POLITECNICO DI TORINO

Master's Degree in Master Degree in Data Science and Engineering

Master's Degree Thesis

Temporal co-location pattern discovery in spatiotemporal data through parallel computing

Supervisors

Prof. Paolo GARZA

Dott. Luca COLOMBA

Candidate

Gianvito LITURRI

ACADEMIC YEAR 2022/2023

Abstract

This master thesis investigates co-location patterns: categories of entities that frequently appear close to each other. In the co-location pattern mining problem, the categories are called features and each entity that belongs to a specific category is named instance. In the urban field, an example can be "John's Restaurant" which is an instance of the feature "Restaurant". Co-location patterns are important since they reveal underlying correlations between entities. Several algorithms were proposed in literature to tackle the spatial co-location mining task. A step forward in state-of-the-art methods consists in implementing parallel approaches. The idea behind it is to divide the entire dataset into independent partitions that will be processed in parallel. Most of the proposed solutions consist in sequential algorithms. Instead, the few algorithms that use a parallel approach have a common issue: the partitions that should be processed in parallel are not independent and thus require information sharing, creating a bottleneck in the computation.

This thesis explores one of the most recent parallel algorithms in the context of spatial colocation mining, introduced in the paper entitled "Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation". Given that the authors did not disclose their original parallel implementation, the first step in this thesis consisted in the reimplementation of the aforementioned algorithm using PySpark and Python. To verify the correctness of the algorithm, experimental evaluations were performed on a real dataset related to plants of Gaoligong Mountain and were run on the SmartData@PoliTO cluster. The framework is based on two main concepts: the "neighbor-dependency partition" and "Column Calculation". The first one divides the dataset into completely independent partitions. The second one, instead, can finds patterns efficiently. The results achieved in the experimental evaluations are comparable to the results reported in the paper. Moreover, this thesis proposes a novel extensions to such algorithm, adding the time dimension. The first version of the algorithm proposed wants to find spatial-temporal correlation between people and trajectories. The idea behind this is that if two people are friends or have any other relationship, they will have similar positions within the same timeframes. In the implementation we consider for each person a set of trajectories; when we want to find correlation between two people, we verify if they were close to each other in similar timestamps. We verify closeness in terms of both spatial and temporal distance and not exact position sharing. The first version is tested on a large dataset, named Geolife Trajectories, to recognize relationships among people in Beijing. Furthermore, in a second version of the algorithm we extended the analysis by including points of interests (POIs) in the mining process. Doing so, this second version of the algorithm is able to extract

information about people and the places that they frequently attend together. The proposed methodology demonstrates that the implemented algorithm is effective and efficient for discovering spatial-temporal behavioural patterns using parallel computing techniques.

Summary

In the master thesis, we have investigated co-location patterns which represent entities of certain types that appear frequently close to each other and so may be correlated. In the urban field, an example can be the presence of supermarkets and schools in nearby areas. These kinds of insights are very valuable in a lot of contexts, from logistics to medicine. Knowing that two or more categories of shops are frequently close to each other could lead to a better handle on the shipments and therefore save money. One of the most important advantages of co-location patterns is that they can show underlying correlations between different types of data that would be otherwise impossible to identify using typical analysis approaches. This can result in a better understanding of complex systems and the ability to make more informed decisions. In the co-location pattern mining problem, the categories are called features and each entity that belongs to a specific category is named instance. In the urban field, an example can be "John's Restaurant" which is an instance of the feature "Restaurant". In literature, there is a multitude of algorithms that mine co-location patterns. The best one depends on the metrics chosen. Are we interested in a spatial correlation? Understanding how the entities occur inside the patterns could provide useful information and insights to users or system's administrators? Are there frequent, strong association that occur much more frequently than the others? Throughout this thesis, first different state-of-the-art works are analyzed in order to understand their purposes and field of application and especially their strengths and weaknesses. These algorithms are divided into three categories according to the building of the instance table. The instance table is a large and time-consuming table computed for each pattern, which contains all the entities, called instances, that form the pattern. The categorization of spatial colocation mining algorithms is performed by considering the way in which different algorithms handle the construction of such instance table. A step forward in state-of-the-art has been made by designing parallel methods run on powerful hardware such as GPUs and clusters. These technologies speed up the discovery of patterns and allow the scalability and so usage of large real datasets. However, the state-of-the-art algorithms that use parallel computing are stuck because of two problems. The process of finding co-location patterns consists of sequential

steps run one after the other and so, designing a parallel approach in which each partition is completely independent of the others is a tricky task. The second problem is related to the construction of the instance table. The process of building the table, as already mentioned, is an expensive task that could lead to checking and storing all possible combinations of instances that take part in the pattern. If the number of instance categories in the pattern increase, this task will become extremely time-consuming. In the context of this thesis, the paper entitled "Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation" is presented. Such paper tackles the aforementioned problems and demonstrates the effectiveness of the proposed approach in terms of execution time and computational complexity. Based on this work, this thesis proposes a Python re-implementation of the proposed methodology (which was not disclosed by the original authors). All the experimental results were evaluated by running the code on the SmartData@PoliTO cluster of the Politecnico of Torino. The proposed architecture is based on two main parts. The first one is related to how instances are partitioned in order to achieve parallelization. The solution proposed is called "neighbor-dependency partition" and allows parallelization by dividing the spatial dataset into subsets of instances through the prefix strategy. The strategy assigns to each instance a prefix through which completely independent partitions are built. In other words, each defined partition can compute a set of co-location patterns without further information. This solution solves the first problem. The problem of building the instance table, instead, is addressed through a novel algorithm called "Column Calculation". This method allows finding all the instances that take part in the pattern without computing the entire instance table. This algorithm will be used for each pattern that each partition computes. Furthermore, the paper introduces two novel pruning techniques that allow the reduction of the search space before applying the Column Calculation method and therefore furtherly reduce the execution time. Using the two methods introduced in combination with the pruning techniques, competitive results are obtained. The experiments are conducted on a real dataset about plants in Gaoligong Mountain, located in the Szechwan Highlands in the southern part of China. The dataset consists of 11062 spatial instances divided into 10 features. The results obtained through the re-implementation have been compared with the results reported in the paper, demonstrating similar performances. Using the cluster, we are able to mine 160 different patterns in around 15 seconds. Most of these patterns have 2, 3 and 4 sizes. Few co-location patterns have sizes equal to 5 or 6. The second part of this thesis extends the analyzed spatial colocation mining algorithm to the temporal domain. The idea behind this was to adapt the algorithm studied so that it can identify a new kind of correlation: a spatial-temporal one. The new algorithm defines a new neighborhood not just geographically, but also considering the time distance among points. In a nutshell, two entities are close if and only if they

are in the same place in the same time frame. Therefore, the algorithm proposed wants to map the entire dataset into instance pairs in which the instances respect the temporal and spatial constraints. To do this, a parallel approach is developed based on the time dimension. So, the entire dataset is divided into time windows according to the day of the timestamp, and each partition is processed at the same time. The problem with such approach is the presence of pairwise relationships between boundary points (i.e., a point at the end of a time window is a neighbor of a second point at the beginning of the following time window): points where one-day ends and the next begins. To solve this problem, a new partition is defined to process boundary points. Each partition produces a list of pairs of points that respect the temporal and spatial constraint. In conclusion, all these pairs are used as an input of the algorithm re-implemented for the paper to obtain spatiotemporal prevalence patterns. To test our implementation, the Geolife Trajectories dataset is used. This dataset contains 17,621 trajectories with a total distance of about 1.2 million kilometers. Most of the points are located in Beijing (China). The total number of time-stamped points is around 20 million. Due to computational issues and limited resources, one-sixth of the original dataset was considered in this work, with a total number of 3.3M points. The first version of the algorithm aims to recognize relationships among people in Beijing. Furthermore, in a second version of the algorithm, we extended the analysis by including points of interest (POIs), obtained by OpenStreetMap, in the mining process. Doing so, this second version of the algorithm is able to extract information about people and specific places that they frequently attend together. A third version of the algorithm has been implemented that is able to mine patterns related to people and categories of places that they frequently attend together. In conclusion, we are able to process 3.3M points and achieve 1620 spatiotemporal patterns in 2 minutes around. Most of these patterns have 2,3 entities involved. Few patterns have their sizes equal to 4. Overall, the experiments conducted have obtained good results in terms of parallelization degree, execution time and quality of the extracted patterns, considering the size of the input dataset.

Table of Contents

Chapter 1

Introduction

1.1 The importance of the data

In the modern world, data are essential to our ability to comprehend and navigate our surroundings. We now assess and make decisions based on data in a very different way as a result of the growing accessibility of massive datasets and the development of cutting-edge technologies like Artificial Intelligence (AI) and Big Data. For organizations, governments, and people to gain a competitive edge and make wise decisions, the capacity to analyse and extract insights from huge amounts of data has become crucial.

One of the primary benefits of using AI and Big Data is the capacity to identify patterns and insights that would otherwise be impossible to discern. This can lead to more accurate predictions, increased efficiency, and the capacity to make real-time decisions. The usage of these technologies also allows firms to obtain a more in-depth understanding of their clients, allowing them to better adapt their products and services.

Aside from these advantages, the usage of AI and Big Data can have a good impact on society as a whole. Large dataset analysis, for example, can be used to enhance public health outcomes, anticipate and prevent natural disasters, and even aid in the mitigation of climate change consequences.

In general, the value of data in today's world cannot be emphasized. Organizations that want to stay competitive and make educated decisions must be capable of leveraging the potential of AI and Big Data. The advantages of these technologies go beyond the commercial sphere and have the potential to help society as a whole.

1.2 Co-location pattern

One of the thesis's key topics is co-location patterns, which refer to the spatial interaction between different sorts of data points, such as the clustering of specific types of enterprises in a single area or the proximity of certain types of medical facilities to one another. These patterns can be extremely useful in a multitude of sectors, including medical, logistics, and urban planning. In the area of medicine, for example, co-location patterns can be used to identify locations with a high density of specific types of illnesses, which can help public health officials target their resources more effectively. Co-location patterns can be used in logistics to determine ideal distribution hubs for a particular collection of products. Co-location patterns, furthermore, can be utilized in urban planning to identify regions with a high demand for specific types of housing or commercial development. In the 1.1 is shown an example of pattern in the urban context.

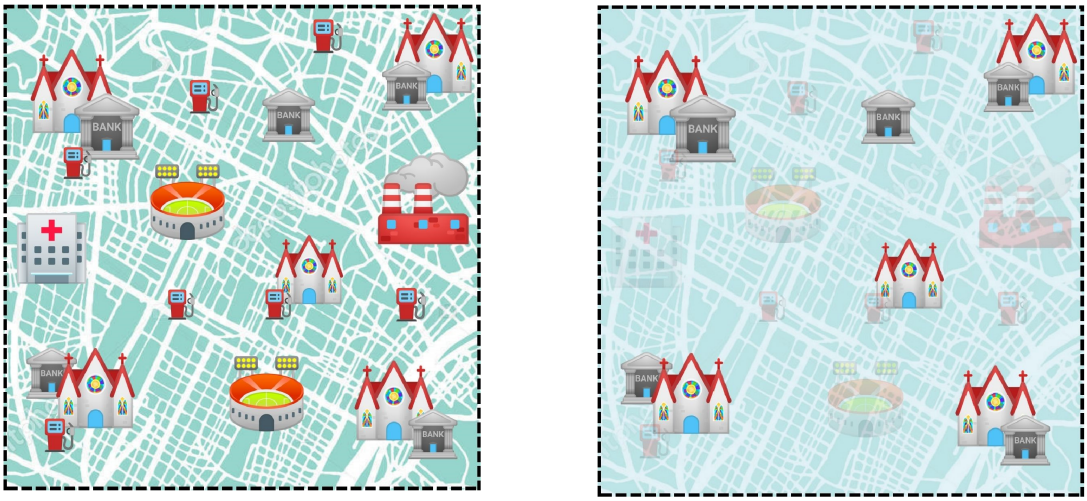


Figure 1.1: Urban co-location pattern example.

One of the most important advantages of co-location patterns is that they can show underlying correlations between different types of data that would be impossible to identify using typical analysis approaches. This can result in a better understanding of complicated systems and the ability to make more informed decisions.

Co-location patterns have been discovered using a variety of existing technologies, such as grid-based algorithms, density-based algorithms. These techniques, however,

have some drawbacks and might not be appropriate for all types of data or circumstances.

Parallel computing is a method that can be helpful in locating co-location patterns. This method enables the simultaneous processing of large amounts of data, which is advantageous when the volume of data is too great for a single machine to handle. The capacity to process data in parallel has become more crucial as a result of the daily increase in data generation.

"Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation" is one study that has been reimplemented. This work describes a parallel computing solution to co-location pattern mining that takes advantage of neighbor-dependency partitioning and column calculation. This technique has various advantages, one of which is its capacity to manage massive amounts of data. Furthermore, in terms of discovering co-location patterns, this study has been shown to be more efficient and successful than standard techniques.

Overall, the capacity to identify co-location trends can provide valuable insights and lead to more informed decision-making in a range of disciplines. Because it allows for the simultaneous analysis of massive volumes of data, parallel computing can be an excellent method for discovering co-location patterns.

1.3 Spatial-temporal correlation

The detection of spatial and temporal relationships between objects in space is the second essential theme of this thesis. This refers to the capacity to recognize patterns and relationships among various forms of data depending on their location as well as the time at which they were captured. This is significant because it provides for a more thorough knowledge of the dynamics of complex systems, which can lead to better decisions.

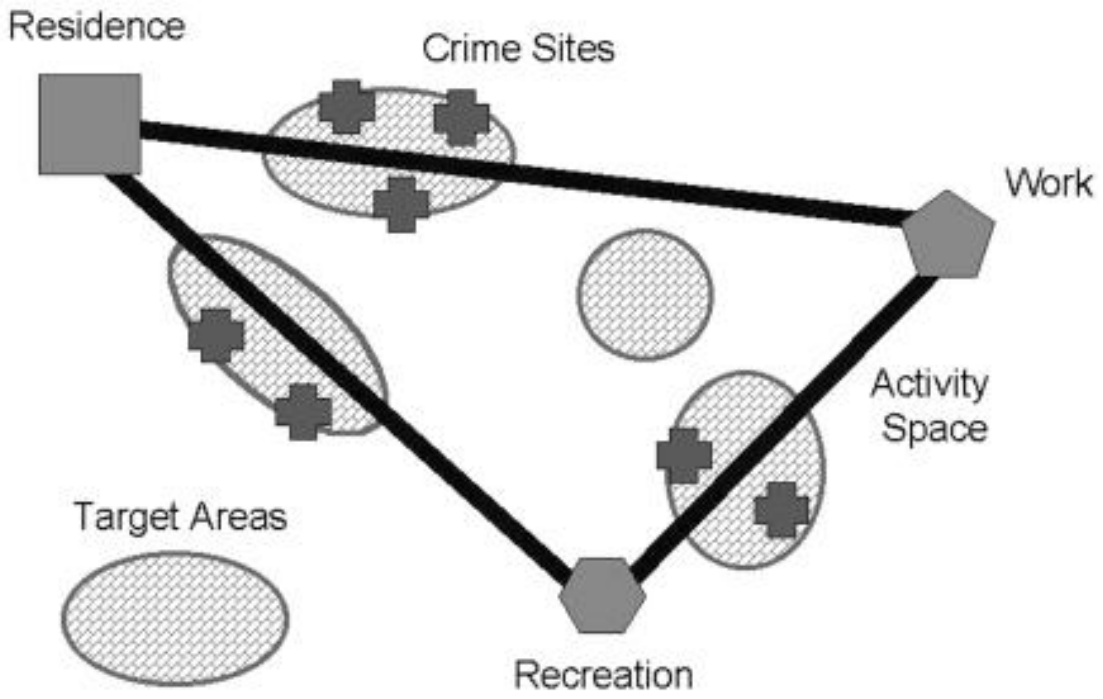


Figure 1.2: Common patterns in crime data.

This form of analysis has a wide range of potential applications in everyday life. It can, for example, be used to find patterns in crime data as we can see in figure 1.2, allowing law enforcement organizations to better target their resources. It can also be used to detect patterns in traffic data, assisting municipal planners in optimizing traffic flow and reducing congestion. It can also be used to detect trends in meteorological data, which can aid meteorologists in forecasting severe weather events and assisting emergency management in planning for them.

Finding geographical and temporal patterns can help us understand people's routines and actions in connection to points of interest and other people. It is possible to get insights about an individual's daily routines, mobility habits, and preferences by studying data on their location and movement over time. For example, by examining data on an individual's position in relation to coffee shops and cafes, it is feasible to deduce their morning routine and coffee preference.

Furthermore, by evaluating data about an individual's location in relation to other individuals, insights regarding their social network and patterns of interaction can be gained. This type of analysis can be used to identify group behavior patterns, such as where people tend to congregate or when certain groups are most active.

It is crucial to stress, however, that this type of study creates serious ethical and privacy problems. The acquisition and analysis of data on an individual's position and movement can be extremely sensitive, and any data collection and

analysis must be done in a way that respects an individual's privacy and rights.To guarantee that data is collected and utilized ethically, it is critical to have clear and transparent policies in place.

In this thesis, the state-of-the-art algorithm for colocation mining [1] was extended to the temporal dimension. in the analysis to find spatial and temporal association. This technique will consider the time relationship between different types of data as well as the spatial relationship. This will enable for a more full understanding of the dynamics of complex systems, which can lead to better decision-making.

