineering*, 355:323–330, 2015. DOI: 10.1007/978-3-319-11104-9_38.

[143] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: a deep learning framework for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5668–5675. AAAI Press, 2019. DOI: 10.1609/aaai.v33i01.33015668.

[144] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, 2017. DOI: 10.1007/s10462-016-9477-7.

[145] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1):63–85, 2021. DOI: 10.1007/s41019-020-00151-z.

[146] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 316–324. Association for Computing Machinery, 2011. DOI: 10.1145/2020408.2020462.

[147] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 99–108, 2010. DOI: 10.1145/1869790.1869807.

[148] Dongxiang Zhang, Mengting Ding, Dingy Yang, Yi Liu, Ju Fan, and Heng Tao Shen. Trajectory simplification: an experimental study and quality analysis. *Proceedings of the VLDB Endowment*, 11(9):934–946, 2018. DOI: 10.14778/3213880.3213885.

[149] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):103–114, 1996. DOI: 10.1145/235968.233324.

[150] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3), 2015. DOI: 10.1145/2743025.

[151] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *UbiComp 2008 - Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 312–321. Association for Computing Machinery, 2008. DOI: 10.1145/1409635.1409677.

[152] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: a collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–39, 2010.

[153] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW'09 - Proceedings of the 18th International World Wide Web Conference*, pages 791–800, 2009. DOI: 10.1145/1526709.1526816.

APPENDICES

Appendix A

Spatial-Temporal Cluster Relationships

Spatial-temporal objects, such as vehicles or people, move and form groups called spatial-temporal clusters. These clusters follow the movement of the objects and may enter or leave other clusters, and are said to have spatial-temporal cluster relationships. This study in this thesis identifies 14 spatial-temporal cluster relationships, which are explained in Section 3.2.2. However, the section gives only a high-level explanation of the relationships for simplification purposes and defer in-depth discussion to the appendix. Therefore, the purpose of this appendix is to describe spatial-temporal cluster relationships in a much deeper perspective, discussing how they are defined and how they can be found. This appendix starts with an informal description of the relationships, then presents the language that will be used in a more formal description, and proceeds to describe each spatial-temporal cluster relationship in a more formal way in a subsection.

Spatial-temporal clusters are groups of spatial-temporal objects. Since spatial-temporal objects move with time, it is said that spatial-temporal clusters also move with time. Note that it is possible to identify the location of a spatial-temporal cluster by defining its coordinates to be the average of the coordinates of every object that is in the cluster. This location is not useful in density-based clustering, but simplifies the discussion on the location of a spatial-temporal cluster. Spatial-temporal objects in several different clusters can approach each other and form an even larger cluster. In other words, spatial-temporal clusters can approach each other and merge. Similarly, several spatial-temporal objects inside a large cluster may follow different paths, creating two different groups, or clusters, and dividing the large cluster. In other words, a large spatial-temporal cluster can split, forming other clusters. In other situations, a spatial-temporal cluster may receive a new spatial-temporal object, or lose one of the objects it already contains. Interactions between clusters, such as merge, split, or receive or lose a spatial-temporal object, are spatial-temporal cluster relationships. The proposed approach identifies 14 of these relationships, which are presented in Table 3.1. The following paragraphs present the foundation for the description each of the 14 relationships and each section describe a spatial-temporal cluster relationship in a more formal way.

Let $O = \{o_1, \ldots, o_m\}$ be a set of m spatial-temporal objects. Let o_k be a spatial-temporal object k. Note that spatial-temporal objects exist throughout time. In that case, $o_k = [o_{k,t_j}]$ is a vector of occurrences of a spatial-temporal object, where o_{k,t_j} represents a spatial-temporal object k during timestamp t_j.

Let $C = \{c_1, \ldots, c_n\}$ be a set of n clusters. Let c_i be cluster i. Note that a cluster also exists throughout time. For that reason, c_i is a vector of the occurrences of the clusters c_{i,t_j}, *i.e.* $c_i = [c_{i,t_j}]$, where t_j denotes discrete timestamps. Let c_{i,t_j} be cluster i during timestamp t_j. Once a timestamp is set, a cluster is defined by the spatial-temporal objects it contains. For example, in DBSCAN, a cluster is the set of all of its core and border points (where points represent spatial-temporal objects). Therefore, $c_{i,t_j} = \{o_{k,t_j}\}$ during timestamp t_j, for all o_{k,t_j} that satisfies the particular clustering technique being used. Note that it is assumed that a valid cluster has at least *min_cluster* spatial-temporal objects.

START

The START relationship characterizes the beginning of a cluster. See Figure A.1a for a visual representation.

In DBSCAN, a cluster c_i undergoes a START relationship if

- $c_{i,t_{j-1}} = \varnothing$
- there exists one spatial-temporal object (or data point), say o_{k,t_j}, such that there exists at least *min_cluster* spatial-temporal objects, each of them individually represented by o_{ℓ,t_j}, such that $distance(o_{k,t_j}, o_{\ell,t_j}) < \varepsilon$,

where ε is a parameter that denotes the radius of a core point and is used to calculate density.

The first condition simply states that the cluster does not exist in the previous timestamp. The second condition is a requirement for a density-based cluster to exist, which is that there exist enough spatial-temporal objects (or data points) mutually close, increasing the density.

END

The END relationship characterizes the end of a cluster. See Figure A.1b for a visual representation.

In DBSCAN, a cluster c_i undergoes an END relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- there does not exist a spatial-temporal object (or data point) in c_i, say o_{k,t_j}, such that there exists $min\_cluster$ spatial-temporal objects in c_i, each of them individually represented by o_{ℓ,t_j}, such that $distance(o_{k,t_j}, o_{\ell,t_j}) < \varepsilon$,

where ε is a parameter that denotes the radius of a core point and is used to calculate density.

The first condition simply states that the cluster does exist in the previous timestamp. The second condition falsifies the requirement for a density-based cluster to exist, by asserting that spatial-temporal objects that would form a cluster are not mutually close.

GROUP

Intuitively, a cluster c_i undergoes a GROUP relationship when a group of enough spatial-temporal objects decides to move together. See Figure A.1c for a visual representation. Formally, a cluster c_i undergoes a GROUP relationship if the following three conditions are satisfied:

- $c_{i,t_{j-1}} = \varnothing$
- $c_{i,t_j} \neq \varnothing$
- there does not exist cluster $c_{k,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$.

The first and the second conditions simply assert the creation of the cluster. The third condition states that the number of spatial-temporal objects forming the new cluster and also coming from a preexisting cluster is not enough to form a cluster. If that was the case, the relationship would be a DETACH.

DISPERSE

Intuitively, a cluster c_i undergoes a DISPERSE relationship when its spatial-temporal objects decide to go to different ways. See Figure A.1d for a visual representation. Formally, a cluster c_i undergoes a DISPERSE relationship if the following three conditions are satisfied:

- $c_{i,t_{j-1}} \neq \varnothing$
- $c_{i,t_j} = \varnothing$
- there does not exist cluster c_{k,t_j} such that $|c_{i,t_{j-1}} \cap c_{k,t_j}| \geq min\_cluster$.

The first and the second conditions simply assert the ending of a cluster. The third condition states that the number of spatial-temporal objects spreading from the ending cluster and also going to an existing cluster should not be enough to form a cluster. If that was the case, the relationship would be a JOIN.

JOIN

Intuitively, a cluster c_i undergoes a JOIN relationship when it joins an existing cluster. See Figure A.1e for a visual representation. Formally, a cluster c_i undergoes a JOIN relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- $c_{i,t_j} = \varnothing$
- there exists only one cluster c_{k,t_j} such that $|c_{i,t_{j-1}} \cap c_{k,t_j}| \geq min\_cluster$.

The first and the second conditions simply assert the ending of a cluster. The third condition states that there exists one, and only one, cluster where enough spatial-temporal objects go to. If that was not the case, then the relationship would be a DISPERSE. If there existed more than one cluster where spatial-temporal objects went to, then the relationship would be a SPLIT.

DETACH

Intuitively, a cluster c_i undergoes a DETACH relationship when it leaves an existing cluster. See Figure A.1f for a visual representation. Formally, a cluster c_i undergoes a DETACH relationship if

- $c_{i,t_{j-1}} = \varnothing$
- $c_{i,t_j} \neq \varnothing$
- there exists only one cluster $c_{k,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$.

The first and the second conditions simply assert the creation of a cluster. The third condition states that there exists one, and only one, cluster where enough are coming from. If that was not the case, then the relationship would be a GROUP. If there existed more than one cluster where spatial-temporal objects came from, then the relationship would be a MERGE.