1.4 Thesis structure

The structure of the thesis is as follows:

1. In Chapter 2, we will formally discuss the problem of co-location patterns and offer a summary of the previous technologies that have been utilized to identify these patterns. This chapter will give an overview of the available methodologies and strategies for detecting co-location patterns, as well as their strengths and weaknesses.

2. The work [1] will be presented in Chapter 3. This chapter will give a full overview of how this article works, including its methodology and strategies, as well as the outcomes of its reimplementation.

3. In chapter 4, we will explain how the technology given in chapter 3 has been modified to detect spatial and temporal correlations. This chapter will provide the findings of experimental experiments conducted using the proposed strategy, as well as its performance.

4. In the conclusion section, we will summarize the data acquired throughout the thesis and analyze potential future advances of the suggested technology. This will include a discussion of the possible benefits of the suggested method, as well as any limitations or obstacles that remain.

Chapter 2

Spatial co-location pattern mining

Spatial data is undergoing a rapid increase as a result of the development of location-based services. A great deal of important knowledge is hidden in spatial data, therefore it is critical to automatically identify intriguing and previously unknown, but possibly beneficial knowledge from spatial data.

Spatial co-location pattern mining (SCPM), one of the spatial knowledge discovery tasks, seeks to uncover correlations between spatial variables. A co-location pattern (or co-location) is a collection of spatial features whose elements are frequently found in spatial neighborhoods together. Because they are typically positioned near each other, the feature set {Casino, Hotel, ATM} may be a real-world example of co-locations. As it was said, Co-locations can provide crucial insights into a variety of applications. Botanists, for example, might use extracted co-locations to uncover symbiotic interactions between different species, which is beneficial to the conservation of species variety. Other sectors of application include public health, transportation, urban building, and other location-based services, among others.

SCPM is based on Shekhar et al. research [2], which created a distance-based interest metric called the participation index to estimate the prevalence of co-locations. Following that, some extended studies of SCPM are offered to handle various types of data or achieve various mining objectives. The finding of co-locations is a difficult operation since a co-participation location's index depends on all of its row instances (also known as table instances), and recognizing row instances from spatial data is inherently time-consuming.

This chapter is divided into two subsections in which the following concepts are described:

1. the spatial co-location pattern problem, its formulation and an example of

how to find a pattern in a small set of elements

2. the state of the art of technologies used to find spatial co-location patterns with their advantages and drawbacks

2.1 Problem definition

A spatial feature, in a spatial dataset, describes the geographic object type with a Name. In an urban context, geographic object types can be a church, bank, restaurant, etc. A spatial instance of a specific feature, instead, describes a physical object with an identification number and coordinates.

Let:

- $F = \{f_1, f_2, ..f_m\}$ be the set of features
- $O = \{o_1, o_2, ..o_n\}$ be the set of instances belonging to F

An instance is represented by a tuple of three elements:

$$< \ feature\ type,\ id,\ location > \tag{2.1}$$

As it is seen in figure 2.1, it is a small dataset of spatial points in which there are 4 feature types described by an uppercase letter {A,B,C,D}. Each point describes an instance and it is characterized by its feature type, an identification number, and its position within the representation. $A.1$ point is an example of an instance belonging to the A feature type.

Given a distance threshold min_d; if the distance among two instances is less than the distance threshold, it is said that those instances have a neighbor relationship denoted as:

$$R(o_i, o_j) \iff distance(o_i, o_j) \leq min\_d \tag{2.2}$$

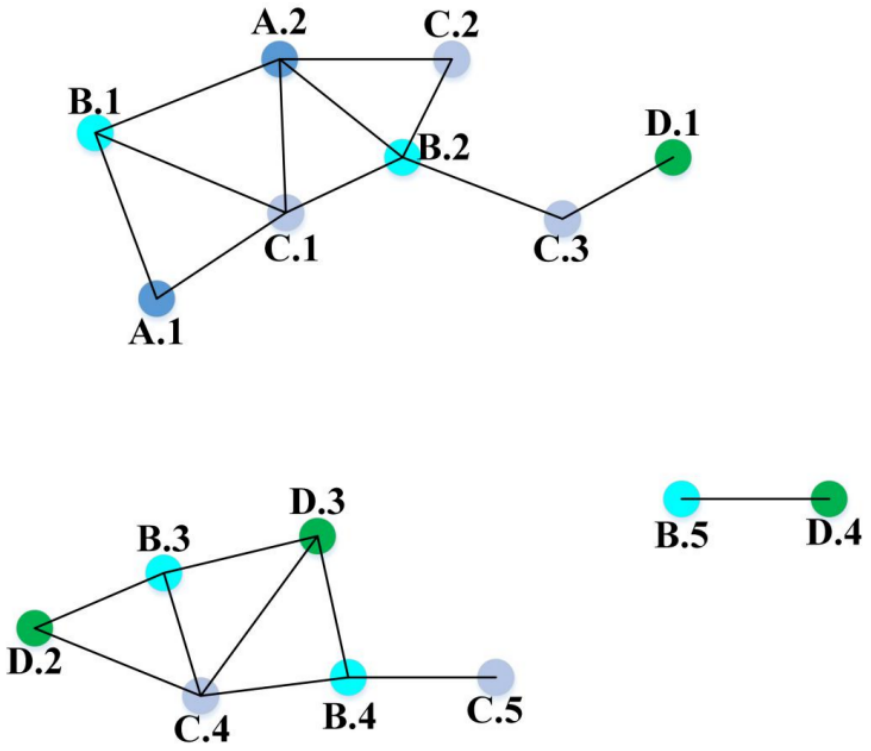


Figure 2.1: Instances example.

As it is possible to see in Figure 2.1, the neighbor relationship is represented by a black line that links two points. A co-location pattern $C = \{f_1, f_2, \ldots, f_k\}$ is a non-empty subset of the feature set $F, C \subseteq F$, whose instances frequently form cliques under the constraint of the neighbor relationship R. For convenience, we arrange C features in ascending lexicographic order and apply this constraint to all future co-locations. The size of C refers to the number of features in co-location C. It is called row instance (RI) or co-location instance, a subset of instances that follow these constraints:

1. RI covers all features of C and there is at most one instance for each feature

2. for each pair of instances in the RI, the neighbor relationship must be satisfied

if these constraints are followed, we call such instances involved in the row instance clique.

	A	B	C
	A.1	B.1	C.1
	A.1	B.1	C.2
	A.1	B.2	C.2
	A.2	B.2	C.2
PR	2/2	2/5	2/5

Table 2.1: Table instance of pattern A,B,C

It is called the table instance of C: $T(C)$, the collection of all row instances of C. The table represents the table instance of the co-location pattern {A,B,C}, An example of row instance is {A.1,B.1,C.1}

To compute the prevalence of a co-location pattern, the interest measure is the Participation Index, defined by Shehkar and Huang [2]. Let f_i be a feature in C, $f_i \in C$, The participation ratio (PR) of f_i in C, denoted as $PR(f_i, C)$, is defined as:

$$PR(f_i, C) = \frac{number\ of\ distinct\ instances\ of\ f_i\ in\ C}{number\ of\ instances\ of\ f_i} \tag{2.3}$$

Written as $PI(C)$, the participation index, instead, is the minimum of the participation ratio of all the features in C:

$$PI = min_{i=1}^{k}\{PR(f_i, C)\} \tag{2.4}$$

Considering pattern $\{A, B, C\}$ in the table 2.1, it is seen that $PR(A, \{A, B, C\}) =$ $2/2$ since all instances of feature A are presented, $PR(B, \{A, B, C\}) = 2/5$, since only two (B.1, B.2) out of five instances of B are included in the table instance of $\{A, B, C\}$. Similarly, $PR(C, \{A, B, C\})$ can be computed.

Set a prevalance threshold called min_prev, A co-location pattern C is prevalent if it follows the rules:

$$PI(C) \leq min\_prev \tag{2.5}$$

An useful feature of the partecipation index is that it satisfies the anti-monotone property, in other words, the partecipation index of a certain pattern C is no bigger than the partecipation indexes of all its subset patterns. It is an useful property since thanks to this one, it is possible to cut the search space and speed up the searching of the spatial co-location pattern.

2.2 Related works

As it is said at the beginning of the chapter, Spatial co-location pattern mining (SCPM) derives from the work of Shekhar et al. [2], who defined the interest measure based, known as the participation index (PI). Based on that work, to analyze different kinds of data or obtain other mining objectives, researchers produced extended studies. The current approaches to find co-location patterns can be categorized as follows:

- The partitioning-based technique, which partitions the space into multiple cells using a grid, then a transaction involving all the objects within the cell is created. Then, it specifies supports based on the created transactions as if they were standard transaction data. Only instances within individual cells are evaluated in this approach, whereas those across cells are ignored since two objects within distance d but across cells boundaries are ignored.

- The construction-based technique, which heuristically builds instances of a given label set and counts the number of produced instances as support. Because of the heuristic nature of this approach, certain instances of a label set may be missed.

- The enumeration-based approach, which counts all row occurrences for a given label set. No instances are lost with this approach, however, the support definition is not anti-monotonic and illogical.

- The participation-based strategy, which captures all potential instances and has a support definition that satisfies the anti-monotonicity characteristic, is the one that is most frequently utilized. The participation-based technique, like the enumeration-based approach, takes into account all potential row instances, but rather than counting each row instance individually, it organizes the row instances into various groups and then counts the groups.

Another challenging problem is the calculation of the table instance, being it a time-consuming task. Three methods to compute such table are:

- full-join approach: relies mainly on computing the join operation between table instances to identify collocation instances. This strategy can produce accurate collocation sets and is comparable to the Apriori method. Due to the growing number of collocations and instances of their tables, the algorithm's scalability on dense and substantial geographical datasets presents challenges.

- The partial-join approach: involves constructing a set of disjoint cliques in spatial instances to identify instances of co-location belonging to a common clique, called intraX instances, and all instances that have at least one cut

neighbor relation called interX instances. Then, intraX and interX instances are merged to compute the PI. This method drastically decreases the number of costly join operations required to discover table instances. However, the performance is determined by the spatial dataset's distribution, specifically the number of cut neighbor relations.

- The join-less method: it compresses the spatial neighbor relationships between instances into a star neighborhood. By scanning the star neighborhood and performing a 3-time filtering process, all feasible table instances for each co-location pattern were generated. The join-less co-location mining approach is efficient because it uses an instance-lookup scheme to discover co-location table instances rather than an expensive spatial or instance join operation.

Researchers have found several methods to speed up the identification of row instances such as Join-based approach [3], Joinless approach [4], iCPI-tree approach [5]. However, because they rely on serial processing and are restricted by the computational capacity of a single computer, most systems are underpowered and inefficient for huge spatial data analysis. Furthermore, some parallel SCPM methods based on MapReduce [6, 7, 8] or GPU [9, 10] have been developed to handle enormous geographical data, but they are still stuck with co-location participation index calculation. In the next subsections, we will present different works related to co-location pattern mining.

2.2.1 Buffer-Based Approach

Ge et al. [11] use a buffer-based model for measuring the spatial relationship of extended objects and mining co-location patterns. The usage of a buffer-based model has the advantage of taking into account complex spatial features like lines, polygons, etc. On the other side, this kind of approach leads an high computational complexity because of the expensive buffer-level overlay operation.

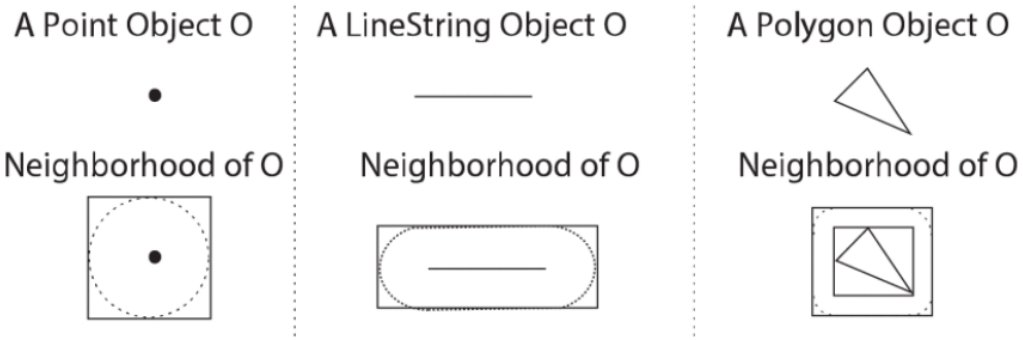


Figure 2.2: minimum object bounding box used in buffer-based approach.

To address this issue, that work [11] presents a coarse-level co-location mining approach based on the filter-and-refine paradigm. Within the framework, a set of rigorous upper bounds based on geometric properties is created. Such upper bounds are used to gradually prune the search space. Furthermore, a join-less schema is adopted to decrease the calculation cost of size-k (k > 2) co-location patterns. Overall, this approach is still not scalable for massive data and cannot be used for mining size-k (k>2) co-location patterns.

2.2.2 Maximal dynamic co-locations

Although there are numerous ways for mining spatial co-location patterns, none of them can detect dynamic correlations between spatial elements. The term dynamic refers to the change over time of the objects in space.

A possible use case, in which dynamic correlations could be useful, is the identification of a polluted area; pollution is a phenomenon that you will observe only if you monitor the same area at different time points. An example is shown in Figure 2.3 in which the same space of a polluted area at two different point in time is represented. The instances in black are destroyed by pollution. Using present approaches, the decrease in the number of instances is not revealed since patterns are computed as a percentage of the common instance over the total ones and so the pattern remains the same.

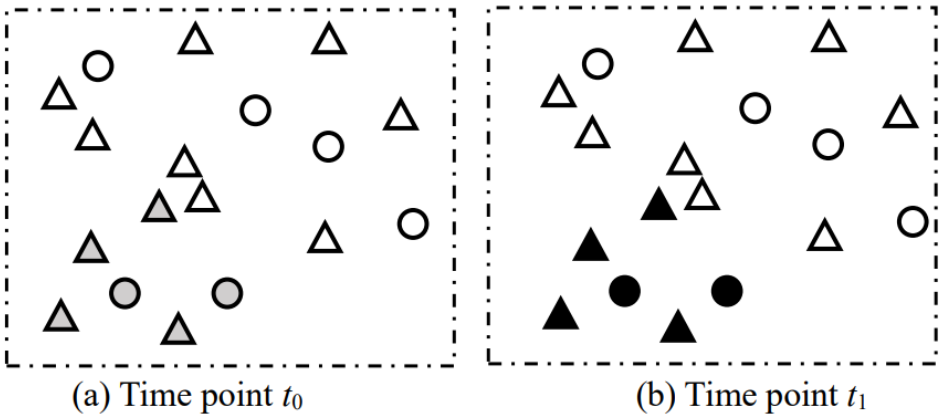


Figure 2.3: Different instances of a polluted area in two different timestamps.

To address these kinds of problems, Yao et al. [12] produced a work in which in which objects in space at different time points are taken into account to find dynamic correlations between spatial elements.

2.2.3 Kernel-density based model

A possible approach used to compute co-location patterns is the density-weighted distance model (DWD). It is a statistical method that uses the distance between objects in a data set to identify areas where these objects are most densely concentrated. When calculating the distance, the density of data in a given area is taken into account, as more densely populated areas tend to have lower average distance values than sparser areas. Once the densest areas of concentration have been identified, more detailed information on the location of the data can be provided. Using this model, you can find the most frequently occurring instances in a space and then define a co-location pattern. Based on this approach, Yao et al. [12] provided a co-location pattern mining approach with a density-weighted distance thresholding consideration. This new technique is known as SGCT-K which focused on efficiency and storage improvements. To begin, a size-2 instance database is built, which contains not only instance pairs that satisfy a distance threshold but also their distance weights, which are determined from the KDE model. The predominant patterns are then determined using a novel verification approach combined with the density-weighted distance thresholding consideration.

2.2.4 Weighted participation index

Based on a previous work, in which the degree of dispersion metric is proposed to quantify the difference in the number of the number of instances for characteristics in a spatial dataset; [13] extends such work, proposing the rare intensity to quantify the rare strength of features in a pattern. The rare intensity takes into account the degree of dispersion in the number of instances for features in the entire spatial dataset as well as that in a co-location at the same time. This work presents a novel interest measure termed the weighted participation index (WPI), which may locate co-locations with or without unusual features; additionally, WPI satisfies a conditional anti-monotone property that can be used to efficiently tune the search space.

2.2.5 Fraction-score measure

Chan et al. [14] proposed a new support measure called Fraction-Score that takes into account all possible row instances while avoiding the over-counting problem that occurs when multiple row instances within a group share the same object (as the participation-based approach does) as well as when multiple row instances across groups share objects. The main idea is to count each group as a fractional unit of prevalence rather than a full one, with the fraction value derived by amortizing an object's contribution across all row occurrences in which the object is involved.

2.2.6 Intra-feature connection

Current SCPM techniques only focus on a pattern's interfeature connection: in a nutshell, what are the categories that frequently appear close? However, it could be useful to know how the instances appear in a certain pattern. This kind of insight is called intrafeature connection and can lead to further knowledge about the pattern.

Figure 2.4: An example of spatial dataset.

An example is shown in the Figure 2.4 in which we can see that in the pattern $\{bank,\ restaurant\}$, we notice there are several restaurants close to a single bank. But there are not necessarily multiple banks appearing in the neighborhood of a restaurant. That phenomenon may show that the presence of banks can encourage the clustering of restaurants, but that banks typically do not occur in groups, which is more advantageous for the layout of the city than the traditional co-locations just taking the association between features into account. So, there are optimum reasons to study these kinds of characteristics of patterns. In fact, [15] provided the intra-coupling congregation association index called IntraCAI to capture the intra-coupling relation per feature in a co-location so as to enrich the semantics of co-locations.

2.2.7 Knowledge-based approach

Patterns are not equal in terms of value. Some patterns could be more interesting than other ones, all depending on the user that will use these information to make decisions. Therefore, the mining techniques should be built on substantial user interaction to find user-preferred co-location patterns. To do this, the knowledge-based approach uses prior knowledge about the properties, relationships, and processes to bind system elements in order to identify the most significant patterns of co-location. This approach uses existing knowledge about biology, geography, physics, etc. to filter the data and identify colocations that are most likely and

relevant based on the specific context. So, [16] provided a work based on the knowledge and ontologies to discover user-preferred co-locations. The work follows the model presented in FIgure 2.5

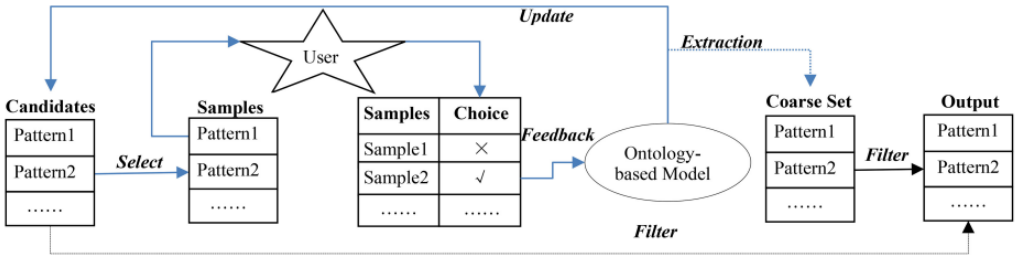


Figure 2.5: Schema to discover user-preferred co-locations.

The domain knowledge is obtained by specialists, which includes the connections between concepts in a particular domain. This knowledge is integrated into the pipeline used to find semantics co-location patterns, which can help users with various requirements.

2.2.8 Join-less approach

Yoo et al. in [4] proposed: two novel neighborhood models called star neighborhood partitioning and clique neighborhood partitioning, and compared the models. The neighborhood partition model (simply, the clique partition model) partitions a spatial data set into sets of objects having clique relationships. The it intruduces a new join-less schema, based on the star neighborhood partitioning, that uses an instance lookup instead of an expensive spatial or instance join operation for filtering co-location instances.

2.2.9 iCPI-tree approach

Wang et al. in [5] introduced a structure called iCPI-tree, useful to materialize neighbor relationships.

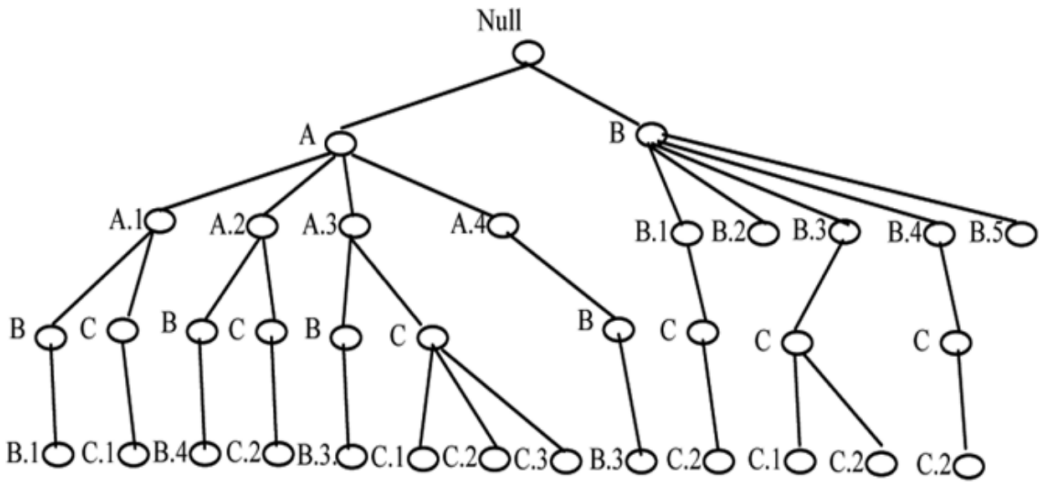


Figure 2.6: An example of iCPI-tree of spatial dataset.

As it is shown in Figure 2.6, in this case, the spatial dataset is represented by a tree structure with features and instances. This particular design choice improves the efficiency of the co-location pattern algorithm, discovering in less time all the co-location patterns and increasesing the scalability, allowing the processing of large amount of data.

2.2.10 Density strategy

Xiao et al. [17] proposed a co-location pattern algorithm based on a density function that divides the space by a grid to process sub-spaces with a sufficient number of instances. The result is that not all instances of a candidate will be identified. This strategy applies a pruning strategy based on an upper bound and the prevalence threshold to understand which sub-spaces to process to identify co-location pattern. This novel strategy reduces a lot the number of joint operations between the table instances of subpatterns.

2.2.11 GPU-based algorithms

A way to improve efficiency and scalability is proposed by [9, 10] through the usage of GPUs. This design choice leads to the possibility to parallelize the execution of different neighbor partitions as long as the algorithm itself is parallelizable and so the building of the patterns is not sequentially. Thanks to the power of the hardware, it is possible to find co-location patterns in a real dataset and scalability is ensured. The two main drawbacks are the higher cost of the GPU and its limited

amount of memory. In conclusion, [10] adds some GPU optimizations including reducing unnecessary memory transfer and using batch memory transfer to replace frequent small memory transfers.

2.2.12 MapReduce-based model

Yoo et al. in [6] proposed a MapReduce-based parallel co-location mining algorithm. MapReduce is a programming paradigm used to perform distributed computing on a large amount of data. MapReduce abstracts away the complexity of working with distributed systems, such as computational parallelization and job allocation, to simplify parallel processing.

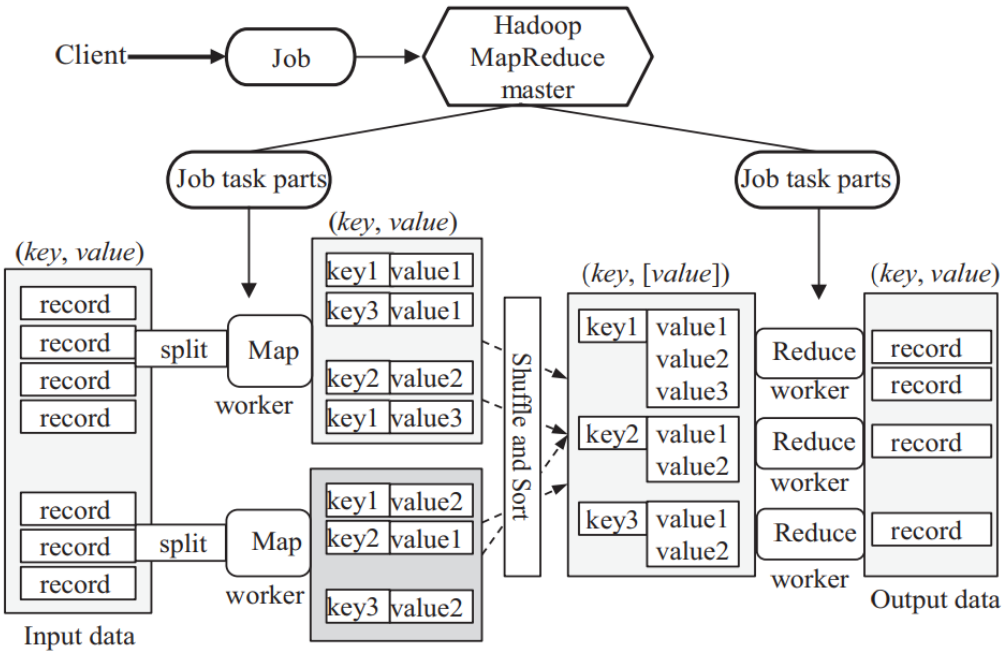


Figure 2.7: Schema of MapReduce program.

The abstraction is shown in the Figure 2.7 and it is based on two main functions: map and reduce. When a job is started, the input data is divided into physical blocks and distributed among cluster nodes. This type of data partitioning and distribution is known as sharding, and each component is referred to as a shard. In each shard, there is a list of key-value pairs that will be mapped by the map function to new key-value pairs. Then, in the shuffle phase, all values with the same key are grouped together and in conclusion, in the reduce step, a function is

applied on the grouped value associated to the same key. The work, proposed by Yoo et al., discovers prevalent co-located event sets through two main tasks:

- Spatial neighborhood partition, which divides the colocation search space.
- Parallel co-located event set search, that searches the instance synchronously by a map worker and then the reduce worker merges instance sets to find co-location pattern.

The general schema of the algorithm is presented in the Figure 2.8.

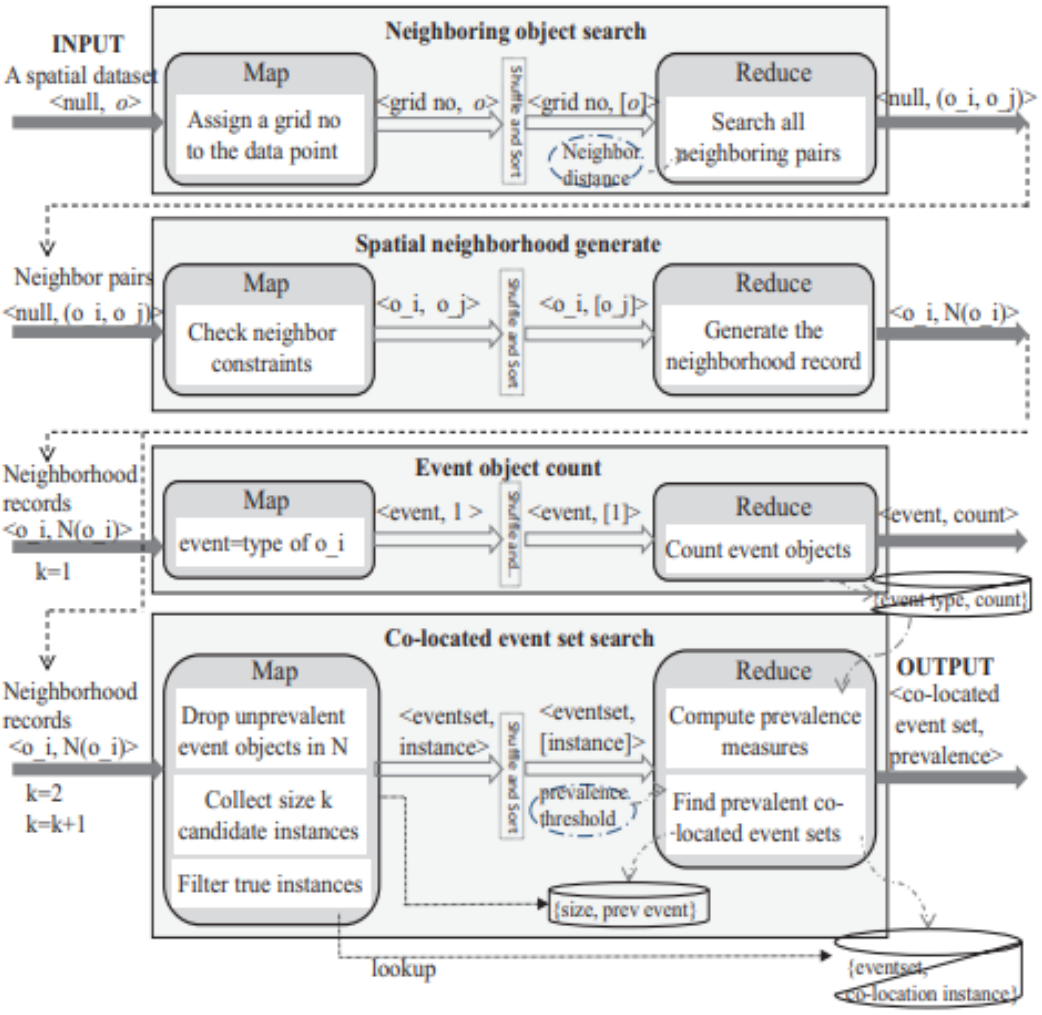


Figure 2.8: MapReduce-based algorithm schema.

Based on this work, J. S. Yoo et al. in [7] introduced a novel method to efficiently search for neighbor relationships and a schema that shares discovered row instances among nodes by HBase, a distributed NoSQL database that runs on Hadoop. This approach is called ParColoc and it is able to process a large amount of data thanks to the power of parallel processing on the cluster. A drawback of this model is that the possible row instances generation and the clique relationship checking are expensive and so, Yang et al. in [8] addressed this problem by proposing a novel MapReduce-based parallel algorithm based on ordered-clique-growth, named PCPM_OC, in contrast to ParColoc. This approach is based on expanding ordered cliques in parallel. The ordered cliques expanding operation is performed fastly because it can re-use the previously processed information, and there is not any non-clique instance generated. In conclusion, these two algorithms, ParColoc and PCPM_OC, have two main problems:

- Complete table instance: it is computed for each pattern to obtain the participation index. For huge datasets, this task is unfeasible.
- The Shuffle phase: since there is poor computing independence, all generated rows must be shared over the network in the shuffle phase and so, for a large amount of data, it will be unfeasible.

To address these problems, a novel parallel SCPM approach is proposed based on neighbor-dependency partition and column calculation, which will be presented in the next chapter.

Chapter 3

Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation

In this chapter, the work that has been re-implemented in this Master thesis is described, namely "Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation", it was written by Yang et al. [1]. In this work, a novel SCPM approach is proposed to find co-location patterns. Compared to previous work based on the MapReduce paradigm, this one solves several problems regarding the independence of the neighbor partitions and the computation of the table instance for each co-location pattern. The principal contributions of this paper are:

- neighbor-dependency partitions: a novel parallelization mechanism that divides the entire spatial dataset into several overlapped subset. It is done by the usage of a prefix strategy that is useful to build independent portion of dataset that will be processed in parallel. This method solves the first problem introduced at the end of the previous chapter.

- Column-Calculation: a new approach to compute the participation ratio unless the presence of the entire table instances. This solution solves the second big problem previously cited and enables the scalability of the algorithm.

- two novel pruning strategies methods based on "Eligible instances": strategies that allow us to understand whether a certain pattern could be prevalent or not, without the need to actually execute the entire process.

3.1 Neighbor-Dependency Partition

In this section, the "neighbor-dependency partition" mechanism is described. Such novel strategy divides the dataset into independent overlapped subsets that will processed at the same time. To better explain the reimplemented methodology, the toy dataset shown in Figure 1.1 will be used to introduce several examples throughout this chapter. The building of the Neighbor-Dependency Partitions is done in three steps:

1. Materialize the neighbor relationship: we want to catch the neighbor relationship information (the black link between points in Figure 1.1) and put them in a table in order to be processed. Given a point o, we define the neighbor set of instances with the following formula:

$$Neigh(o) = \{o' \mid o' \in O,\ R(\ o\ , o'\ ),\ o' \not\equiv o\} \tag{3.1}$$

   Then, the neighbor dataset is defined as the set of all instances and their neighbor sets as it is shown in Table 3.1.

Instance	Neighbor Set	Instance	Neighbor Set
A.1	{B.1, C.1}	C.2	{A.2, B.2}
A.2	{B.1, B.2, C.1, C.2}	C.3	{B.2, D.1}
B.1	{A.1, A.2, C.1}	C.4	{B.3, B.4, D.2, D.3}
B.2	{A.2, C.1, C.2, C.3}	C.5	{B.4}
B.3	{C.4, D.2, D.3}	D.1	{C.3}
B.4	{C.4,C.5,D.3}	D.2	{B.3, C.4}
B.5	{D.4}	D.3	{B.3,B.4,C.4}
C.1	{A.1,A.2,B.1,B.2}	D.4	{B.5}

Table 3.1: Neighbor dataset

Table 3.2: Neighbor dataset with prefix

Instance	Prefix	Neighbor Set
A.1		{B.1, C.1}
A.2		{B.1, B.2, C.1, C.2}
B.1	A	{A.1, A.2, C.1}
B.2	A	{A.2, C.1, C.2, C.3}
B.3		{C.4, D.2, D.3}
B.4		{C.4,C.5,D.3}
B.5		{D.4}
C.1	A,B	{A.1,A.2,B.1,B.2}
C.2	A,B	{A.2, B.2}
C.3	B	{B.2, D.1}
C.4	B	{B.3, B.4, D.2, D.3}
C.5	B	{B.4}
D.1	C	{C.3}
D.2	B,C	{B.3, C.4}
D.3	B,C	{B.3,B.4,C.4}
D.4	B	{B.5}

2. Prefix computation: we want to add a useful string to each instance in order to divide the dataset into independent partitions. Given an instance o_i, whose feature is $o_i.t = f_i$, if there is any instance o_j with f_j as a feature in the neighbor set of o_i and f_j is less than f_i in alphabetical order, so f_j is a prefix feature of the instance o_i. The choice of using the alphabetical order is made since it is used as an alphabetical identifier to distinguish features and sort features of a pattern in ascending lexicographic order.