MERGE

Intuitively, a cluster c_i undergoes a MERGE relationship when it is created by the combination (or union) of two or more clusters. See Figure A.1g for a visual representation. Formally, a cluster c_i undergoes a MERGE relationship if

- $c_{i,t_{j-1}} = \varnothing$
- $c_{i,t_j} \neq \varnothing$
- there exists at least two clusters $c_{k,t_{j-1}}$ and $c_{\ell,t_{j-1}}$, $k \neq i$, $\ell \neq i$, such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$ and $|c_{i,t_j} \cap c_{\ell,t_{j-1}}| \geq min\_cluster$.

The first and the second conditions simply assert the creation of a cluster. The third condition states that at least two clusters sent enough spatial-temporal objects to take part in the new cluster formation. If that was not the case, the relationship would be either a GROUP or a C_ENTER.

SPLIT

Intuitively, a cluster c_i undergoes a SPLIT relationship when it disappears by begin divided into two or more clusters. See Figure A.1h for a visual representation. Formally, a cluster c_i undergoes a SPLIT relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- $c_{i,t_j} = \varnothing$
- there exists at least two clusters c_{k,t_j} and c_{ℓ,t_j}, $k \neq i$, $\ell \neq i$, such that $|c_{i,t_{j-1}} \cap c_{k,t_j}| \geq$ $min\_cluster$ and $|c_{i,t_{j-1}} \cap c_{\ell,t_j}| \geq min\_cluster$.

The first and the second conditions simply assert the ending of a cluster. The third condition states that the ending cluster sent enough spatial-temporal objects to at least two clusters. If that was not the case, the relationship would be either a DISPERSE or a C_LEAVE.

C_ENTER

Intuitively, a cluster c_i undergoes a C_ENTER relationship when it receives a group of spatial-temporal objects from a cluster (enough to be a cluster by themselves), ending the sender cluster. Note that this relationship is similar to the JOIN relationship, but it is described from the perspective of the cluster who received another cluster. See Figure A.1i for a visual representation. Formally, a cluster c_i undergoes a C_ENTER relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- $c_{i,t_j} \neq \varnothing$
- there exists a cluster $c_{k,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$
- $c_{k,t_j} = \varnothing$.

The first and the second conditions simply assert that the cluster existed on both timestamps. The third condition states that enough spatial-temporal objects entered the cluster coming from a previously existing cluster. The fourth condition asserts that the previously existing cluster ended. If that was not the case, the relationship would be a C_IN.

C_LEAVE

Intuitively, a cluster c_i undergoes a C_LEAVE relationship when a group of spatial-temporal objects (enough to form a new cluster) leaves the cluster, creating a new cluster. Note that this relationship is similar to the DETACH relationship, but it is described from the perspective of the cluster who released another cluster. See Figure A.1j for a visual representation. Formally, a cluster c_i undergoes a C_LEAVE relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- $c_{i,t_j} \neq \varnothing$
- there exists a cluster c_{k,t_j} such that $|c_{i,t_{j-1}} \cap c_{k,t_j}| \geq min\_cluster$
- and $c_{k,t_{j-1}} = \varnothing$.

The first and the second conditions simply assert that the cluster existed on both timestamps. The third condition states that enough spatial-temporal objects left the cluster to a form a new cluster. The fourth condition asserts that the new cluster did not exist before. If that was not the case, the relationship would be a C_OUT.

T_ENTER

Intuitively, a cluster c_i undergoes a T_ENTER relationship when a spatial-temporal object enters it, either from another cluster or not. See Figure A.1k for a visual representation. Formally, a cluster c_i undergoes a T_ENTER relationship if

- $o_{k,t_{j-1}} \notin c_{i,t_{j-1}}$
- $o_{k,t_j} \in c_{i,t_j}$

These two conditions are required, but not sufficient. More conditions are required depending on whether the spatial-temporal object entered the cluster from another cluster or not. In case o_k did *not* enter the cluster from another cluster, *i.e.* $o_{k,t_{j-1}} \notin c_{\ell,t_{j-1}}$ for all $\ell = 1, \ldots, n$, then one the following two condition is necessary:

- $c_{i,t_{j-1}} \neq \varnothing$
- there exists cluster $c_{\ell,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{\ell,t_{j-1}}| \geq min\_cluster$.

If spatial-temporal object o_k *did* enter the cluster from another cluster, the following two conditions are necessary: let $c_{\ell,t_{j-1}}$ be the cluster o_k came from, *i.e.* $o_{k,t_{j-1}} \in c_{\ell,t_{j-1}}$.

- $|c_{i,t_j} \cap c_{\ell,t_{j-1}}| < min\_cluster$
- there exists at most one cluster $c_{m,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{m,t_{j-1}}| \geq min\_cluster$.

The first two conditions simply assert that a spatial-temporal object entered a cluster. If this spatial-temporal object did not come from a cluster, then the two additional conditions assert that the spatial-temporal object is not part of a GROUP relationship. If this spatial-temporal object did come from a cluster, then the two additional conditions assert that the spatial-temporal object is not part of a C_ENTER (first additional condition) relationship or a MERGE (second additional condition) relationship.

T_LEAVE

Intuitively, a cluster c_i undergoes a T_LEAVE relationship when one of its spatial-temporal objects leaves it, either to another cluster or not. See Figure A.1l for a visual representation. Formally, a cluster c_i undergoes a T_LEAVE relationship if

- $o_{k,t_{j-1}} \in c_{i,t_{j-1}}$
- $o_{k,t_j} \notin c_{i,t_j}$

These two conditions are required, but are not sufficient. More conditions are required depending on whether the spatial-temporal object left to another cluster or not. In case o_k did *not* leave to another cluster, *i.e.* $o_{k,t_j} \notin c_{\ell,t_j}$ for all $\ell = 1, \ldots, n$, then one the following two conditions is necessary:

- $c_{i,t_j} \neq \varnothing$
- there exists cluster c_{ℓ,t_j} such that $|c_{i,t_{j-1}} \cap c_{\ell,t_j}| \geq min\_cluster$.

If spatial-temporal object o_k *did* leave to another cluster, the following two conditions are necessary: let c_{ℓ,t_j} be the cluster o_k went to, *i.e.* $o_{k,t_j} \in c_{\ell,t_j}$.

- $|c_{i,t_{j-1}} \cap c_{\ell,t_j}| < min\_cluster$

- there exists at most one cluster c_{m,t_j} such that $|c_{i,t_{j-1}} \cap c_{m,t_j}| \geq min\_cluster$.

The first two conditions simply assert that a spatial-temporal object left a cluster. If this spatial-temporal object did not leave to a cluster, then the two additional conditions assert that the spatial-temporal object is not part of a DISPERSE relationship. If this spatial-temporal object did leave to a cluster, then the two additional conditions assert that the spatial-temporal object is not part of a C_LEAVE (first additional condition) relationship or a SPLIT (second additional condition) relationship.

C_IN

Intuitively, a cluster c_i undergoes a C_IN relationship when a group of spatial-temporal objects (enough to form a new cluster) leaves a cluster and enters c_i, immediately after. See Figure A.1m for a visual representation. Formally, a cluster c_i undergoes a C_IN relationship if

- $c_{i,t_j} \neq \varnothing$
- there exists a cluster $c_{k,t_{j-1}}$ such that $|c_{i,t_j} \cap c_{k,t_{j-1}}| \geq min\_cluster$
- $c_{k,t_{j-1}} \neq \varnothing$

The first condition simply asserts that the receiving cluster exists after the transition. The second condition states that the number of spatial-temporal objects exchanged is large enough to be deemed a cluster. The third condition states that the cluster who sent spatial-temporal objects existed before the transition.