3. Create the partitions: a rule is defined such that each partition of the dataset is independent. Given the neighbor dataset with prefix D, the neighbor-dependency partition centered on feature f is defined, denoted as D(f), such that it must contain:

   - All instances of feature f and their neighbor set.
   - All instances o_j and their neighbor set $Neigh(o_j)$, that have f in their prefix.

Table 3.3: Neighbor dependency partition of A

Instance	Prefix	Neighbor Set
A.1		{B.1, C.1}
A.2		{B.1, B.2, C.1, C.2}
B.1	A	{A.1, A.2, C.1}
B.2	A	{A.2, C.1, C.2, C.3}
C.1	A,B	{A.1, A.2, B.1, B.2}
C.2	A,B	{A.2, B.2}

Table 3.4: Neighbor dependency partition of B

Instance	Prefix	NeighborSet
B.1	A	{A.1, A.2, C.1}
B.2	A	{A.2, C.1, C.2, C.3}
B.3		{C.4, D.2, D.3}
B.4		{C.4,C.5,D.3}
B.5		{D.4}
C.1	A,B	{A.1, A.2, B.1, B.2}
C.2	A,B	{A.2, B.2}
C.3	B	{B.2, D.1}
C.4	B	{B.3, B.4, D.2, D.3}
C.5	B	{B.4}
D.2	B,C	{B.3, C.4}
D.3	B,C	{B.3, B.4, C.4}
D.4	B	{B.5}

Table 3.5: Neighbor dependency partition of C

Instance	Prefix	Neighbor Set
C.1	A,B	{A.1, A.2, B.1, B.2}
C.2	A,B	{A.2, B.2}
C.3	B	{B.2, D.1}
C.4	B	{B.3, B.4, D.2, D.3}
C.5	B	{B.4}
D.1	C	{C.3}
D.2	B,C	{B.3, C.4}
D.3	B,C	{B.3, B.4, C.4}

Table 3.6: Neighbor dependency partition of D

Instance	Prefix	Neighbor Set
D.1	C	{C.3}
D.2	B,C	{B.3, C.4}
D.3	B,C	{B.3,B.4,C.4}
D.4	B	{B.5}

Tables 3.3, 3.4, 3.5, 3.6 represent neighbor-dependency partitions centered on features A,B,C,D, extracted by the neighbor dataset represented in Table 3.2. As it is shown, these tables are a portion of the original dataset. Given the neighbor-dependency partition centered in f: $D(f)$ and $f_1, f_2...f_n$ are all the features in alphabetical order, $D(f)$ can produce only the patterns that start with the f feature. In example $D(A)$ can find all combinations that start with A: $\{AB, AC, AD, ABC, ABD, ACD, ABCD\}$. In the same way, $D(B)$ can produce the following patterns: $\{BC, BD, BCD\}$. As it can be seen, the greater the feature of the neighbor-dependency partition(in alphabetical order), the less will be the number of possible patterns that can be found. The set of all patterns found by all the neighbor-dependency partitions is the complete set of all possible patterns.

Overall, this novel method proposes a solution to the parallelization problem. Based on the prefix strategy, each neighbor-dependency partition centered in f is an independent partition able to find all patterns starting with f. On every partition, the SCPM algorithm is run without the need of communication between different partitions. This characteristic allows the system to scale in the case of large amount of data and so reduces the time needed.

3.2 Column Calculation

Being it time-consuming, the building of the table instance is the second big problem to solve. Useful to compute the participation ratio, the instance table for the algorithms considered in the Related Work chapter was built for each pattern. Column Calculation is the method used to find the participation ratio without building the entire table instance. Considering the table instance shown in Table 2.1, it is seen that there are two rows that describe all possible instances in the pattern: $\{A.1, B.1, B.1\}, \{A.2, B.2, B.2\}$. These rows contain all the information necessary to compute the participation index. The Column Calculation algorithm was developed based on this observation. This approach puts a step forward in the state-of-the-art methods to find co-location patterns. In this subsection the Column Calculation algorithm is described.

Given a co-location pattern C and feature f belonging to C, if there is an instance o of feature f that is in at least a row instance, o is called participation instance of f in C and it is denoted as $PIns(f, C)$. Given the table instance (Table 2.1), $A.1$ is a participation instance of A in the pattern $\{A, B, C\}$ since $A.1$ is present in the row instance $\{A.1, B.1, C.1\}$. Now the participation index can be written in this way:

$$PR(f, C) = \frac{|PIns(f, C)|}{|N(f)|} \tag{3.2}$$

As seen in the formula 3.2, the participation ratio of a certain feature f and a specific pattern C is determined by the number of all participating instances of f in C over the total number of instances of the same feature f. Then, the participation index can be computed by the minimum of all participation ratios over all the features of the co-location pattern. Said that, the problem is how to find efficiently the participating instances for each feature. To do this, we can leverage on an important property of subpatterns. Given a size-k (k>2) co-location C' and its size- $(k + 1)$ super pattern C, $C' \subset C$, then:

$$PIns(f, C) \subseteq PIns(f, C') \quad if\ f \in C, f \in C' \tag{3.3}$$

This formula means that given two patterns such that the first is a subset of the other, for example:

- $\{A, B, C\}$ size=3
- $\{A, B\}$ size=2

the participating instances of A in the pattern $\{A, B, C\}$, ($PIns(A, \{A, B, C\})$ will be at most the participating instances of A in the pattern of size k-1 , $\{A, B\}$, but no more. The same thing is valid for the participating instances of the B feature.

$$PIns(A, \{A, B, C\}) \subseteq PIns(A, \{A, B\}) \tag{3.4}$$

$$PIns(B, \{A, B, C\}) \subseteq PIns(B, \{A, B\}) \tag{3.5}$$

This property can be seen like an upper bound on the number of participating instances of a feature in the colocation pattern C. The next step is to define possible candidates that could be participating instances. The candidate participating instance set of feature f in a size-(k+1) (k > 2) co-location C, denoted as $CPIns(f, C)$, is defined as:

$$CPIns(f, C) = \bigcap_{C' \in C_k^f} PIns(f, C') \tag{3.6}$$

Where C_k^f is the set of all size-k sub-patterns of C containing feature f. Example, given PIns(A,{A,B}) PIns(A,{A,C}), the candidate participating instance set of feature A in pattern $\{A, B, C\}$ will be:

$$CPIns(A, \{A, B, C\}) = PIns(A, \{A, B\}) \bigcap PIns(A, \{A, C\}) \tag{3.7}$$

In other words, the candidate instances of feature f of a certain pattern C are done by the intersection of participating instances of the same feature f of all possible subpatterns of C that contain f.

In this way, candidate instances are the starting point to find the participating instances. Once the candidates are computed, the aim is to verify if o, belonging to CPIns(f,C), takes part in at least one row instance of the instance table of the pattern C. If o takes part in a row instance, it means that o will have a neighbor relationship with other instances in the row and so these other instances will be in the neighborhood of o. Then, Neigh(o) is taken and the instances are divided into small disjoint groups based on their features by the following formula:

$$groupN(o, f) = \{\ o' \mid o' \in Neigh(o),\ o'.t = f\ \} \tag{3.8}$$

where $o'.t$ is the feature type of instance o'. Example:

$$groupN(A.2, B) = \{B.1, B.2\} \tag{3.9}$$

$$groupN(A.2, C) = \{C.1, C.2\} \tag{3.10}$$

To find the row instance which belongs to o, the best way is to compute the cartesian product between the grouped neighbor sets of o and then test the instance combination to find a combination S forming a clique. Example:

$$\begin{gathered} groupN(A.2, B) \times groupN(A.2, C) = \\ \{B.1, C.1\}, \{B.1, C.2\}, \{B.2, C.1\}, \{B.2, C.2\} \end{gathered} \tag{3.11}$$

$\{B.1, C.1\}$ are a clique since C.1 is in the neighborhood of B.1 and so $\{A.2,\ B.1,\ C.1\}$ is a row instance of the pattern A,B,C. Said that, A.2 will be a participating instance of the set$PIns(A, \{A,\ B,\ C\})$, B.1 of the set $PIns(B, \{A,\ B,\ C\})$ and C.1 of the set $PIns(B, \{A,\ B,\ C\})$. This process should be done for each instance of each feature of the co-location pattern. However, this search space is still too large and it will be furtherly narrowed down further. Before applying the cartesian product between grouped neighbors, these ones can be further filtered by the usage of the participation instances when they are known, otherwise by the candidate instances as in the following formula:

$$RSSI(o, f) = \begin{cases} groupN(o, f) \bigcap PIns(o, f),\ if\ PIns(o, f)\ known \\ groupN(o, f) \bigcap CPIns(o, f),\ otherwise \end{cases} \tag{3.12}$$

in which C is a co-location pattern, o is an instance with feature $o.t$ that is different from f. The new instance set is called the row instance search space of instance o on feature f and it is denoted as $RISS(o, f)$.

Thanks to this approach the search space is reduced from groupN(o,f) to RSSI(o,f) and so it reduces the number of combinations obtained by the cartesian product and consequently the number clique relationship to check. The algorithm uses a heuristic strategy to reduce the time taken to find the row instances. In addition to confirming that instance o is a participating instance of feature f in C, the discovery of a row instance RI of pattern C containing instance o ($o.t = f$) also establishes that the other instances contained in RI are likewise participating instances of other features in C. To meet the goal of searching for a row instance to verify many candidate participating instances, it is expected that the searched row instance RI for verifying instance o also contains as many unconfirmed instances of other features as possible. In light of this, we first sort all $RISS(o, f')$ on other features, $f' \not\equiv f$, and place the unconfirmed instances (i.e., instances that have not been included in $PIns(f, C)$in the front, so that they can be searched preferentially. To summarize, the Column calculation method starts from the candidate instances obtained by sub-patterns and wants to check them. The checking is based on the cartesian product between feature groups of the neighbor of the candidate to check. Each feature group is filtered through the candidate of the same feature and therefore the search space is further reduced. Every combination obtained by the cartesian product, is tested until a row instance is found. This solution allows the algorithm to not create the entire instance table for each pattern.

3.3 Pruning strategies

The pruning strategies proposed are based on the observation that the neighbor-dependency partitions centered on a specific feature f gives us some useful knowledge to understand which instances are able to expand themselves and potentially become participating instances. To explain the two pruning strategies adopted, we first introduce the concept of Eligible instances of a specific feature f' respect to a Neighbor-Dependency Partition $D(f)$, denoted as

$$EIns(f', D(f)) \tag{3.13}$$

The definition provided by the paper [1] is:
"Given a neighbor-dependency partition $D(f)$, for any feature $f' \in F$ and instance o with feature f' in $D(f)$, if there is instance o' in the neighbor set of o and the feature type of o' greater than f' in alphabetical order, i.e., $o' \in Neigh(o)$ and $o'.t > f'$, o is an expandable instance of feature f'.

The set of all expandable instances of feature f' in $D(f)$ is represented as $EIns(f', D(f))$". An example can be $C.3$ present in the neighbor-dependency partition $D(B)$ shown in table 3.4.

Observing the neighborhood of $C.3$, we observe object $D.1$, an instance of feature D, which is alphabetically greater than feature $C.3$ itself. Thus, $C.3$ is an expandable instance of C in $D(B)$. This means that $C.3$ can form a clique and could be a participating instance. Next, the expandable rate of f' in $D(f)$ and is defined as follows:

$$p(f', D(f))) = \frac{|EIns(f', D(f))|}{|N(f')|} \tag{3.14}$$

The expandable rate represents an upper bound of the participation ratio, computed using the eligible instances. Given a prevalence threshold min_prev, The first pruning strategy is based on the following rule:

$$p(f, D(f)) < min\_prev \tag{3.15}$$

In a specific neighbor-dependency partition D(f), if the expandable rate of f is less than the prevalence threshold, all patterns that start with f will be non-prevalent. The second pruning strategy is based on the following rule:

$$p(f_i, D(f_1)) < min\_prev \tag{3.16}$$

Given a size-k (k $>$ 2) co-location $C = \{f_1, f_2, \ldots, f_k\}$, if there is at least a feature f_i with 1$<i<$k that has its corresponding expandable rate less than the prevalence threshold, the pattern C will be non-prevalent. Overall, these two pruning techniques are based on the concept of Eligible instances, instances that are able to create cliques. The number of all Eligible instances for a specific feature f with respect to the total number of instances f give us helpful information to prune the search. In fact, the expandable rate of f allow the algorithm to cut the search without applying the column calculation method on a useless pattern. These two techniques avoid waste of computational time.

3.4 MapReduce ALGORITHM

This section presents two MapReduce-based algorithms to compute co-location patterns based on Neighbor-Dependency Partition and Column Calculation. As introduced in Chapter 2 2, mapReduce is a programming paradigm for processing big data sets with a parallel, distributed algorithm on a cluster. It is based on two main functions called Map and Reduce in which, in the map phase, data are mapped into <key,value> pairs, and then in the reduce step all the pair with the same key are processed together to obtain a new <key, value> pair.

3.4.1 Requirements

The MapReduce algorithm for spatial colocation mining is named PCPM-NDPCC. Before introducing it, the following information must be computed:

1. for each feature, the number of instances must be computed. They are necessary to evaluate the participation ratio and apply the aforementioned pruning strategies.

2. the set of all instance pairs satisfying the neighbor relationship.

The second requirement is a big problem when the dataset is very large since it may require the computation of all pairwise distances. In the re-implementation of the algorithm, proper data structures were adopted to reduce the computational cost for neighborhood evaluation. More specifically, the BallTree algorithm was used.

This method is used for spatial division of data points and their allocation into certain regions. Ball Tree Algorithm can be considered a metric tree. Metric trees group and arrange data points based on the metric space they are situated in. The points don't have to be in vectors or have limited dimensions when using metrics. The algorithm divides the data into two clusters. Each cluster is surrounded by a circle (2D) or sphere (3D). The sphere is also known as a hypersphere. [18]

The name Ball tree algorithm is derived from the cluster's sphere form. Each cluster represents a tree node. How the algorithm works is based on these main steps:

1. the centroid of the whole space is computed.

2. Get the point p at the maximum distance from the centroid. It will be the center of the first cluster and child node.

3. the point farthest from the point p (center of the first cluster) will be the center of the second cluster.

4. the others data points are allocated to the nearest cluster.

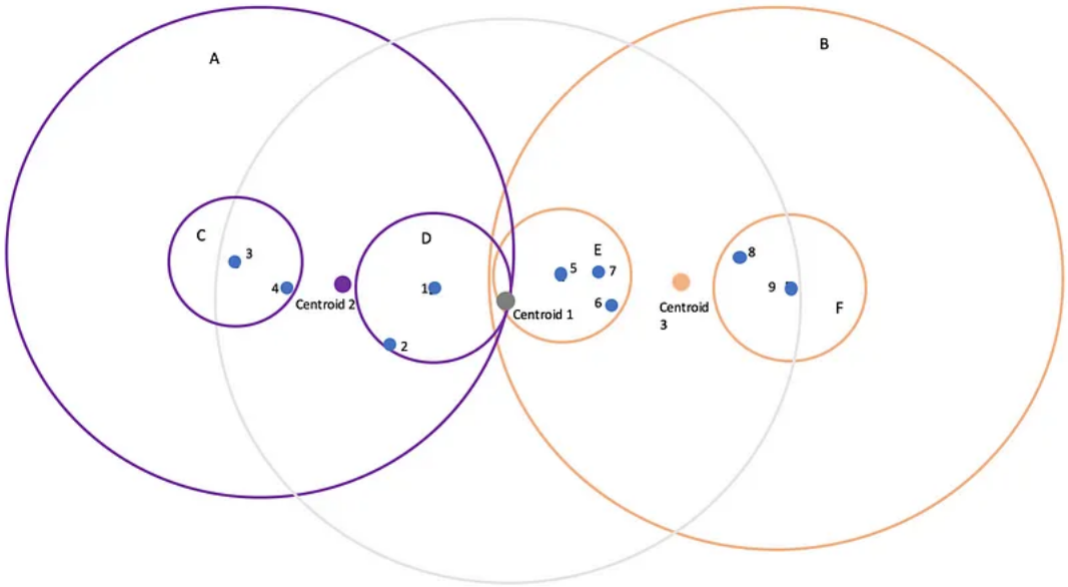


Figure 3.1: Representation of the BallTree algorithm [19].

Within each cluster, the procedure of separating the data points into two clusters/spheres is continued until a predetermined depth is attained. This results is a nested cluster with more and more circles as it is shown in the Figure 3.1.

3.4.2 Implementation

After this brief explanation of the Ball Tree algorithm, it is started to shown how the PCPM-NDPCC algorithm work. Its aim is to build the neighbor database with prefix. Given all instance pairs $< o_i, o_j >$ satisfying the neighbor constraint and such that $o_i.t < o_j.t$, the algorithm can be divided into three main phases:

1. map method: processes the input and returns two pairs: $< o_i, o_j >, < o_j, o_i >$.

2. reduce phase: all pairs with the same key are collected together and we obtain for each instance o_i its neighborhood.