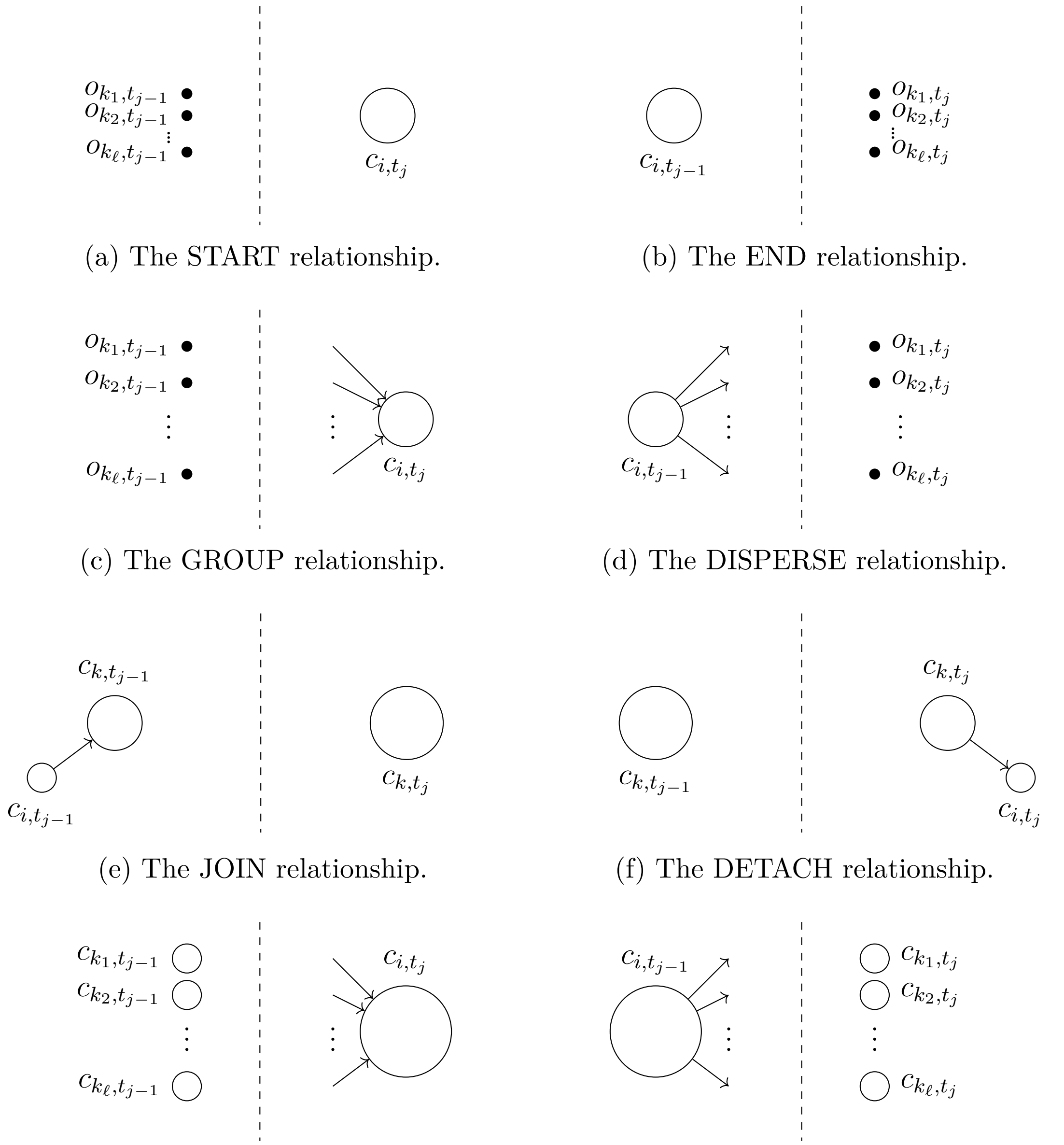
C_OUT

Intuitively, a cluster c_i undergoes a C_OUT relationship when a group of spatial-temporal objects (enough to form a new cluster) leaves c_i and enters another cluster, immediately after. See Figure A.1n for a visual representation. Formally, a cluster c_i undergoes a C_OUT relationship if

- $c_{i,t_{j-1}} \neq \varnothing$
- there exists a cluster c_{k,t_j} such that $|c_{i,t_{j-1}} \cap c_{k,t_j}| \geq min\_cluster$

- $c_{k,t_j} \neq \varnothing$

The first condition simply asserts that the cluster sending spatial-temporal objects exists before the transition. The second condition states that the number of spatial-temporal objects exchanged is large enough to be deemed a cluster. The third condition states that the receiving cluster exists after the transition.



(a) The START relationship.

(b) The END relationship.

(c) The GROUP relationship.

(d) The DISPERSE relationship.

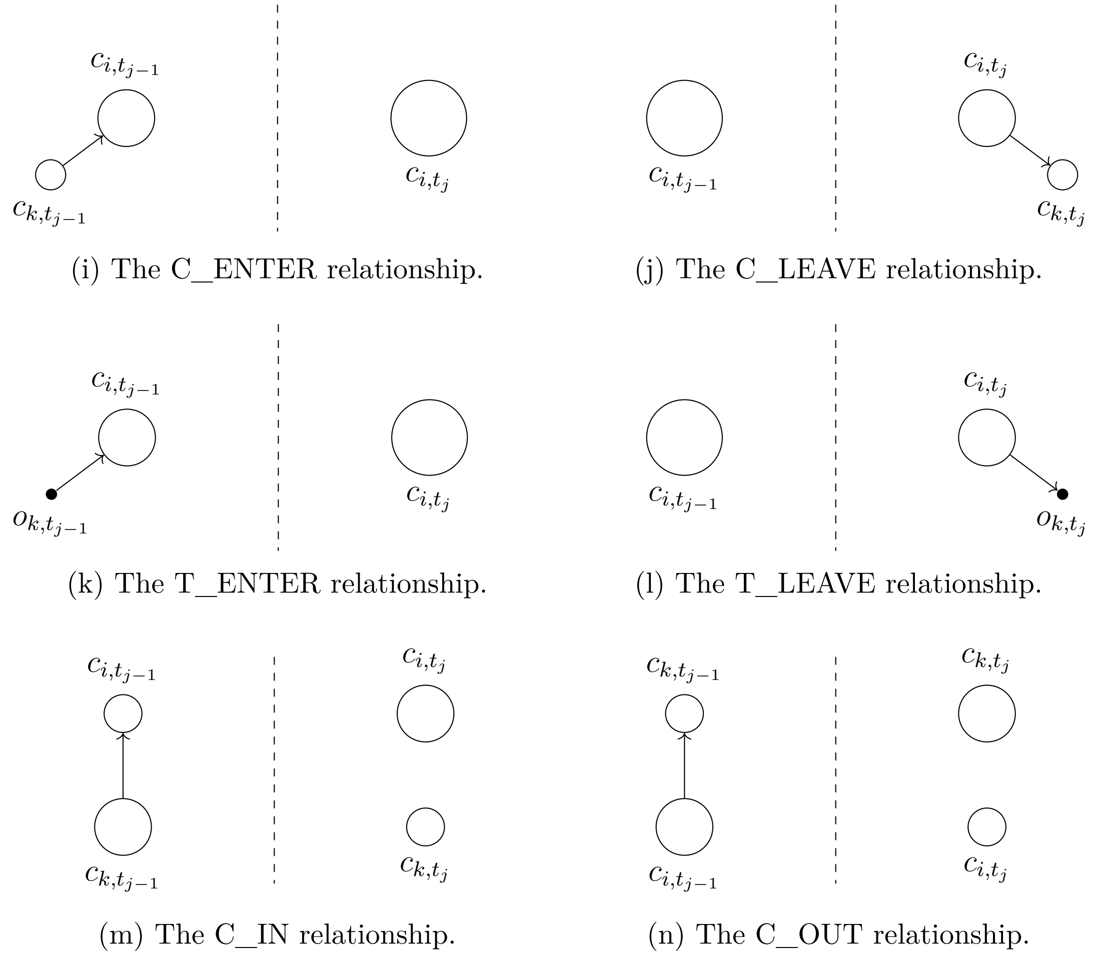
(e) The JOIN relationship.

(f) The DETACH relationship.

(g) The MERGE relationship.

(h) The SPLIT relationship.

Figure A.1: The spatial-temporal cluster relationships.



(i) The C_ENTER relationship.

(j) The C_LEAVE relationship.

(k) The T_ENTER relationship.

(l) The T_LEAVE relationship.

(m) The C_IN relationship.

(n) The C_OUT relationship.

Figure A.1: The spatial-temporal cluster relationships. *Continued from previous page.*

Appendix B

Case Study Queries

This appendix contains the Cypher code for the queries in the four case studies presented in this thesis.

B.1 Case Study 1

Code related to case study 1 is presented here. The codes refer to the queries performed in Section 4.2. The code is presented in the same order as they are described in the section, and in the same order of the questions in Table 4.1.

B.1.1 Structure

This subsection presents the code of queries related to the structure of spatial-temporal clusters.

Show a specific cluster.

Showing a specific spatial-temporal cluster makes its structure visible. This is important for exploratory analyses. The code in Listing B.1 returns cluster c_{100}. The parameter `clusterId` identifies a spatial-temporal cluster through several timestamps. The parameter is set to 100.

```
MATCH (c:Cluster{clusterId:100}) RETURN c
```

Listing B.1: The code to return a specific spatial-temporal cluster.

How many clusters are there in the dataset?

The total number of spatial-temporal clusters in the dataset is an important value depending on the calculations being performed. The code in Listing B.2 returns a count of all spatial-temporal clusters. Note that Neo4j returns nodes, not clusters. Counting the number of nodes would mean counting the number of occurrences of all spatial-temporal clusters in all timestamps. The parameter `DISTINCT` solves this problem.

```
MATCH (c:Cluster) RETURN count(DISTINCT c.clusterId)
```

Listing B.2: The code to return the number of spatial-temporal clusters in the dataset.

How many clusters are there around a specific location?

It is important to know the number of clusters near a location for exploratory studies. The code in Listing B.3 uses latitude and longitude coordinates to specify the location and a range, in meters, to return the number of clusters near the location. The code searches for all clusters and the clause `WHERE` filters the result. The code in Listing B.4 is a new visualization of the results, distributing clusters by the hour of the day.

```
WITH
    41.890278 AS latitude,
    12.492222 AS longitude,
    500 AS maxDistance
MATCH (c:Cluster)
WHERE
    point.distance(c.location, POINT({latitude: latitude, longitude:
   longitude})) ⩽ maxDistance
```

```
RETURN count(DISTINCT c.clusterId) AS Count, collect(DISTINCT c.clusterId)
    AS ClusterID
```

Listing B.3: The code to return the number of spatial-temporal clusters around a specific location.

```
WITH
    41.890278 AS latitude,
    12.492222 AS longitude,
    500 AS maxDistance
MATCH (c:Cluster)
WHERE
    point.distance(c.location, POINT({latitude: latitude, longitude:
   longitude})) ⩽ maxDistance
RETURN c.timestamp.hour AS Hour, count(DISTINCT c.clusterId) AS Number,
   collect(DISTINCT c.clusterId) AS ClusterID ORDER BY Hour
```

Listing B.4: The code to return the number of spatial-temporal clusters around a specific location, but with a different visualization with clusters distributed by the hour of the day.

What clusters start (end) around a specific location?

Analysis on the spatial-temporal clusters that start or end at a specific location is a starting point for further inspections on data. The code in Listing B.5 returns all spatial-temporal clusters that exist around a specific location, defined by latitude and longitude coordinates, and within a range, defined by `maxDistance`. Note the restriction `cStart:True` to indicate that only the first node in the evolution of a spatial-temporal is returned.

```
WITH
    41.896 AS latitude,
    12.4825 AS longitude,
    100 AS maxDistance
MATCH (c:Cluster{cStart:True})
```

```
WHERE point.distance(c.location, POINT({latitude: latitude, longitude:
   longitude})) ⩽ maxDistance
RETURN c.clusterId AS ClusterID ORDER BY ClusterID
```

Listing B.5: The code to return the spatial-temporal clusters that start around a specific location.

What clusters exist for more than a number of minutes?

Spatial-temporal clusters that exist for a longer time may have interesting relationships. For this reason, it is important to identify these clusters so that further analysis can be performed. The code in Listing B.6 returns the clusters as well the time they exist provided that they exist for at least 60 minutes. The calculation of the time a cluster exist is done in `duration.between(c1.timestamp,c2.timestamp)`.

```
MATCH (c1:Cluster{cStart:True})-[:RELATION*0..]→(c2:Cluster{cEnd:True})
WHERE c1.clusterId = c2.clusterId AND
duration.between(c1.timestamp,c2.timestamp).minutes ⩾ 60
RETURN DISTINCT c1.clusterId AS ClusterID
```

Listing B.6: The code to return the spatial-temporal clusters that exist for more than a given number of minutes.

What is the shortest (or longest) cluster with respect to the time it exists?

For some statistical analysis, extreme values such as shortest or longest, are important. These values can be used for normalization purposes and other advanced calculations. The code in Listing B.7 returns the clusters as well as the time they exist, but present the results in descending order such that the longest spatial-temporal clusters is at the top. A variation of this code has `LIMIT 1` at the end to return only the first entry, that is the longest cluster if there are no draws.

```
MATCH (c1:Cluster{cStart:True})-[:RELATION*0..]->(c2:Cluster{cEnd:True})
WHERE c1.clusterId = c2.clusterId
RETURN DISTINCT c1.clusterId AS ClusterID, duration.between(c1.timestamp,
   c2.timestamp).minutes AS Duration ORDER BY Duration DESC
```

Listing B.7: The code to return the longest cluster with respect to the time it exists in the dataset.

What is the largest cluster with respect to the number of spatial-temporal objects?

A spatial-temporal cluster containing many spatial-temporal objects is a sign of important phenomena and should be further investigated. The code in Listing B.8 returns all nodes, or occurrences of spatial-temporal clusters in timestamps and `max(c.size)` aggregates the result, showing only the cluster and the largest number of spatial-temporal objects that it has at some point during its existence.

```
MATCH (c:Cluster)
RETURN c.clusterId AS ClusterID, max(c.size) AS Size ORDER BY Size DESC
```

Listing B.8: The code to return spatial-temporal clusters and the maximum number of spatial-temporal objects that they contain at any point during their existence.

What clusters have more than a given number of spatial-temporal objects?

Depending on the investigation, it may be interesting to list all spatial-temporal clusters that contain, at any point during its existence, a number of spatial-temporal objects greater than a given value. The code in Listing B.9 returns all spatial-temporal clusters filtered by the restriction on the size. This restriction is imposed by the code in `WHERE c.size > 10`.

```
MATCH (c:Cluster)
```

```
WHERE c.size > 10
RETURN DISTINCT c.clusterId AS ClusterID ORDER BY ClusterID
```

Listing B.9: The code to return spatial-temporal clusters that have more than a given number of spatial-temporal objects at any point during their existence.

B.1.2 Relationships

This subsection presents the code of queries related to the relationships of spatial-temporal clusters.

What clusters start with a MERGE relationship? (Or What clusters end with a SPLIT relationship?)

The way spatial-temporal clusters start or end their existence is described by the relationship the cluster has at the suitable moment. This relationship can be queried for additional data about the investigation of important phenomena. The code in Listing B.10 returns a list of all spatial-temporal clusters that start with a MERGE relationship. Note that the notation `()-[r:Relationship]->(c:ClustercStart:True)` seeks for nodes `c` having an incoming relationship `r`. The code in Listing B.11 performs a similar query, but returns spatial-temporal clusters that end in a SPLIT relationship.

```
MATCH ()-[r:Relationship]->(c:Cluster{cStart:True})
WHERE r.clusterRelationship = 'MERGE'
RETURN DISTINCT c.clusterId AS ClusterID ORDER BY ClusterID
```

Listing B.10: The code to return spatial-temporal clusters that start in a MERGE relationship.

```
MATCH (c:Cluster{cEnd:True})-[r:Relationship]->()
WHERE r.clusterRelationship = 'SPLIT'
```

```
RETURN DISTINCT c.clusterId AS ClusterID ORDER BY ClusterID
```

Listing B.11: The code to return spatial-temporal clusters that end in a SPLIT relationship.

Show the locations where T_ENTER (or T_LEAVE, MERGE, SPLIT, etc.) relationships happen.

Information about the location of spatial-temporal cluster relationships can help in exploratory investigations about important phenomena. The code in Listing B.12 returns the location of every T_ENTER relationship on the dataset. Since relationships happen between timestamps, the location is the average between the latitude coordinate or longitude coordinate of the cluster before and after the relationship. Note that the restriction `{clusterRelationship:'T_ENTER'}` can be updated to perform queries on other spatial-temporal cluster relationships.

```
MATCH (c1:Cluster)-[r:Relationship{clusterRelationship:'T_ENTER'}]->(c2:
   Cluster)
RETURN DISTINCT
    (c1.location.latitude + c2.location.latitude)/2 AS Latitude,
    (c1.location.longitude + c2.location.longitude)/2 AS Longitude
```

Listing B.12: The code to return the location of a given spatial-temporal relationship in the dataset.

What clusters have more than a given number of T_ENTER (or T_LEAVE) relationships?