3. reduce method: given the pairs: $< o_i, Neigh(o) >$, this method iterates over the neighborhood and checks if there are instances o_j that have their feature $o_j.t$

alphabetically lower than the o feature $o.t$. In the positive case, $o_j.t$ is added in the prefix. The output pair will have this form: $\langle o_i, (prefixF(o_i), Neigh(o_i))\rangle$

The entire algorithm is written as pseudocode in Algorithm 1.

Algorithm 1 Neighbor database generation

```
1: procedure MAP(key = o_i, value = o_j)
2:     ▷ emit(o_i, o_j), emit(o_j, o_i)
3: end procedure
4: procedure REDUCE(key = o, value = Neigh(o))
5:     ▷ prefixF(o) = []
6:     for each x ∈ Neigh(o) do
7:         if x.t < o.t then
8:             ▷ append x.t to prefixF(o)
9:         end if
10:    end for
11:    ▷ emit(o, ⟨prefixF(o), Neigh(o)⟩)
12: end procedure
```

The second MapReduce algorithm analyzes the produced output. The following method aims to build each neighbor-dependency partition and then apply the Column Calculation method to possible patterns. The algorithm is based on two phases:

1. map: given the input $\langle o, (prefixF(o), Neigh(o))\rangle$, firstly, $\langle o.t, (o, Neigh(o))\rangle$ is emitted. Secondly, for each feature f in $prefixF(o)$ $\langle f, (o, Neigh(o))\rangle$ is emitted. In this way the same pair $< o, Neigh(o) >$ will be sent to different neighbor dependency partitions to find patterns.

2. reduce: the input of this step is the pair $< f, D(f) >$. Each partition will produce all possible patterns that start alphabetically with f. The searching process of co-location patterns uses an Apriori-like bottom-up search and so it starts from patterns of dimension 2 to build patterns of dimension 3 and so on. Therefore, each pattern of dimension k gives an upper bound to build patterns of dimension k+1.

Algorithm 2 Partition and co-location search.

1: **procedure** MAP($key = o, value = \langle prefixF(o), Neigh(o)\rangle$)
2: ▷ $emit(o.t, \langle o, Neigh(o)\rangle)$
3: **for each** $f \in prefixF(o)$ **do**
4: ▷ $emit(f, \langle o, Neigh(o)\rangle)$
5: **end for**
6: **end procedure**
7: **procedure** REDUCE($key = f, value = D(f)$)
8: ▷ $P_1 = \{f\}$
9: ▷ $k = 2$
10: ▷ $P = []$
11: **while** $P_{k-1} \not\equiv \emptyset$ **do**
12: ▷ $C_k = gen\_candidate\_pattern(P_{k-1}, k)$
13: ▷ $filter\_candidate\_pattern(C_k, P_{k-1}, D(f))$
14: ▷ $PIns = search\_participating\_instance(C_k, D(f))$
15: ▷ $calculate\_PI(C_k, PIns)$
16: ▷ $Pk = select\_prevalent\_pattern(C_k)$
17: ▷ $P = P \cup P_k$
18: ▷ $k = k + 1$
19: **end while**
20: ▷ $emit(f, P)$
21: **end procedure**

After the first iteration of Algorithm 2 (reduce phase), all patterns of dimension 2 are computed, All the following iterations starts with patterns of size k and produce as output patterns of size k + 1. In the first iteration, generation of patterns of size 2 is done by concatenating the feature f and the others features in the partition. Example: given f=A and others features are: $\{B, C, D\}$, the candidate patterns, at the first iterations will be $\{AB, AC, AD\}$. From the second iteration onwards, the candidate patterns are generated by the join operation between frequent patterns. Example: confirmed patterns of dimension 2 $\{AB, AC, AD\}$ the candidate patterns of dimension 3 will be $\{ABC, ABD, ACD\}$. As the next step, the candidate patterns are filtered through the two pruning techniques previously explained. Then, for each candidate pattern, the Column Calculation technique is applied. Once the Participating instances for each feature of the pattern are obtained, the Participation Ratio is calculated. Then, Participation Ratio is compared with the prevalence threshold set at the beginning of the pipeline. If the participation ratio is grater than the threshold, the pattern of dimension k will be prevalent and will be used in the next iteration to build candidate

patterns of dimension k+1. The validated pattern is added to the set of all prevalent patterns. In conclusion, when the loop is finished, the output of the algorithm will be the pair $< f, P >$ in which P is the set of all prevalent patterns of any dimension that start with f.

To conclude this subsection, a complexity analysis is performed/given for each individual step to better comprehend the computation complexity of each individual step of the algorithm. Such analysis is fundamental to understand the change in execution time as the data volume grows. For each step of the algorithm its complexity is shown in Table 3.7.

Step	Complexity
Map Algorithm 1	O(|R|)
Reduce Algorithm 1	O(|O|*max(|Neigh(o)|))
Map Algorithm 2	O(|O|* max(|PrefixF(o))
Reduce Algorithm 2	$O(maxK * \|C_k\| * k * n_1 * n_2^{k-1})$

Table 3.7: Complexity analysis.

In the first step. the algorithm checks all pairwise neighbor relationship. $|R|$ is the number of instance pairs and so the complexity is $O(|R|)$. In the second one, given the number of the instances $|O|$ and the number of instances in the neighborhood $|Neigh(o)|$, the algorithm scans for each instance and its neighborhood and so, the complexity will be $O(|O| * max(|Neigh(o)|))$ In the map method of the second algorithm, given the number of instances $|O|$ and the number of prefix of instance o $|PrefixF(o)|$, the method iterates all the instances and for each of them their prefixes. In the last step, the candidate generation step with the pruning methodologies is considered. The candidate generation process starts with patterns of size 1 and is iteratively repeated k-1 times to obtain patterns of size up to k. At each step and for each pattern, the method checks all candidate instances n_1 for all features k and for each candidate instance, all combination of the remaining features n_2^{k-1} are tested, in which n_2 is maximum length of $CPIns(f, C)$. In conclusion the overall complexity is $O(maxK * |C_k| * k * n_1 * n_2^{k-1})$. It can be noted that the entire algorithm is performed in parallel and two pruning techniques are applied and therefore the actual complexity is lower than the worst case described in the complexity analysis.

3.5 Experimental Evaluation

This section presents the experimental setup and the obtained results of the reimplemented algorithm.

3.5.1 Setup

The experiments are conducted on a real dataset about plants in Gaoligong Mountain, located in the Szechwan Highlands in the southern part of China. The dataset consists in 11062 spatial instances divided into 10 features. The names of the plant categories are written in a foreign language and so they are mapped through capital letters:$\{B, J, K, L, M, O, P, V, W, X, Y\}$. The distribution of the dataset is reported in: 3.2.

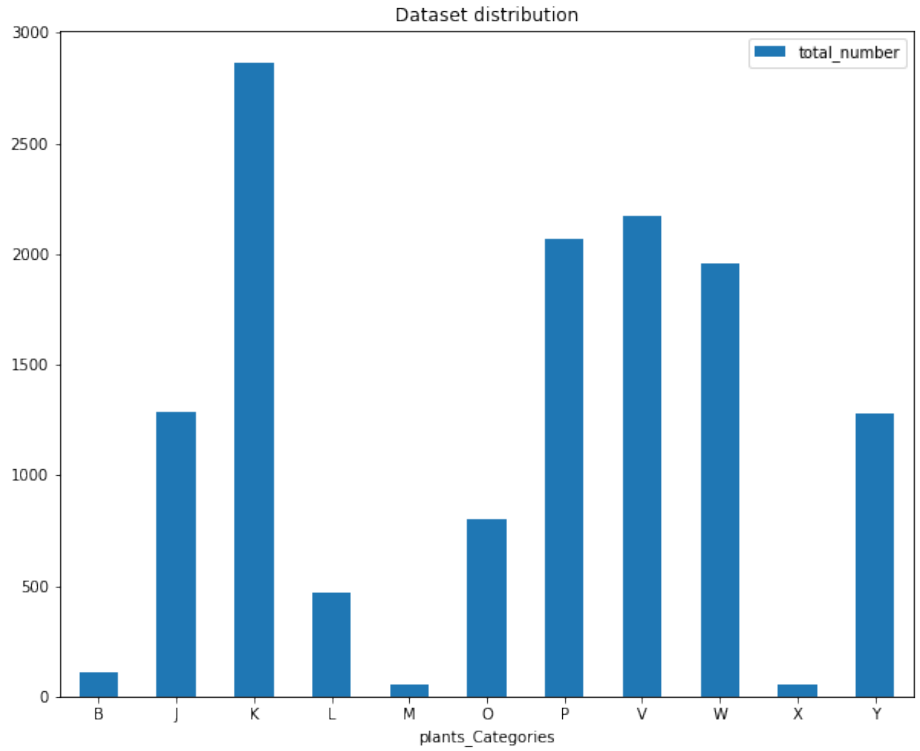


Figure 3.2: number of plants for each category.

As seen in the distribution plot the category with the maximum number of plants is the K category with 2866 plants. Instead the X is the category with

the minimum number of plants that are 53. The average number of plants per categories is 1194. The previously described algorithm is implemented in Python, using the PySpark library and run on the SmartData@PoliTO cluster. This cluster is composed of 36 nodes with a maximum capacity of 220 TB of data (around 650 TB of raw disk space).

3.5.2 Execution time performance

In this subsection, the performance of the algorithm are evaluated and the experiments presented in the original paper are conducted. A comparison between the official implementation and the version developed in the context of this thesis is done. The first experiment wants to investigate the growth of the execution time as the input distance threshold varies. As a reminder, increasing such thresholds increases the number of pairwise neighbor relationships between objects. The experiment is conducted by setting the prevalence threshold to 0.1 and varying the distance threshold. The thresholds chosen are 3, 3.5, 4, 4.5, 5 km. The behavior is shown in Figure 3.3. In the chart can be seen that there are two different sub-regions in the graph: before and after 4 km.

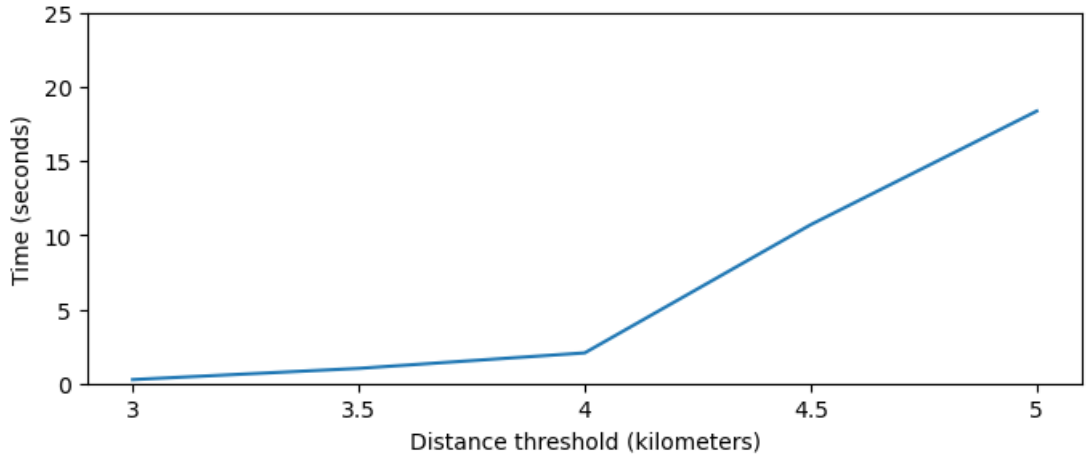


Figure 3.3: Execution time as distance threshold increases.

The behaviour of the function from 3 to 4 km is different with respect to from 4 to 5 km. It is possible since larger the distance threshold, larger the size of the patterns and so the time to check a specific pattern increases.

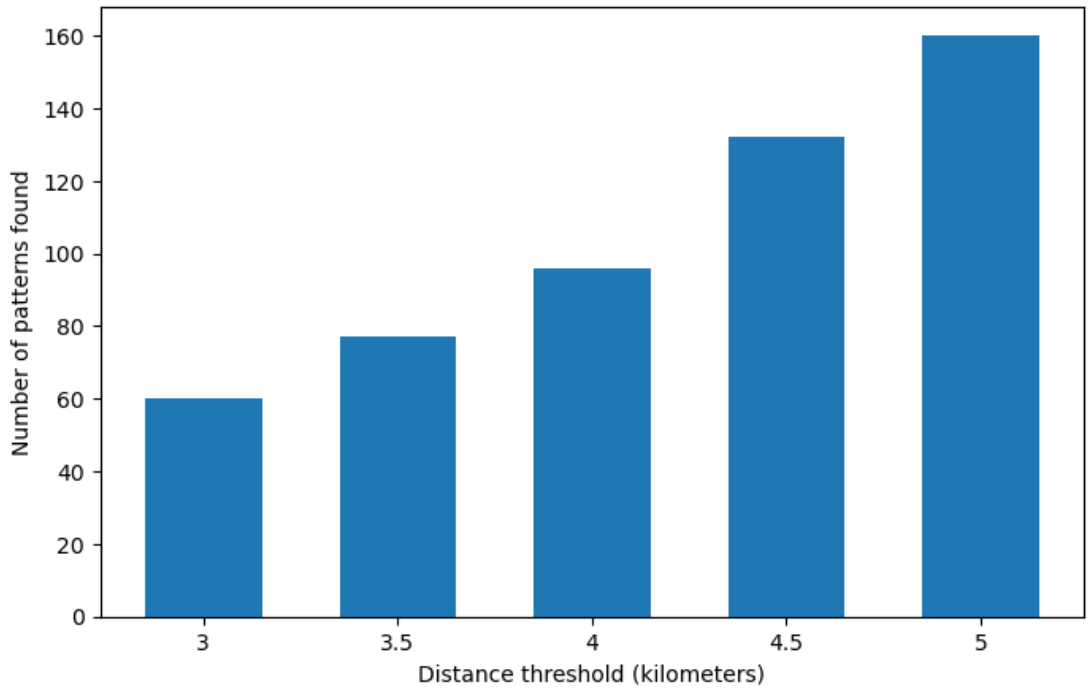


Figure 3.4: Number of mined colocation patterns for each considered distance threshold.

The better quantify the growing complexity of the problem, Figure 3.4 shows the number of extracted co-location patterns by varying the distance threshold.

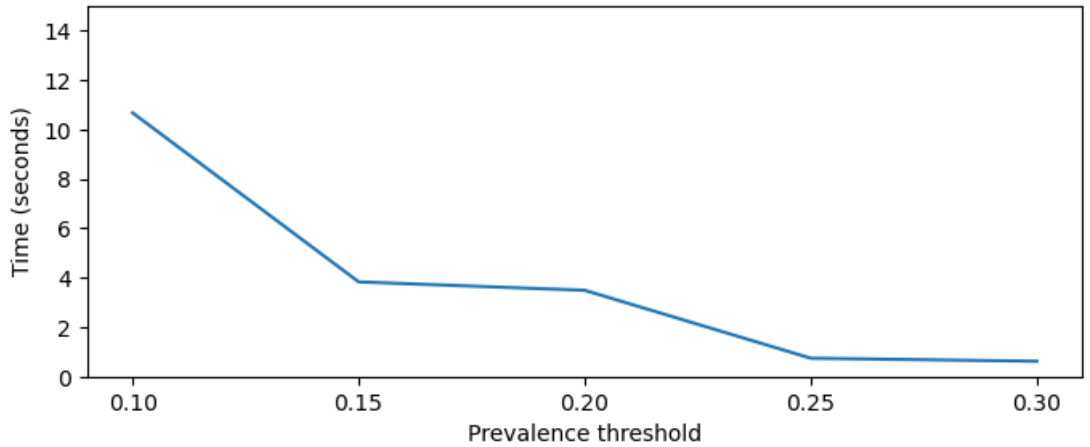


Figure 3.5: The time spent according to the prevalence threshold.

Looking at the two plots, it is seen that in the case that will call "a" in which

the distance threshold is equal to 5 km, the number of patterns founds is almost 160. Considering the the ratio: number of patterns over the the execution time, 8 patterns per second are found.

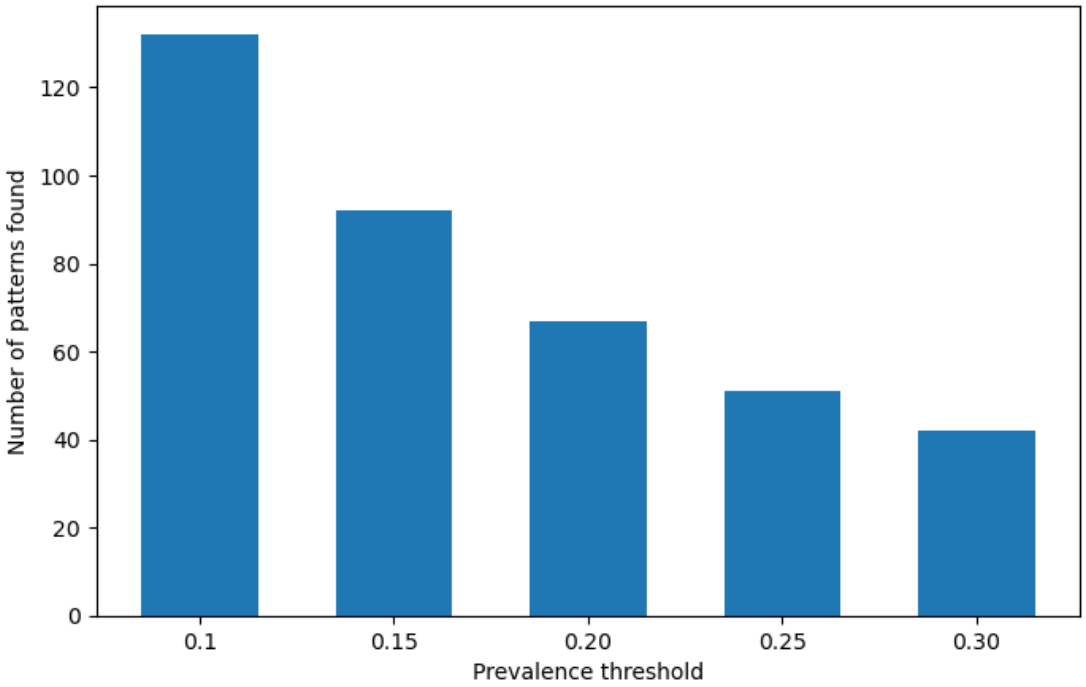


Figure 3.6: Number of patterns found according to the prevalence threshold.