In general, spatial-temporal clusters have a number of relationships. However, some clusters present an uncommon number of these relationships, which may indicate important phenomena that is worth investigating. The code in Listing B.13 returns all clusters with the additional restriction that, in the sequence of nodes that represent this spatial-temporal cluster, there should be more than a given number of spatial-temporal cluster relationships of a given type. In this specific case, the code presents clusters with more than 15 T_ENTER relationships, given by the second restriction in the WHERE clause. The code first uses a list comprehension operation to create a list of all occurrences of a given relationship, as in [r in relationships(path) WHERE r.clusterRelationship = 'T_ENTER'|r], and then uses the function size() to count the number of elements in the list.

```
MATCH path = (c1:Cluster{cStart:True})-[:Relationship*0..]->(c2:Cluster{
   cEnd:True})
WHERE
    c1.clusterId = c2.clusterId AND
    size([r in relationships(path) WHERE r.clusterRelationship = 'T_ENTER'
   |r]) > 15
RETURN DISTINCT c1.clusterId AS ClusterID
```

Listing B.13: The code to return spatial-temporal clusters that have more than 15 T_ENTER relationships during their existence.

What clusters have a MERGE relationship before a given time?

Filtering the occurrence of spatial-temporal cluster relationships based on the time can help identify phenomena at specific moments of the day. The code in Listing B.14 returns clusters who have a MERGE relationship before 8 AM. The filter per hour is in WHERE c.timestamp.hour < 8. Because Neo4j uses a 24-hour clock, there is no ambiguity with 8 PM. Notice that the restriction }clusterRelationship:'MERGE'} can be updated for other spatial-temporal cluster relationships.

```
MATCH ()-[r:Relationship{clusterRelationship:'MERGE'}]->(c:Cluster{cStart:
   True})
WHERE c.timestamp.hour < 8
RETURN DISTINCT c.clusterId AS ClusterID ORDER BY ClusterID
```

Listing B.14: The code to return spatial-temporal clusters that have a MERGE relationship before 8 AM.

B.1.3 Structure

This subsection presents the code of queries related to the graph representation of spatial-temporal cluster evolution.

Show the evolution of a given cluster, that is all the cluster evolution paths starting at the given cluster.

Cluster evolution paths are a sequence of nodes representing spatial-temporal clusters that are connected and not restricted by only one cluster. From a start node, there could be many cluster evolution paths. Investigating cluster evolution paths give several insights about the movement of objects and help find important phenomena. The code in Listing B.15 returns all cluster evolution paths from a given spatial-temporal cluster. The chosen cluster is c_{505}. The restrictions `{pStart:True, clusterId:5805}` substitute the `WHERE` clause for restrictions and direct the query to look for nodes that start a cluster evolution path and belong to the evolution of cluster c_{5805}.

```
MATCH path = (c1:Cluster{pStart:True, clusterId:5805})-[:Relationship*0..{
   pMain:True}]->(c2:Cluster{pEnd:True})
RETURN path AS ClusterEvolutionPath
```

Listing B.15: The code to return all the cluster evolution paths starting at the beginning of the spatial-temporal cluster c_{5805}.

How many cluster evolution paths there exists in the database?

In exploratory analysis, it is important to know the total number of properties or characteristics, such as the total number of cluster evolution paths in the database. This number can be used for performance improvements or in other types of analyses. The code in Listing B.16 returns the number of all cluster evolution paths that can be found in the database. Note that, as cluster evolution paths are sequence of nodes, and not edges, only one of possibly many edges between two nodes is enough to represent a connection between these nodes. For this reason, and for performance reasons, the restriction `{pMain:True}` serves as an optimization.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
RETURN count(path) AS NumberOfClusterEvolutionPaths
```

Listing B.16: The code to return the number of all cluster evolution paths in the database.

Show all cluster evolution paths and the location they start and end.

Additional information about cluster evolution paths, such as the location they start or end, can be used to explain the movement they represent or to begin a deeper investigation. The code in Listing B.17 returns every cluster evolution path in the database alongside the location they started and ended. These locations are described in terms of their latitude and longitude coordinates, which are stored in `c1.location.latitude` or `c1.location.longitude` for the start position and similarly for the end position.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
RETURN
    apoc.coll.toSet([c IN nodes(path) | c.clusterId]) AS
   ClusterEvolutionPaths,
    c1.location.latitude  AS StartLatitude,
    c1.location.longitude AS StartLongitude,
    c2.location.latitude  AS EndLatitude,
    c2.location.longitude AS EndLongitude
```

Listing B.17: The code to return all cluster evolution paths and the location they start and end.

What cluster evolution paths start (end) around a specific location?

When specifying a location of interest, spatial-temporal clusters that start, or end, at this location can be retrieved, examined, to identify interesting phenomena. The code in Listing B.18 returns the cluster evolution paths that start around the location defined by the coordinates in `latitude` and `longitude`, within a radius of 500 meters.

```
WITH
    41.794700 AS latitude,
    12.250700 AS longitude,
    500 AS maxDistance
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
WHERE point.distance(c1.location, POINT({latitude: latitude, longitude:
   longitude})) <= maxDistance
RETURN apoc.coll.toSet([n IN nodes(path) | n.clusterId]) AS
   ClusterEvolutionPaths
```

Listing B.18: The code to return all cluster evolution paths that start around a specific location.

Show the location of every cluster change in all cluster evolution paths.

Cluster evolution paths are a sequence of cluster occurrences in timestamps. Depending on the evolution of a particular cluster, there could be a cluster change in this sequence. For example, if a spatial-temporal cluster c_1 splits into clusters c_2 and c_3, the cluster evolution path from the beginning of c_1 to the end of c_2 contains a cluster change. Investigating the location of these cluster changes identify the location where decisions are made, especially the ones that result in a change of cluster. The code in Listing B.19 returns all cluster evolution paths and the location where cluster changes happen. Note that the restriction `WHERE c1.clusterId <> c2.clusterId` limits the results to paths that necessarily have a cluster change. Since a cluster path can have more than one cluster change, the result is shown as a list of locations.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
WHERE c1.clusterId <> c2.clusterId
WITH path, apoc.coll.pairs(nodes(path)) AS pairs
UNWIND([pair IN pairs WHERE pair[0].clusterId <> pair[1].clusterId | [(
   pair[0].location.latitude + pair[1].location.latitude)/2, (pair[0].
   location.longitude + pair[1].location.longitude)/2]]) AS location
```

```
RETURN apoc.coll.toSet([n IN nodes(path) | n.clusterId]) AS
   ClusterEvolutionPath, collect(DISTINCT location) AS Location
```

Listing B.19: The code to return the location of every cluster change in all cluster evolution paths.

What are the cluster evolution paths that start in the morning and end in the afternoon?

Limiting the investigation of spatial-temporal clusters to moments of the day results in a deeper investigation of some phenomena of interest. The code in Listing B.20 returns all cluster evolution paths that start in the morning and end in the afternoon alongside their start and end times. Morning is defined as the time between 6 AM and 12 PM and is represented in the code by the restrictions `c1.timestamp.hour >= 6` and `c1.timestamp.hour < 12`. Afternoon is defined as the time between 12 PM and 6 PM and is represented, in a 24-hour clock, by the restrictions `c2.timestamp.hour >= 12` and `c2.timestamp.hour < 18`.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
WHERE
    c1.timestamp.hour >= 6 AND
    c1.timestamp.hour < 12 AND
    c2.timestamp.hour >= 12 AND
    c2.timestamp.hour < 18
RETURN
    apoc.coll.toSet([n IN nodes(path)|n.clusterId]) AS
   ClusterEvolutionPaths,
    c1.timestamp AS PathStartTime,
    c2.timestamp AS PathEndTime
```

Listing B.20: The code to return all cluster evolution paths that start in the morning and end in the afternoon.

Show all cluster evolution paths and the time they start and end.

As a more generic investigation than the one in the previous query, an exploratory investigation may list all cluster evolution paths and the time they start and end to identify interesting phenomena, such as a common time for cluster creation. The code in Listing B.21 returns all cluster evolution paths in the database alongside their start and end time, which are stored in `c1.timestamp` and `c2.timestamp`.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
RETURN
    apoc.coll.toSet([c IN nodes(path) | c.clusterId]) AS
   ClusterEvolutionPaths,
    c1.timestamp AS StartTimestamp,
    c2.timestamp AS EndTimeStamp
```

Listing B.21: The code to return all cluster evolution paths in the database alongside the time they start and end.

What is the cluster evolution path with the greatest number of relationships?

Investigating data that presents extreme values, such as the cluster evolution paths that have the greatest number of relationships, help uncover interesting phenomena that is not visible at first. The code in Listing B.22 returns the cluster evolution path with the greatest number of relationships. The restriction `LIMIT 1` instructs the code to return only the result at the top of the result table, which is the expected result, since the results are ordered. However, in the case of a draw, important results can be missing.

```
MATCH path = (c1:Cluster{pStart:True})-[:RELATION*0..{pMain:True}]->(c2:
   Cluster{pEnd:True})
RETURN
    apoc.coll.toSet([n IN nodes(path)|n.clusterId]) AS
   ClusterEvolutionPaths,
    size(relationships(path)) AS NumOfRelationships
ORDER BY NumOfRelationships DESC
```

```
LIMIT 1
```

Listing B.22: The code to return the cluster evolution path with the greatest number of spatial-temporal relationships.

What cluster evolution paths have more (or less) than a given number of relationships?

The analysis of the cluster evolution path with the greatest number of relationship help identify interesting phenomena, however, the analysis on all cluster evolution paths that have more, or less, than a given number of spatial-temporal cluster relationship help identify common results that can be further investigated to identify other types of interesting phenomena. The code in Listing B.23 returns all cluster evolution paths that have more than 100 relationships, alongside their actual number of relationships. The restriction on the number of relationships is at the `WHERE` clause `size(relationships(path)) > 100`, which can be updated as needed.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
WHERE size(relationships(path)) > 100
RETURN
    apoc.coll.toSet([n IN nodes(path)|n.clusterId]) AS
   ClusterEvolutionPaths,
    size(relationships(path)) AS NumOfRelationships
ORDER BY NumOfRelationships
```

Listing B.23: The code to return the cluster evolution paths that have more than 100 spatial-temporal relationships.

What is the cluster evolution path with the greatest number of spatial-temporal objects during its existence?

Spatial-temporal clusters are formed by spatial-temporal objects that group during their movement. Some of these clusters contain many of these objects, which indicates that further investigation can identify interesting phenomena or an outlier case. The code in Listing B.24 returns the cluster evolution path that contains the greatest number of spatial-temporal objects at some point in its existence among all paths in the database. The aggregating function `max()` and the list comprehension notation together in `max([n IN nodes(path)|n.size])` calculate the maximum number of spatial-temporal objects at any point during the existence of a given cluster evolution path, and since results are ordered by this number, the code is instructed to return only the first entry of the result table.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
RETURN
    apoc.coll.toSet([n IN nodes(path)|n.clusterId]) AS
   ClusterEvolutionPaths,
    apoc.coll.max([n IN nodes(path)|n.size]) AS MaxNumOfObjects
ORDER BY MaxNumOfObjects DESC
LIMIT 1
```

Listing B.24: The code to return the cluster evolution path with the greatest number of spatial-temporal objects during its existence.

What is the cluster evolution path with the greatest average number of spatial-temporal objects during its existence?

A more advanced investigation avenue is to examine cluster evolution paths that have the greatest average of spatial-temporal objects during existence, instead of paths that have the greatest number of objects at any point during existence. Both approaches are valuable, but return different types of results. The code in Listing B.25 returns the cluster evolution paths with the greatest average number of spatial-temporal objects during its existence. The average is calculated in `(maxNumOfObjects - minNumOfObjects)/numOfClusters` and is based on the number of cluster occurrences that are transformed into nodes in the database. Other approaches may average the result by the time the cluster exists or by maximum number of objects the cluster has at any point during existence.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
WITH
    path,
    apoc.coll.max([n IN nodes(path)|n.size]) AS maxNumOfObjects,
    apoc.coll.min([n IN nodes(path)|n.size]) AS minNumOfObjects,
    size(apoc.coll.toSet([n IN nodes(path)|n.clusterId])) AS numOfClusters
RETURN
    apoc.coll.toSet([n IN nodes(path)|n.clusterId]) AS
   ClusterEvolutionPaths,
    (maxNumOfObjects - minNumOfObjects)/numOfClusters AS AvgNumOfObjects
ORDER BY AvgNumOfObjects DESC
LIMIT 1
```

Listing B.25: The code to return the cluster evolution path with the greatest average number of spatial-temporal objects during its existence.

B.2 Case Study 2

Code related to case study 2 is presented here. The code refers to the analysis presented in Section 4.3 to generate the results Table 4.23.

Case study 2 investigates the ever-increasing or ever-decreasing regions and the calculation of an AROC. The code in Listing B.26 returns cluster evolution paths that contain lengthy ever-increasing regions alongside other information about them. This information includes the spatial-temporal cluster at the start and end of the path, the start and end times, and the number of spatial-temporal objects at the start and end of the path. The code starts with a usual `MATCH` clause and uses the `pairsMin()` function from the apoc library to traverse the path searching for pairs of nodes that contain an increase or stabilization of the number of participants. Each pair that matches this restriction is marked with `True`, or `False` otherwise. Then the procedure `split()`, also from the apoc library, regions of increase by splitting the list of pairs whenever a pair marked as `False` is found. The `max()` filters the results, identifying the ever-increasing regions. The rest of the code aggregates results for cases where more than one ever-increasing region is found. Information about the cluster evolution paths returned by the code is discussed in the respective section.

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]->(
   c2:Cluster{pEnd:True})
CALL apoc.coll.split([pair IN apoc.coll.pairsMin(nodes(path)) | CASE pair[
   0].size <= pair[1].size WHEN True THEN pair ELSE False END],False)
   YIELD value AS pairs
WITH path, collect(apoc.coll.toSet(apoc.coll.flatten(pairs))) AS
   tempSequence, max(size(pairs))+1 AS maxLength
UNWIND tempSequence AS sequence
WITH path, sequence, maxLength
WHERE size(sequence) = maxLength
RETURN
    apoc.coll.toSet([n in nodes(path)|n.clusterId]) AS
   ClusterEvolutionPath,
    sequence[ 0].clusterId AS StartCluster,
    sequence[-1].clusterId AS EndCluster,
    sequence[ 0].timestamp AS StartTime,
    sequence[-1].timestamp AS EndTime,
    sequence[ 0].size AS StartSize,
    sequence[-1].size AS EndSize
```

Listing B.26: The code to return cluster evolution paths with lengthy ever-increasing regions alongside information about their start and end cluster, time, and size.

B.3 Case Study 3

Code related to case study 3 is presented here. The code refers to the analysis presented in Section 4.4 to identify the start of cluster evolution paths, shown in Table 4.24, and to generate part of the results Table 4.25.

Case study 3 investigates the calculation of a similarity value between cluster evolution paths. The process towards the results has two steps. The first step is to generate a list of spatial-temporal clusters that start cluster evolution paths. These clusters are then plotted on a map in QGIS and the ones near the place of interest are selected, creating a filtered list. The second step is to use the filtered list of spatial-temporal clusters to retrieve and calculate important information about the cluster evolution paths they start. The code in Listing B.27 returns the list of spatial-temporal clusters that start cluster evolution paths. Note that the code is optimized and contains four restrictions. The first restriction, `c1.pEnd IS NULL` does not allow the path to end on its first node. The second restriction, `c2.pEnd IS NULL` does not allow the path to end on its second node, removing trivial cases. The third restriction, `id(c1) <> id(c2)` does not allow the cluster to include a loop at the first node. The fourth restriction, `(c:ClusterpStart:True)-->(c2:Cluster)` forces the path to include at least two nodes. The resulting list is then plotted and the filtered list is created. The code in Listing B.28 returns information about the cluster evolution paths started by spatial-temporal clusters in the filtered list. The filtered list is supposed to be provided in `WITH [] AS ListOfClusters`, then a series of calculations and aggregations are performed to the final result. Code for the calculation of the similarity is not performed in Neo4j and is not presented in this appendix.

```
MATCH (c1:Cluster{pStart:True})-->(c2:Cluster)
WHERE
    c1.pEnd IS NULL AND
    c2.pEnd IS NULL AND
    id(c1) <> id(c2)
RETURN DISTINCT c1.clusterId, c1.location.latitude AS latitude, c1.
   location.longitude AS longitude
```

Listing B.27: This code returns a list of spatial-temporal clusters that start cluster evolution paths. The results are restricted to filter out possible uninteresting paths.