In a second experiment we evaluate the execution time as the prevalence threshold varies. The setup of this experiment is based on a fixed value of distance threshold equal to 4.5 km and 5 possible values of prevalence threshold: 0.1, 0.15, 0.2, 0.25, 0.3. As a reminder, prevalence threshold determines whether a pattern is prevalent or not. The higher the prevalence threshold, the lower the number of prevalent patterns since patterns with a low participation ratio will be removed. In fact, it is seen the trend of the function 3.3 is negative. Furthermore it can be noted that the big drop down is obtained from 0.1 to 0.15 of prevalence threshold in which the execution time decreases of 65% from 11 seconds to 4 seconds around.

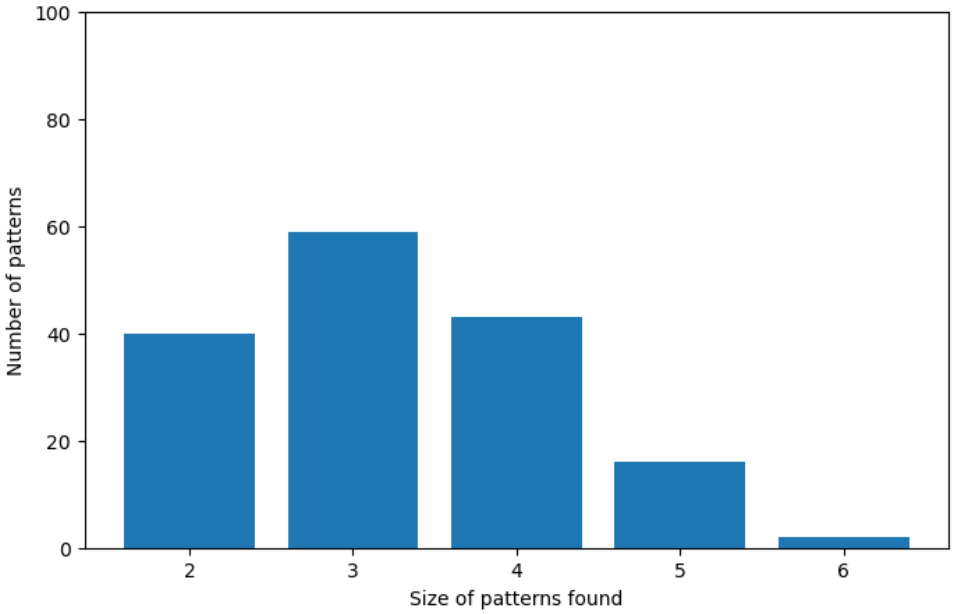


Figure 3.7: Pattern distribution according to the size of the patterns found in the case "a" .

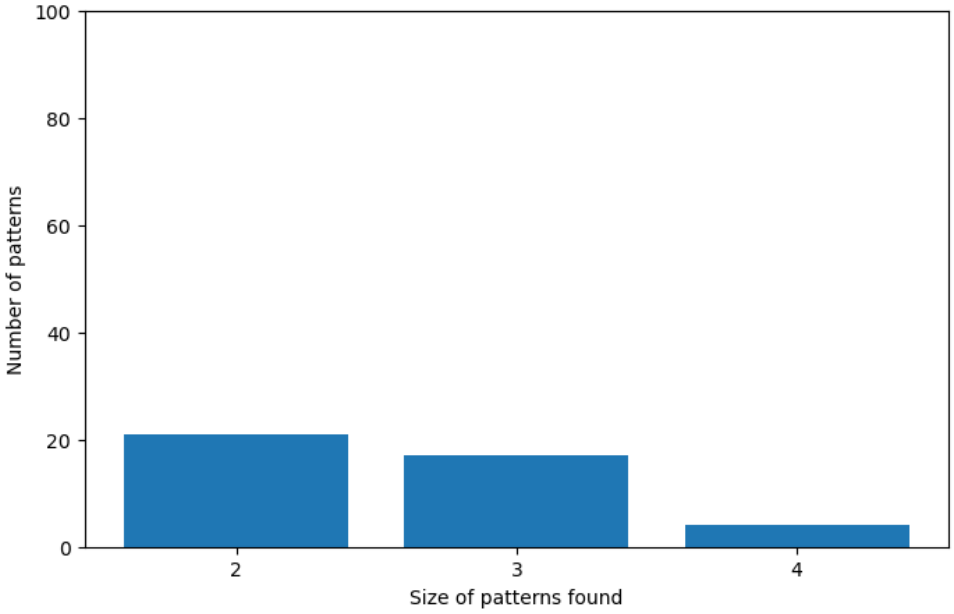


Figure 3.8: Pattern distribution according to the size of the patterns found in the case "b".

Like 3.4, the function 3.6 seems linear but in this case there is a degrowth. In the case that will call "b", in which the prevalence threshold is equal to 0.3 the number of patterns found is 43 with an execution time near to 0. Considering a time of 0.2 seconds, the average number of patterns per second is 215. This number is very different from the previous one; a possible explanation of this phenomenon can be found in the complexity of Column calculation method. In fact, in the complexity analysis the algorithm has an exponential factor and so, average time per pattern depends on the size of the patterns found. Therefore, when it is tested the performance of the method with a higher prevalence threshold, most of the patterns are small and so the average number of patterns per second increases exponentially. For small patterns, patterns with size equal to 2 or 3 are intended. As proof of this in the Figure 3.7 and 3.8 are shown the distribution of the patterns according to their size in the case "a" and "b". In general, most of the patterns have their size equal to 2,3,4.

Chapter 4

Spatial-temporal Pattern Mining based on Neighbor-Dependency Partition and Column Calculation

The topic of "Spatial-Temporal Pattern Mining," which is the extraction of space-time models, or events that occur in a given location at a certain time, will be covered in the subsequent thesis chapter.

Data mining techniques are needed to identify connected events both geographically and temporally since their identification can be challenging and calls for the examination of a lot of data.

The significance of this kind of study comes from the fact that there are frequent connections between events taking place in a particular location during a certain time. For instance, traffic patterns, high-crime regions, and user behaviors in certain places may all be studied using spatial temporal pattern detection.

Spatial-temporal pattern extraction may be used to relate a person's behaviors to other people and locations of interest in order to understand their own routines. For instance, one may examine a person's behavior throughout the course of the day to learn what pursuits and locations they enjoy.

This might be helpful for commercial objectives, such as sending targeted advertising, or for urban planning purposes, such as identifying regions that need greater public service development.

For these reasons, the focus of this experimental part of the thesis is Spatial-Temporal Pattern Mining, especially in human behavior understanding.

Figure 4.1: People use case.

In conclusion of this brief introduction, it is said spatio-temporal pattern analysis is an essential activity that may be used in a number of situations ranging from urban safety to tailored marketing. Data mining techniques may be used to detect them, and they are a useful source of information for understanding an individual's routines and the linkages between geographical and temporal occurrences.

4.1 Use Cases

Nowadays, each person, during the day, interacts with a lot of entities. To design a good algorithm that takes into account most of these entities, 3 main use cases have been defined.

4.1.1 People only

Considering the assumption that if two people are friends or they have any sort of relationship between them, they will share a portion of their time together and so they will be in the same place at the same time. Overall, these two people are correlated with each other. An example of this use case is shown in 4.1. In the plot are shown 6 subplots representing 6 consecutive time points that go from the left-top side to right-bottom side of the plot. For each sub-chart, people's positions are displayed and each person is represented by a point of a specific color. The chart wants to analyze the trajectory of the person in red and related neighbors during the time. The red point highlighted with the black circle is the last point of the trajectory and the circle represents its neighborhood. In the table 4.1 the neighbors for each time point is monitored. As can be seen, in the first three instants, the person in red doesn't meet anyone, then he meets the blue person, they go together and in the last time point they meet the person in orange.

Person	Time point	Neighbor
Red	1	{}
Red	2	{}
Red	3	{}
Red	4	{Blue}
Red	5	{Blue}
Red	6	{Blue, Orange}

Table 4.1: Neighbors dataset of the red point in People only use case

4.1.2 People-POI

The second use case is related to a sort of relation between a person and a point of interest. A point of interest (POI) is a specific point location that someone may find useful or interesting like a restaurant or a shop.

Figure 4.2: People-POI use case.

The based assumption is that a person and a POI have a sort of relation if the person frequently attends that place and therefore a portion of his time is spent in that location. In that case, the person and the POI are correlated. The plot 4.2 represents an example of people-POI correlation. The chart has the same structure of the previous one: it consists of 6 sub-charts that represent the position of the person in red in 6 consecutive time points. The unique difference with respect to the plot 4.1 is the point in blue that represents a point of interest and so its position doesn't change over time. In the example, the person in red goes two times to the POI; in fact, through the table 4.2 it is seen that the POI appears in the neighborhood of two red points, in other words,the POI appears in the neighborhood of the person in red in two different timestamps.

Person	Time point	Neighbor
Red	1	$\{\}$
Red	2	$\{\}$
Red	3	$\{Blue\}$
Red	4	$\{\}$
Red	5	$\{\}$
Red	6	$\{Blue\}$

Table 4.2: Neighbors dataset of the person in Red in People-POI use case.

4.1.3 People-vehicle

The motivation of the People-vehicle use case is based on the assumption that most of time the people share the same transport vehicle to move like their own vehicle or public vehicles (tram, metro or bus). Therefore, the person is correlated with the transport vehicle. The vehicles can be seen like people since they can move and their position can change unlike the POI. Therefore this use case can be mapped as the people-only case considering the vehicles like people.

4.2 Implementation

In this section, it will be presented the implementation of the algorithm, some reasoning useful to understand how to efficiently design the method and consider the time dimension in the process.

4.2.1 Reasoning

Considering the previous use cases, the mandatory data to perform spatial-temporal correlation are the following:

$$< person\_id,\ timestamp, (latitude,\ longitude) > \tag{4.1}$$

Thus, for each person, it is wanted to know its id number and the list of positions and related timestamps. Since the aim of the algorithm is to find the correlation among people, the id number of the person can be considered the feature of the spatial dataset and each point with coordinate and timestamp the instance of a specific person. It is applied a change of paradigm, summarized in the table 4.3 with respect to the mapping in spatial correlation.

Type of correlation	**Feature**	**Instance**	**Purpose**
Spatial	The category of the entities	The entity	Correlation among categories
Spatial-temporal	The entity itself	The entity in a specific location in a certain time	Correlation among entities

Table 4.3: Change of paradigm

Before, the instances were the different entities and the features were the categories of these entities and the aim of the process was to find the correlation among categories. Now, the features are the entities, the instances are the points of the entities in space and time, and the aim of the method is to find correlations between entities. The main difference is that in spatial correlation it is taken into account entities that do not move like plants, points of interest. So, to perform spatial-temporal pattern mining, it is added the time dimension. As identifier of the instances, the tuple $< person\_id, timestamp >$ can be used since the same person cannot be in two places at the same time and so this tuple is unique. The desired output of our method, which must be provided as input to 1, must be of the following form:

$$< o_i, o_j > \tag{4.2}$$

in which the feature of o_i is greater alphabetically of o_j and these two instances have the neighbor relationship 2.2 and are under the Time constraint.

$$T(o_i, o_j) \Longleftrightarrow |o_i.ts - o_i.ts| < min\_t \tag{4.3}$$

Given two instances o_i,o_j and their timestamps: $o_i.ts$, $o_i.ts$, if the time distance between $o_i.ts$, $o_i.ts$ is less than a certain threshold, they will satisfy the Time relationship $T(o_i, o_j)$.

Given a dataset of N timestamped points, checking each pair of points is a problem since it involves cartesian product with a complexity of $O(N^2)$. In the following section, the solution to this problem is addressed.

4.2.2 Design choices

To summarize the entire algorithm from the input dataset to the desired outputs, figure 4.3 describes the entire pipeline.

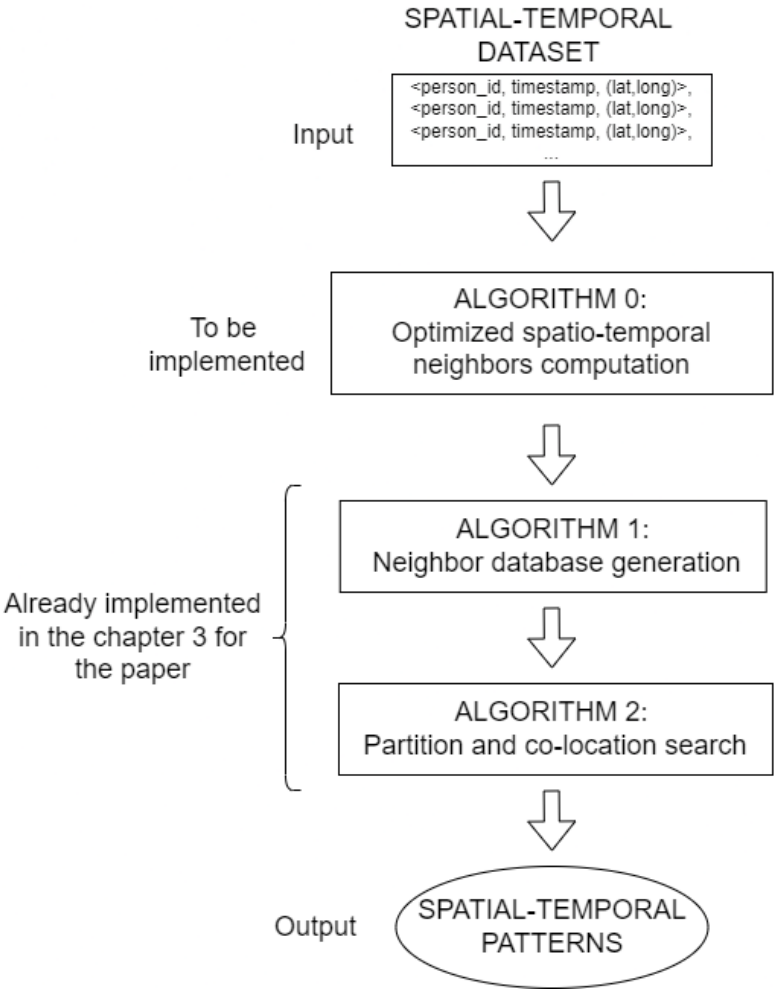


Figure 4.3: Spatial-temporal pipeline.

The algorithm to be developed analyzes data with spatial and temporal information 4.3. The goal of such algorithm is to extract pairs of points satisfying the neighbor (distance) constraint and temporal constraint. A possible solution to this problem can be addressed by building a unique Ball tree and querying for each instances. Despite its efficiency, in case of millions of instances, the BallTree takes a lot of time. So, another method is needed. The solution based on a unique balltree is a sequential, non-parallelizable therefore the method doesn't leverage the power of the parallel computing of the cluster. So, a solution that involves parallelization could be useful.

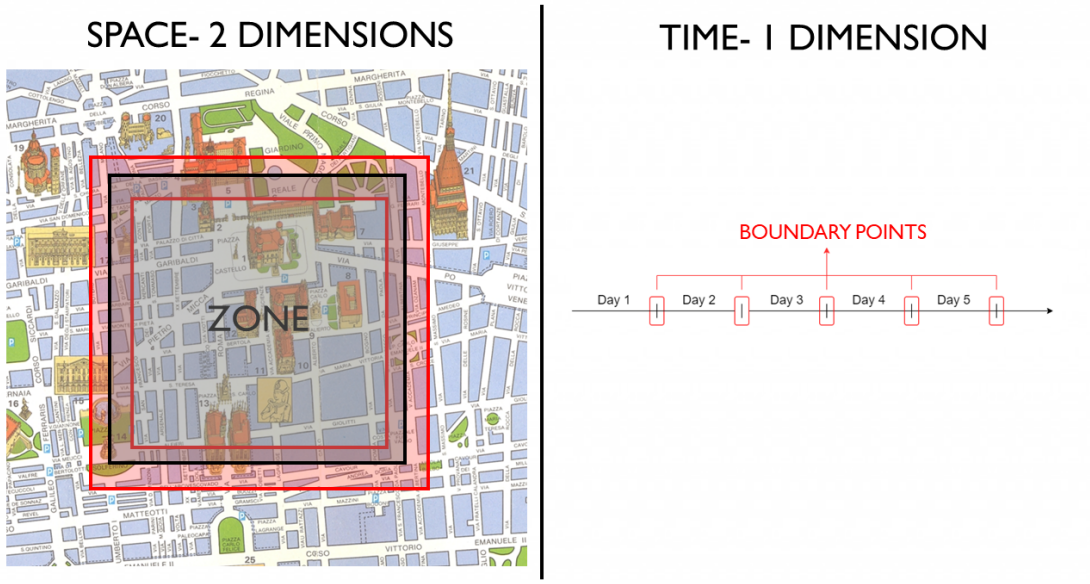


Figure 4.4: Dimension problem.

To parallelize the algorithm, for each instance the set of neighbors must be identified, considering the spatial and temporal dimension. But how to group them? Two possible ways are along the space that has 2 dimensions or along the time with 1 dimension. A visual example is shown in figure 4.4, in which are compared these two cases. On the left it is analyzed the choice of the space and on the right the time. Starting from the left, given a zone delimited by the black rectangle that groups a set of instances, all points located near the black line are named boundary points and are critical, due to the possibility of having neighbors outside the considered partition (black box). In fact, if a BallTree is applied in the zone in black, the neighbors of boundary points may be incomplete. Analyzing the time choice, it is seen that, imagining splitting the time into days, for each day, boundary points are located only at the beginning and at the end of the day.

As can be seen, the choice of time seems easier to handle. However, this choice depends on the distribution of the dataset. As an example if the dataset consists of all instances in which timestamps refer to the same day, the choice of the time may be not the correct one. In conclusion,the choice taken is based on the temporal data split: given the entire dataset, multiple partitions are generated based on a one-day split.

4.2.3 People-only algorithm

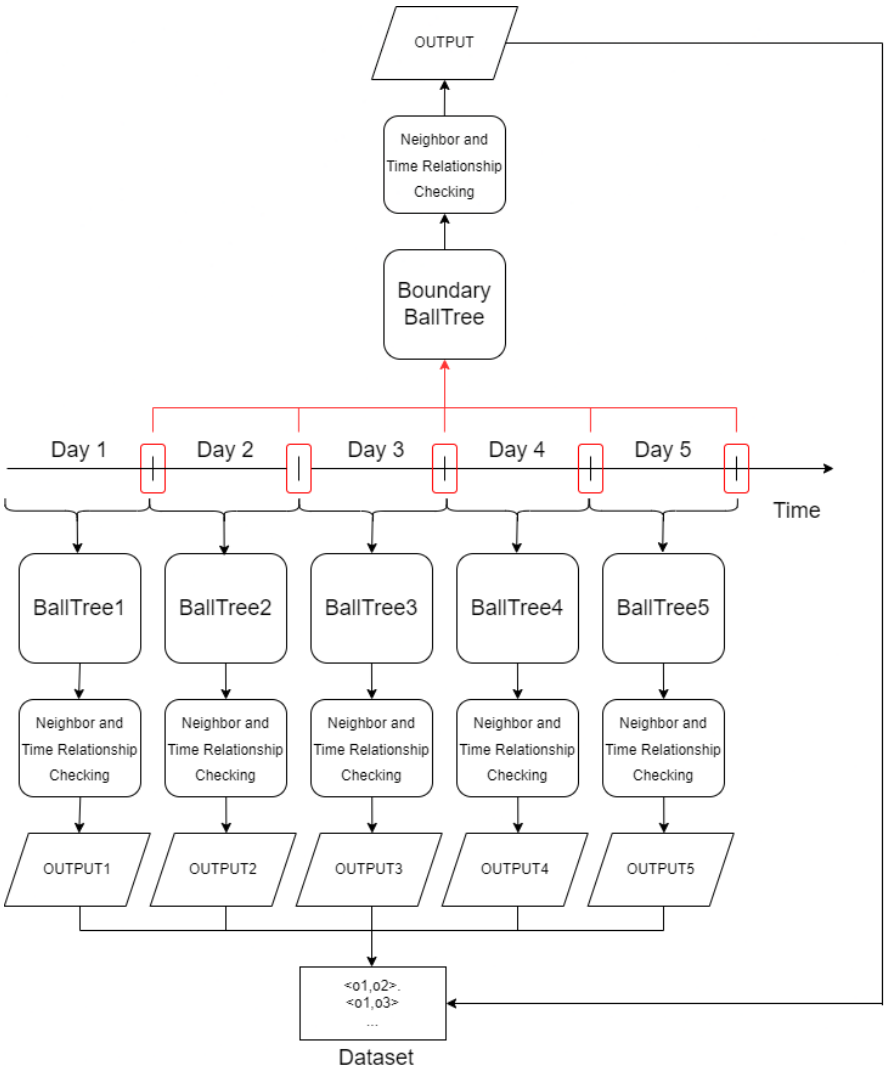


Figure 4.5: Parallel BallTree Design.

This subsection considers just the People-only use case and so, all instances have the timestamp. The figure 4.6 shows the structure of the algorithm. The spatial temporal dataset is divided into sets according to the day of the timestamp. Then, for each day a BallTree is built and for each pair of closed points the time relationship is checked. At the same time, to tackle the aforementioned boundary points' case, all the boundary points (red lines in the schema) are processed by a further Ball Tree to verify the spatial and temporal constraints. The space distance is computed by the haversine distance. The output of each partition is a list of instance pairs $< o_i, o_j >$. In conclusion, all these lists are merged in a single dataset that will be used to feed the Algorithm 1.

4.2.4 People-POI algorithm

This subsection considers the People-POI use case and so, the instances related to the points of interest don't have the timestamp. For this use case, there are two ways to consider the POIs:

- Unique POIs: each point of interest is considered a unique one and so, it doesn't belong to any category. In this case the POI will be mapped in this way:

$$< POI\_id,\ (lat, long) > \tag{4.4}$$

- Category POIs: each point of interest is not considered a unique one and so, it belongs to a category. In this case the POI will be mapped in this way:

$$< POI\_category,\ POI\_id,\ (lat, long) > \tag{4.5}$$

In the first case the feature of the instance is the identifier; instead in the Category POIs case, the feature of the instance is the category to which the POI belongs. In the first case, the entire algorithm will consider all POIs unique and so, will search for correlations between people and specific POIs. An example pattern is be $\{John, Frank'sRestaurant\}$ and its meaning is: John frequently goes to Frank's Restaurant. In the second case, instead, considering a POI part of a category, the entire algorithm will search correlation between people and the category of the POI. An example could be: $\{John, Restaurant\}$, and its meaning is: John frequently goes to Restaurants.

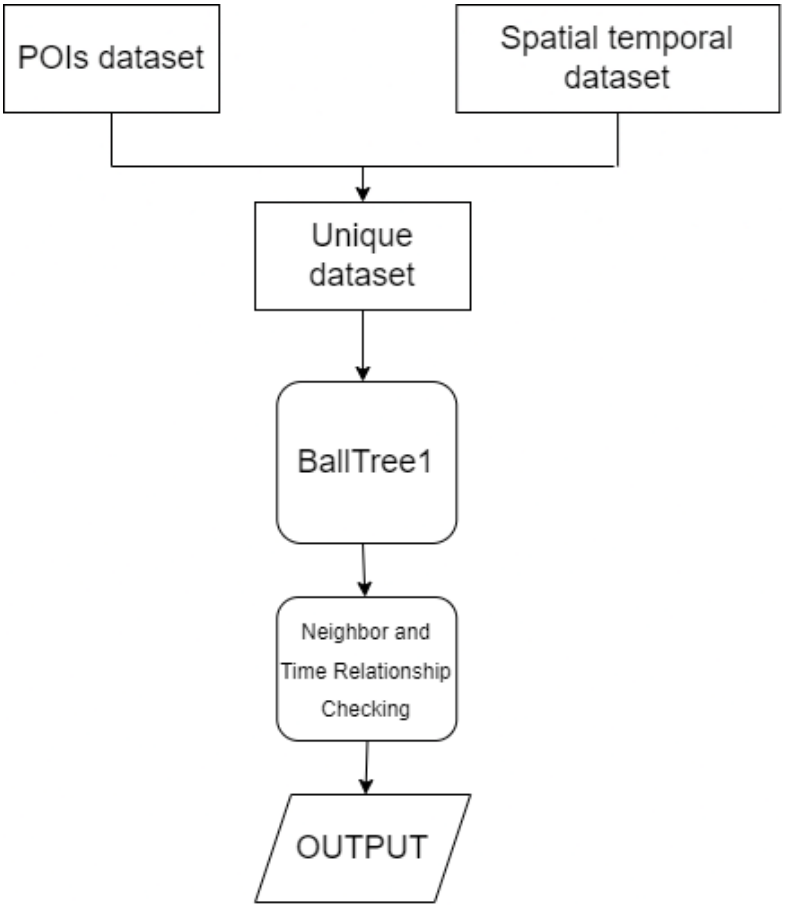


Figure 4.6: People-POI algorithm.

To implement the identification of correlation between people and POIs, the starting point is the algorithm described in Figure 4.6 previously explained. Since the POIs are static, they can't be partitioned according to the temporal strategy mentioned in the previous paragraphs. Therefore, the POIs dataset is added to each partition (also for the boundary partition) to the trajectory Dataset. In this way, the building of the ballTree takes into account also the points of interest. In the last step namely "neighbor and time relationship checking", when the pairs <person,POI> are checked, only the neighbor relationship will be searched without any verification on the temporal constraint due to the absence of a timestamp associated to a POI. To clarify, this second algorithm proposed finds both correlations among people and between people and POIs. Thus, it is an extension to the aforementioned People-only case

4.3 Experimental Evaluation

This section presents the experimental setup and the obtained results of the proposed algorithm.

4.3.1 Setup

The experiments are conducted on a real Dataset called Geolife trajectories. This GPS trajectory dataset was collected in Microsoft Research Asia's Geolife project. It involed 182 participants over a period of 3+ years. A GPS trajectory is represented by a sequence of time-stamped points, each of which contains information about latitude, longitude and altitude. This dataset contains 17,621 trajectories with a total distance of about 1.2 million kilometers. Most of the points are located in Beijing (China). The total number of time-stamped points is around 20 million. Due to computational issues and limited resources, one-sixth of the original dataset was considered in this work, with a total number of 3.3M points. The dataset was sampled with an irregular sampling rate, with a variable time distance between two consecutive samples ranging from 1 to 5 seconds. To reduce the number of points to be processed without any loss of generality of the proposed approaches, we undersampled the dataset to a sampling period of 30 seconds. In figure 4.7 the top 10 people with the highest number of points are shown. The second dataset used is related to the points of interest of Beijing. To obtain this dataset we used OSMnx, that is a Python package that lets you download geospatial data from OpenStreetMap and model, project, visualize, and analyze real-world street networks and any other geospatial geometries. Doing so, 10148 POIs are downloaded : and injected into the dataset. Figure 4.8 showns the top 10 categories of points of interest according to the number of POI.

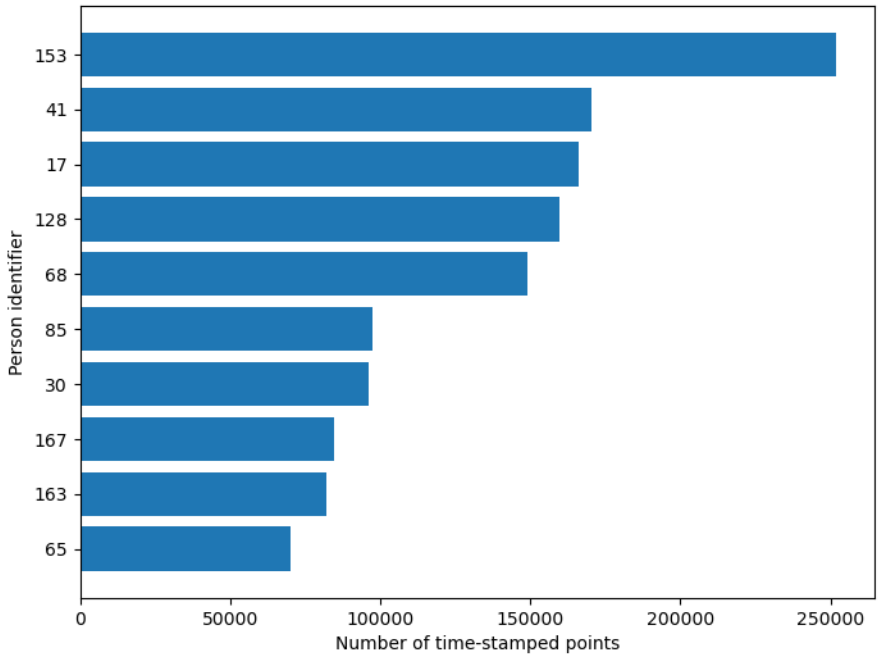


Figure 4.7: Top 10 people according to the number of points .

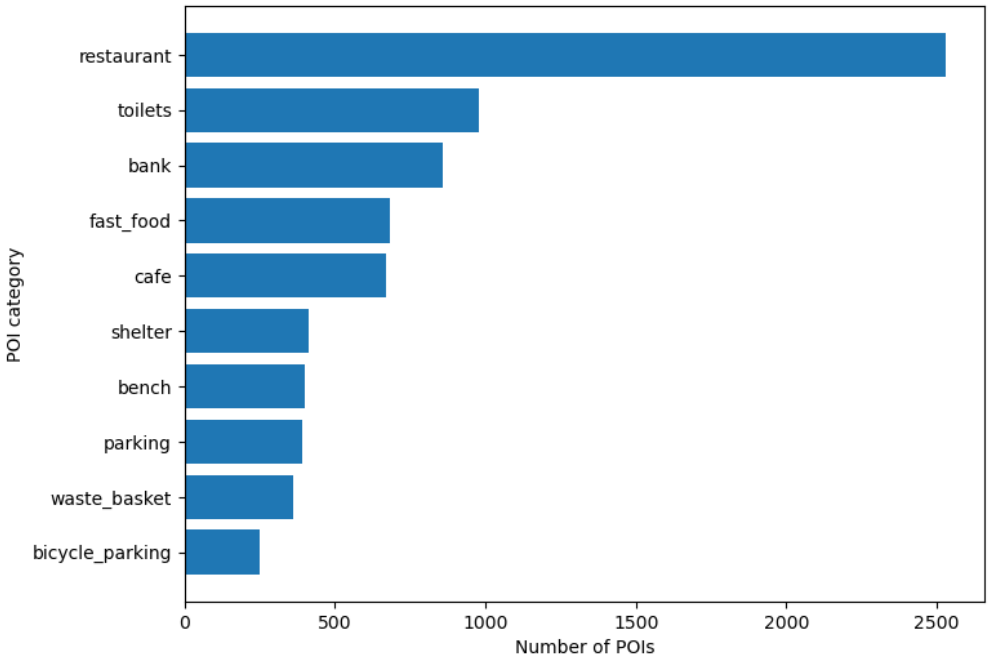
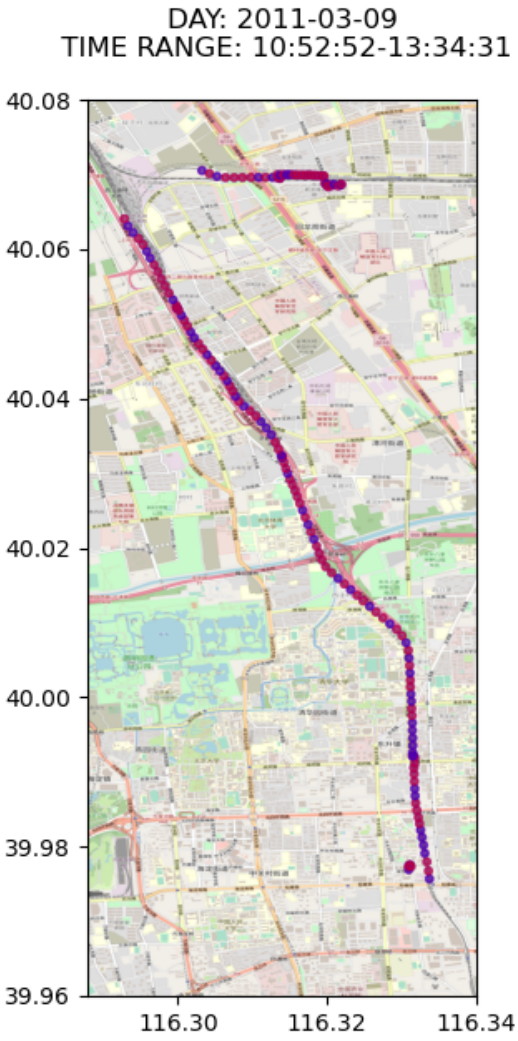


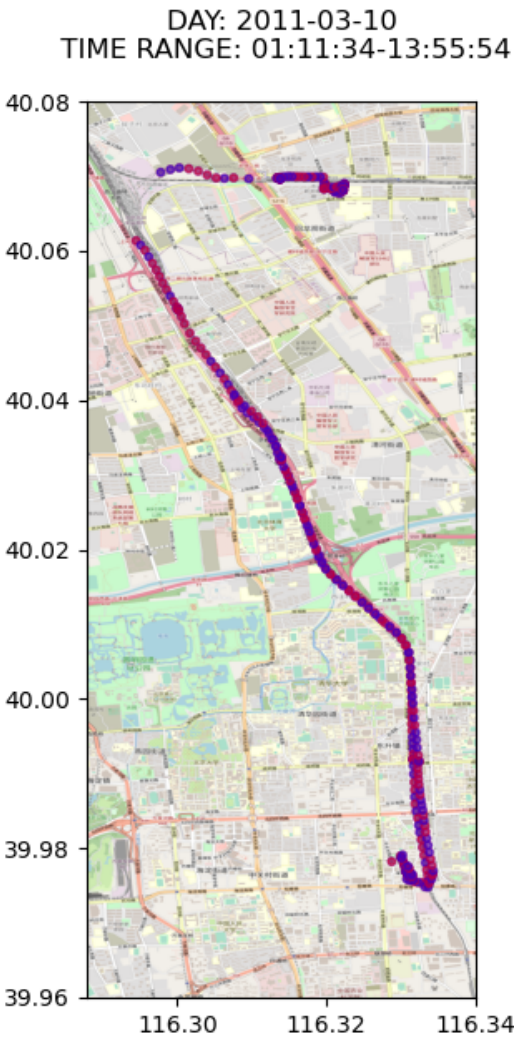
Figure 4.8: Top 10 POI categories according to the number of POI.