```
WITH [] AS ListOfClusters

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*0..{pMain:True}]-> (
   c2:Cluster{pEnd:True})
WHERE c1.clusterId IN ListOfClusters
RETURN
    c1.timestamp.hour AS Hour,
    count(c1.clusterId) AS NumberClustersThatStartClusterEvolutionPaths,
    apoc.coll.toSet(collect(c1.clusterId)) AS
   ClustersThatStartClusterEvolutionPaths,
    size(apoc.coll.toSet(apoc.coll.flatten(collect([n in nodes(path)|n.
   clusterId])))) AS NumberOfClustersInAllClusterEvolutionPaths,
    count(path) AS NumberOfClusterEvolutionPaths
ORDER BY Hour
```

Listing B.28: This code returns information about cluster evolution paths that start with spatial-temporal clusters from a provided list.

B.4 Case Study 4

Code related to case study 4 is presented here. The code refers to the analysis presented in Section 4.5 on the movement represented by cluster evolution paths. The section divides the analysis in four questions and discusses their results. The presentation of the code is divided in four smaller subsections, each for a different question, following the same structure of the section in the thesis.

Case study 4 investigates the movement that cluster evolution paths represent. In general, movement can be described in many ways, such as by its start location, start time, direction, or distance. The analysis explores the connected, graph-based representation of cluster evolution paths to produce results that are aggregated and distributed by different units, such as time of the day, or region. The analysis is divided in four questions, related to direction of movements, their distribution by region and by day, their start time, and their distance. Each of the aspects described are analyzed for cluster evolution paths that start near two train stations. The code in each section is agnostic to the train station being analyzed because it depends solely on the list of clusters provided for the analysis.

Where are clusters evolution paths moving towards?

Analysis on the direction of movement of spatial-temporal clusters indicate the common ways spatial-temporal objects navigate in space. Aggregated results help identify the most important directions towards which improvements can be made. In this question, direction is based on the primary and secondary cardinal points, forming eight directions. The code in Listing B.29 returns each of the eight cardinal points as well as the percentage of cluster evolution paths that moved towards this direction. The results are presented in Table 4.26. To calculate the direction, the cosine of the angle that the line that passes through the cluster evolution path start and end locations has with an approximated horizontal line is found using trigonometric relations based on the latitude and longitude coordinates of the locations. The function `point.distance()` is used for this end. Then, the arc whose cosine has just been found is calculated using the function `acos()`. Lastly, the value of the arc in degrees is returned by the function `degrees()`. The angle in degrees is then compared to a predefined list of angles using the `CASE` clause which defines the direction. Note that the table contain both positive and negative values for angles for cases in which the function `degrees()` is updated to return negative angles. The results are then aggregated and the percentages are calculated.

```
WITH [] AS ListOfClusters

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]->(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH count(path) AS numOfClusterEvolutionPaths, ListOfClusters

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]->(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH numOfClusterEvolutionPaths, path, c1, c2,
    degrees(acos(
        point.distance(
            point({longitude: c1.location.longitude, latitude: c1.location
   .latitude}),
            point({longitude: c2.location.longitude, latitude: c1.location
   .latitude})
        ) /
        point.distance(c1.location, c2.location)
    )) AS degree
WITH numOfClusterEvolutionPaths, path,
    CASE
        WHEN degree >= 0     AND degree < 22.5    THEN 'E'
        WHEN degree >= 22.5  AND degree < 67.5    THEN 'NE'
        WHEN degree >= 67.5  AND degree < 112.5   THEN 'N'
        WHEN degree >= 112.5 AND degree < 157.5   THEN 'NW'
        WHEN degree >= 157.5 AND degree < 202.5   THEN 'W'
        WHEN degree >= 202.5 AND degree < 247.5   THEN 'SW'
        WHEN degree ⩾  247.5 AND degree < 292.5   THEN 'S'
        WHEN degree ⩾  292.5 AND degree < 337.5   THEN 'SE'
        WHEN degree ⩾  337.5 AND degree < 360     THEN 'E'
        WHEN degree < 0      AND degree ⩾  -22.5  THEN 'E'
        WHEN degree < -22.5  AND degree ⩾  -67.5  THEN 'SE'
        WHEN degree < -67.5  AND degree ⩾  -112.5 THEN 'S'
        WHEN degree < -112.5 AND degree ⩾  -157.5 THEN 'SW'
        WHEN degree < -157.5 AND degree ⩾  -202.5 THEN 'W'
        WHEN degree < -202.5 AND degree ⩾  -247.5 THEN 'NW'
        WHEN degree < -247.5 AND degree ⩾  -292.5 THEN 'N'
        WHEN degree < -292.5 AND degree ⩾  -337.5 THEN 'NE'
        WHEN degree < -337.5 AND degree ⩾  -360   THEN 'E'
    END AS direction
WITH
    direction,
    size(collect(path)) AS numOfClusterEvolutionPathsPerDirection,
    numOfClusterEvolutionPaths
RETURN
    direction AS CardinalDirection,
    toFloat(numOfClusterEvolutionPathsPerDirection)/toFloat(
   numOfClusterEvolutionPaths) * 100 AS Percentage
```

Listing B.29: The code to return a distribution of the movement represented by cluster evolution paths based on the direction.

What is the distribution of cluster evolution paths by region and by day?

Further analysis on the movement of spatial-temporal clusters include a distribution of the movement by region and by day. Specific results provide an in-depth analysis of the phenomena under investigation that can possibly improve the quality of the results. In this question, a region is simply one of the eight cardinal points discussed previously and a day refers to the day of the start of the movement represented by the cluster evolution path. The code in Listing B.30 returns the day and the percentage of spatial-temporal clusters that exist on this day and are near the respective train station, the region where the majority of movements represented by cluster evolution paths pointed towards on the respective day and train station, and the percentage of movements that followed this direction on that day and train station. This results are presented in Table 4.27. Note that the code starts with `WITH [] AS ListOfClusters` so that only spatial-temporal clusters starting on each train station are analyzed at a time. Unfortunately, calculating percentages in Neo4j and Cypher can produce very long code because two queries are needed. The first query finds the total value of some measurement, to be used as the denominator, and a second query finds the number of interest to be used as the numerator. The code performs several queries on the database to retrieve data to calculate values such as `numOfClusterEvolutionPaths`, firacodenumOfClusterEvolutionPathsPerDate, and `maxNumOfClusterEvolutionPathsPerDirectionPerDate`, whose names are self-explanatory. The calculation of the direction of movement is the same performed in the previous question.

```
WITH [] AS ListOfClusters


// Percentage

//   numOfClusterEvolutionPaths

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH
    count(path) AS numOfClusterEvolutionPaths,
    ListOfClusters


//   numOfClusterEvolutionPathsPerDate

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH
    Date(c1.timestamp) AS Date,
    count(path) AS numOfClusterEvolutionPathsPerDate,
    numOfClusterEvolutionPaths,
    ListOfClusters


// Region

//   maxNumOfClusterEvolutionPathsPerDirectionPerDate
```

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters

WITH
    Date,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    ListOfClusters,
    path,
    degrees(acos(
        point.distance(
            point({longitude: c1.location.longitude, latitude: c1.location
   .latitude}),
            point({longitude: c2.location.longitude, latitude: c1.location
   .latitude})
        ) /
        point.distance(c1.location, c2.location)
    )) AS degree
WHERE date(c1.timestamp) = Date
WITH
    Date,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    ListOfClusters,
    path,
    CASE
        WHEN degree ⩾ 0     AND degree < 22.5    THEN 'E'
        WHEN degree ⩾ 22.5  AND degree < 67.5    THEN 'NE'
        WHEN degree ⩾ 67.5  AND degree < 112.5   THEN 'N'
        WHEN degree ⩾ 112.5 AND degree < 157.5   THEN 'NW'
        WHEN degree ⩾ 157.5 AND degree < 202.5   THEN 'W'
        WHEN degree ⩾ 202.5 AND degree < 247.5   THEN 'SW'
        WHEN degree ⩾ 247.5 AND degree < 292.5   THEN 'S'
        WHEN degree ⩾ 292.5 AND degree < 337.5   THEN 'SE'
        WHEN degree ⩾ 337.5 AND degree < 360     THEN 'E'
        WHEN degree < 0      AND degree ⩾ -22.5  THEN 'E'
        WHEN degree < -22.5  AND degree ⩾ -67.5  THEN 'SE'
        WHEN degree < -67.5  AND degree ⩾ -112.5 THEN 'S'
        WHEN degree < -112.5 AND degree ⩾ -157.5 THEN 'SW'
        WHEN degree < -157.5 AND degree ⩾ -202.5 THEN 'W'
        WHEN degree < -202.5 AND degree ⩾ -247.5 THEN 'NW'
        WHEN degree < -247.5 AND degree ⩾ -292.5 THEN 'N'
        WHEN degree < -292.5 AND degree ⩾ -337.5 THEN 'NE'
        WHEN degree < -337.5 AND degree ⩾ -360   THEN 'E'
    END AS direction
WITH
    direction,
    size(collect(path)) AS numOfClusterEvolutionPathsPerDirectionPerDate,
    Date,
    ListOfClusters,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate
WITH
    Date,
    max(numOfClusterEvolutionPathsPerDirectionPerDate) AS
   maxNumOfClusterEvolutionPathsPerDirectionPerDate,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    ListOfClusters


//   numOfClusterEvolutionPathsPerDirectionPerDate

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters

WITH
    Date,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    maxNumOfClusterEvolutionPathsPerDirectionPerDate,
    path,
    degrees(acos(
        point.distance(
            point({longitude: c1.location.longitude, latitude: c1.location
   .latitude}),
            point({longitude: c2.location.longitude, latitude: c1.location
   .latitude})
        ) /
        point.distance(c1.location, c2.location)
    )) AS degree
```

```
WHERE date(c1.timestamp) = Date
WITH
    Date,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    maxNumOfClusterEvolutionPathsPerDirectionPerDate,
    path,
    CASE
        WHEN degree ⩾  0     AND degree < 22.5    THEN 'E'
        WHEN degree ⩾  22.5  AND degree < 67.5    THEN 'NE'
        WHEN degree ⩾  67.5  AND degree < 112.5   THEN 'N'
        WHEN degree ⩾  112.5 AND degree < 157.5   THEN 'NW'
        WHEN degree ⩾  157.5 AND degree < 202.5   THEN 'W'
        WHEN degree ⩾  202.5 AND degree < 247.5   THEN 'SW'
        WHEN degree ⩾  247.5 AND degree < 292.5   THEN 'S'
        WHEN degree ⩾  292.5 AND degree < 337.5   THEN 'SE'
        WHEN degree ⩾  337.5 AND degree < 360     THEN 'E'
        WHEN degree < 0      AND degree ⩾  -22.5  THEN 'E'
        WHEN degree < -22.5  AND degree ⩾  -67.5  THEN 'SE'
        WHEN degree < -67.5  AND degree ⩾  -112.5 THEN 'S'
        WHEN degree < -112.5 AND degree ⩾  -157.5 THEN 'SW'
        WHEN degree < -157.5 AND degree ⩾  -202.5 THEN 'W'
        WHEN degree < -202.5 AND degree ⩾  -247.5 THEN 'NW'
        WHEN degree < -247.5 AND degree ⩾  -292.5 THEN 'N'
        WHEN degree < -292.5 AND degree ⩾  -337.5 THEN 'NE'
        WHEN degree < -337.5 AND degree ⩾  -360   THEN 'E'
    END AS direction
WITH
    direction,
    size(collect(path)) AS numOfClusterEvolutionPathsPerDirectionPerDate,
    Date,
    numOfClusterEvolutionPaths,
    numOfClusterEvolutionPathsPerDate,
    maxNumOfClusterEvolutionPathsPerDirectionPerDate
WHERE numOfClusterEvolutionPathsPerDirectionPerDate =
   maxNumOfClusterEvolutionPathsPerDirectionPerDate


RETURN
    Date,
    toFloat(numOfClusterEvolutionPathsPerDate)/toFloat(
   numOfClusterEvolutionPaths) * 100 AS Percentage,
    direction AS Region,
    toFloat(maxNumOfClusterEvolutionPathsPerDirectionPerDate)/toFloat(
   numOfClusterEvolutionPathsPerDate) * 100 AS PercentageFollowingRegion
```

```
ORDER BY Date
```

Listing B.30: The code to return a distribution of the movement represented by cluster evolution paths per region and per day.

What time of the day are clusters evolution paths forming?

Analysis on the time the movement represented by cluster evolution paths start can help identify edge cases or common times that can hold important information about spatial-temporal phenomena in a city. An aggregated analysis distributes the start time in times of the day in an attempt to identify phenomena related to a particular time of the day for city improvements. In this question, the day is divided in early morning, morning, afternoon, and evening, and spatial-temporal clusters are analyzed based on the time of the day the movement they represent start. Each time of the day is comprised of roughly six hours, starting from midnight (inclusive) to 6 AM (exclusive), then from 6 AM (inclusive) to noon (exclusive) and so on. The results are presented in Table 4.28. The code in Listing B.31 returns the time of the day and the percentage of cluster evolution paths that start in that time of the day. The code has to calculate a percentage. Therefore, it starts by retrieving the value of `numOfClusterEvolutionPaths`, which will be used as the denominator, and proceeds to distribute cluster evolution paths in one of the four times of the day using the `CASE` clause. Note the definition of each time of the day in the code inside the `CASE` clause. In the end, the results for each individual cluster evolution path are aggregated and percentages calculated.

```
WITH [] AS ListOfClusters

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH
    count(path) AS numOfClusterEvolutionPaths,
    ListOfClusters
```

```
MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]→(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
WITH
    path,
    numOfClusterEvolutionPaths,
    CASE
        WHEN c1.timestamp.hour ⩾ 0  AND c1.timestamp.hour < 6  THEN '
   Early Morning'
        WHEN c1.timestamp.hour ⩾ 6  AND c1.timestamp.hour < 12 THEN '
   Morning'
        WHEN c1.timestamp.hour ⩾ 12 AND c1.timestamp.hour < 18 THEN '
   Afternoon'
        WHEN c1.timestamp.hour ⩾ 18 AND c1.timestamp.hour < 24 THEN '
   Evening'
    END AS timeOfTheDay
WITH
    timeOfTheDay,
    size(collect(path)) AS numOfClusterEvolutionPathsPerTimeOfTheDay,
    numOfClusterEvolutionPaths
RETURN
    timeOfTheDay AS TimeOfTheDay,
    toFloat(numOfClusterEvolutionPathsPerTimeOfTheDay)/toFloat(
   numOfClusterEvolutionPaths) * 100 AS Percentage
```

Listing B.31: The code to return the time of the day and the percentage of cluster evolution paths that start in that time of the day.

What is the largest distance traveled in the movements represented by cluster evolution path formed near each train station?

The distance of the movement represented by a cluster evolution path indicate how far spatial-temporal objects moved together. Analysis on this distance reveals possible important connections between spatial-temporal objects or, in the case of taxis, opportunities for improvements in the public transit system or car-sharing. In this question, cluster evolution paths starting at each train station have their distances calculated, ranked, and the one that traveled the farthest is returned. The results are presented in Table 4.29.

The code in Listing B.32 returns the spatial-temporal cluster whose distance from its start location to its end location is the largest. The `WITH` clause at the start of the code receives the spatial-temporal clusters for each train station. The code proceeds to filter cluster evolution paths that represent movements of more than 500 meters, as described in the section in the thesis, using the restriction `point.distance(c1.location, c2.location) > 500`. Then, the code returns the cluster evolution paths and their distances, but uses the `ORDER BY` clause to sort the results based on the distance and limit the presentation of the rows to the first one using `LIMIT 1`.

```
WITH [] AS ListOfClusters

MATCH path = (c1:Cluster{pStart:True})-[:Relationship*{pMain:True}]->(c2:
   Cluster{pEnd:True})
WHERE
    point.distance(c1.location, c2.location) > 500 AND
    c1.clusterId IN ListOfClusters
RETURN
    apoc.coll.toSet([n in nodes(path) | n.clusterId]) AS
   ClusterEvolutionPath,
    point.distance(c1.location, c2.location) AS Distance
ORDER BY Distance DESC
LIMIT 1
```

Listing B.32: The code to return the spatial-temporal cluster whose distance from its start location to its end location is the largest.

Glossary

clustering The task of dividing data into groups such that data in the same group are more similar than data in different groups. 25

data analysis Refers to the task of using techniques on data to uncover patterns and discover value. 24

spatial relationship Refers to the relationships that objects have with respect to their location in space. 29

spatial-temporal cluster A cluster of spatial-temporal objects that exists for some period of time and moves during existence. 29

spatial-temporal cluster evolution path A path from the occurrence of a cluster in a timestamp to the occurrence of a cluster in a future timestamp. 57

spatial-temporal cluster relationship Patterns in the movement of spatial-temporal clusters that can be identified, analyzed, and processed to reach valuable conclusions. 29

spatial-temporal clustering The task of grouping spatial-temporal objects based on the similarity of their path, that is, the spatial-temporal data produced. 28

spatial-temporal data Data produced by spatial-temporal objects that is described based on location and time. 23

spatial-temporal data analysis Refers to the task of using techniques on spatial-temporal data to uncover patterns and discover value. 24

spatial-temporal object Real-world objects that is associated with a location and a timestamp. Usually spatial-temporal objects move with time. 23

spatial-temporal relationship Refers to relationships that objects have with respect both their location in space and the moment they exist in time. 33

temporal relationship Refers to relationships that objects have with respect to the moment they exist in time. 32