The previously described algorithms are implemented in Python, using the PySpark library and run on the SmartData@PoliTO cluster.

4.3.2 Pattern visualization

Analyzing and interpreting spatiotemporal patterns is a difficult work because it requires dealing with complex interactions and dependencies that span both in space and time. One way to overcome these challenges is to use visualizations to explore and interpret spatiotemporal patterns. Visualizations allow us to represent complex data in an accessible and intuitive way, making it easier to identify and verify prevalence patterns. Moreover, visualizations can help us understand the meaning and significance of the participation index, a measure that reflects the degree of involvement of an entity in a given spatiotemporal pattern. By visualizing the patterns and the participation index, we can gain a deeper understanding of the spatial and temporal dynamics underlying the data and their practical implications. In this section, two found patterns are analyzed from a visual point of view. Specifically, we investigate the patterns, where two individuals frequently interact in a specific spatiotemporal context or where an individual tends to visit a particular location repeatedly. The visualizations illustrate these patterns by creating graphs for each day in which the individuals are close in both space and time or the individual is close to the point of interest. The participation index is adapted to the spatiotemporal domain, reflecting the percentage of time and space the entities spend together or the individual spends close to the point of interest within the observed period. The settings used to find spatiotemporal patterns are a time constraint of 1 second and 50 meters as a space constraint. The first pattern is related to two people, the first one has 58 as an identifier and it is characterized through 4138 time-stamped points in the dataset.


DAY: 2011-03-09
TIME RANGE: 10:52:52-13:34:31
40.08
40.06
40.04
40.02
40.00
39.98
39.96
116.30
116.32
116.34


DAY: 2011-03-10
TIME RANGE: 01:11:34-13:55:54
40.08
40.06
40.04
40.02
40.00
39.98
39.96
116.30
116.32
116.34

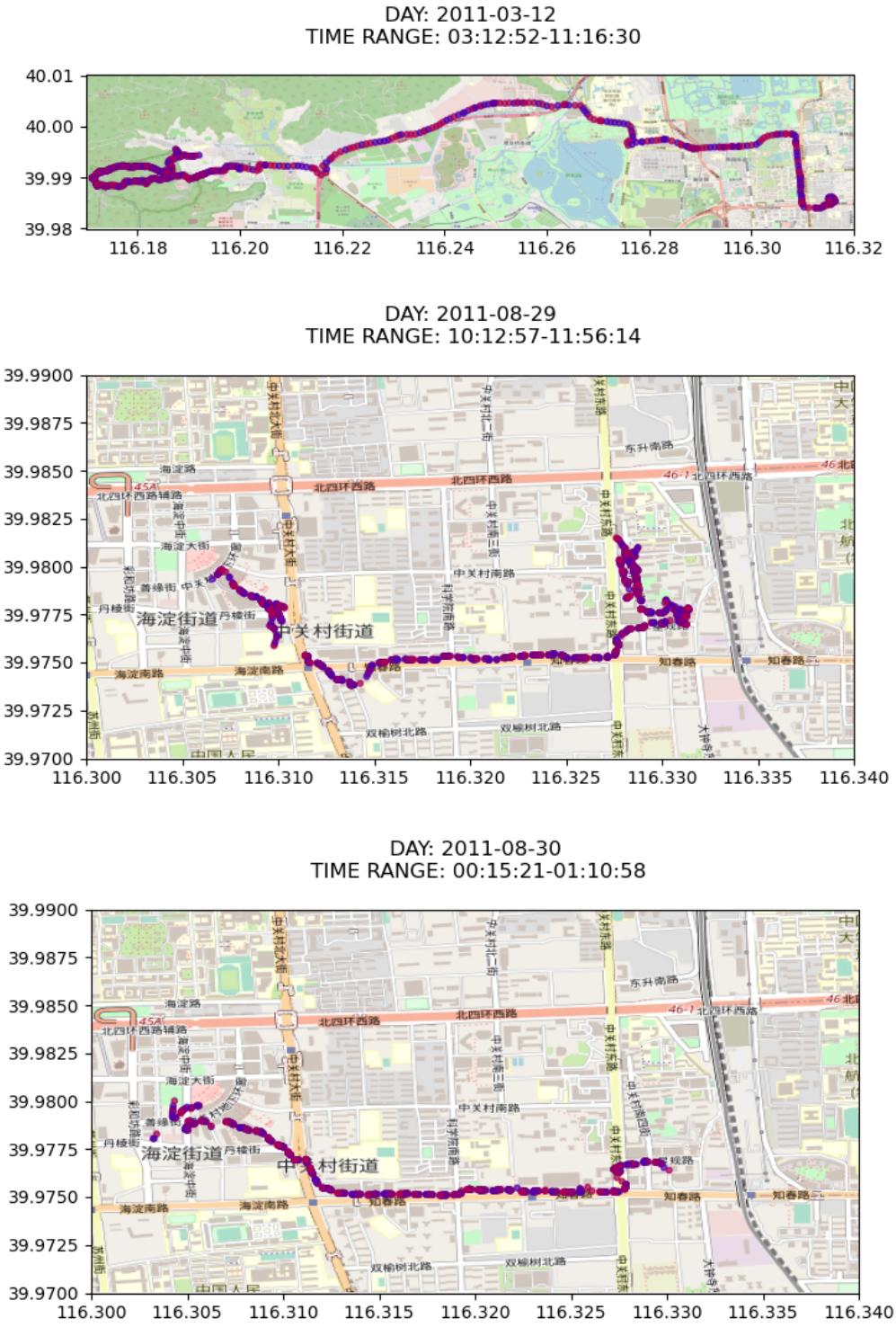
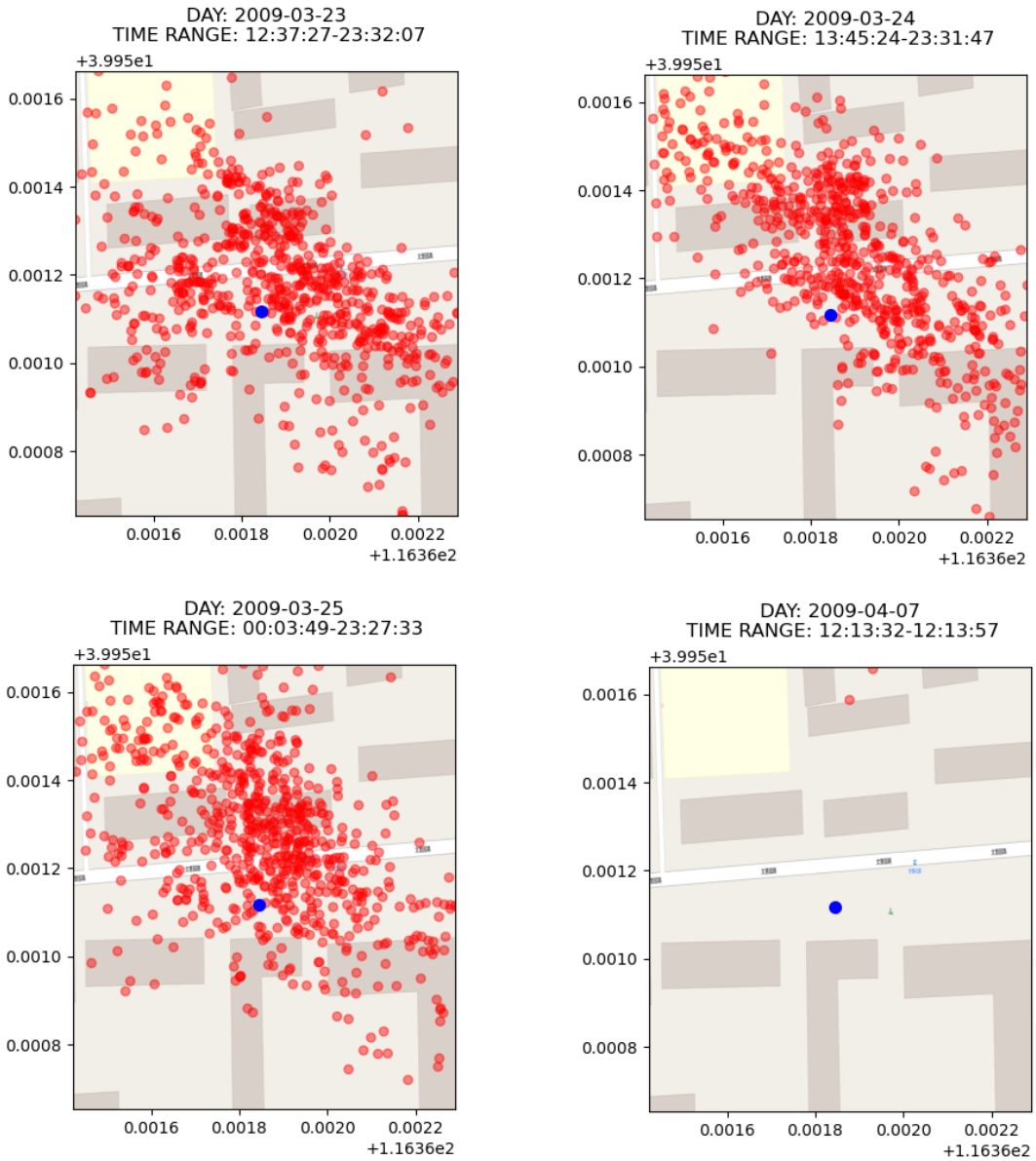


Figure 4.9: All the points of the people 58 and 59 that are close to each other according to the day.

The second person, instead, has 59 as an identifier and it is described through 3937 time-stamped points. The proposed algorithm has found that these two people are correlated with a participation index of 0.95. What is the meaning of such number?

It means that, given all the points of person 58, at least 95% of these ones are closed in terms of time and space to a point of person 59. Specularly, given all the points of person 59, at least 95% of these points are close to a point of person 58.



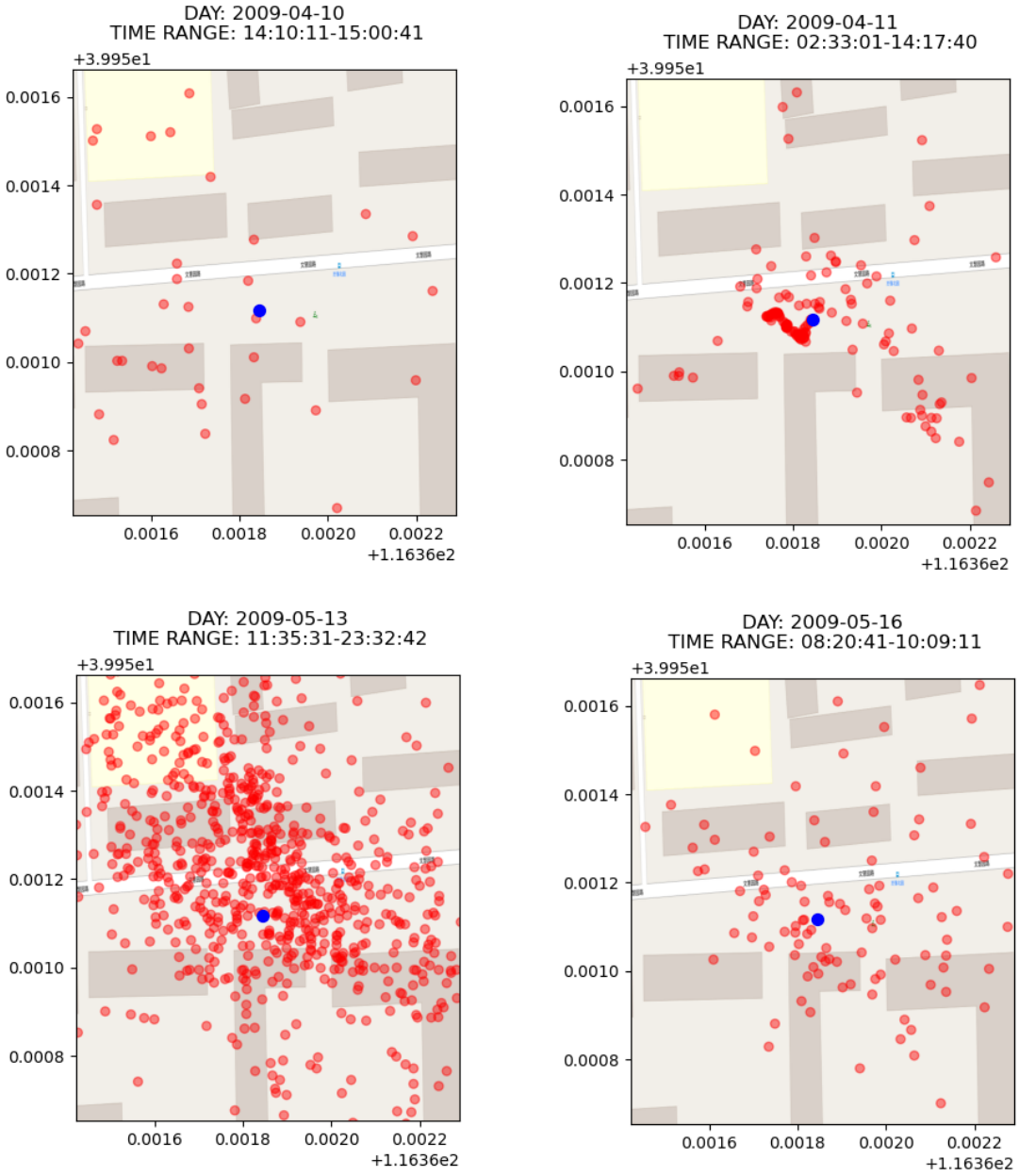


Figure 4.10: All the points of person 40 that are close to the point of interest according to the day.

Said that, in the graphs, person 59 is described by points in red instead, the person 58 by points in blue. As seen in Figure 4.9, each graph is characterized by a day and a time range in which the points are collected. The points in red and blue follow the same trajectory every day. Based on these visual representations, it

is possible that people 58 and 59 have a strong relationship and so they potentially be wife and husband, or friends leaving nearby and sharing the same place of work as two examples. The second pattern to analyze is related to a person and a restaurant. The person has 40 as an identifier and it is characterized through 9348 time-stamped points in the dataset. The proposed algorithm has found that these two people are correlated with a participation index of 0.32. Like the previous case, it means that given all points of person 40, 32% of these ones are close to the point of interest considered. The graphs, in Figure 4.10, have the same structure as the previous ones, and therefore each graph presents a day and a time range. The restaurant is described by a blue point and the person by red points. As can be seen, the person moves around the point of interest. It is possible that the entire zone is full of other services like pubs, supermarkets, and so on. Analyzing also the time ranges reported, it is possible that it is an area frequented both by night and by day.

Overall, these visualizations demonstrate the importance of patterns in understanding spatiotemporal data. We can obtain a better grasp of the underlying processes and their practical implications by incorporating the participation index's extension to the spatio-temporal domain.

4.3.3 Performance

In this paragraph, several experiments are conducted and related results are shown. The first algorithm that has been investigated is the "People-only" method which finds the correlations among people.

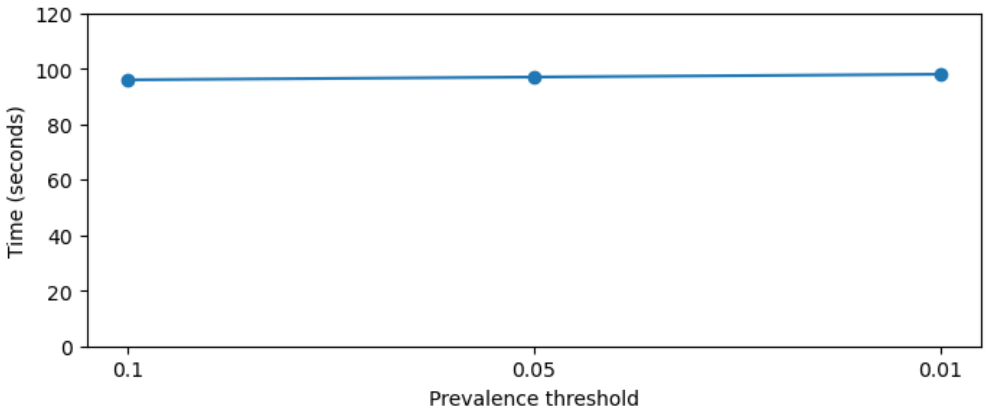


Figure 4.11: Execution time of the People-only algorithm.

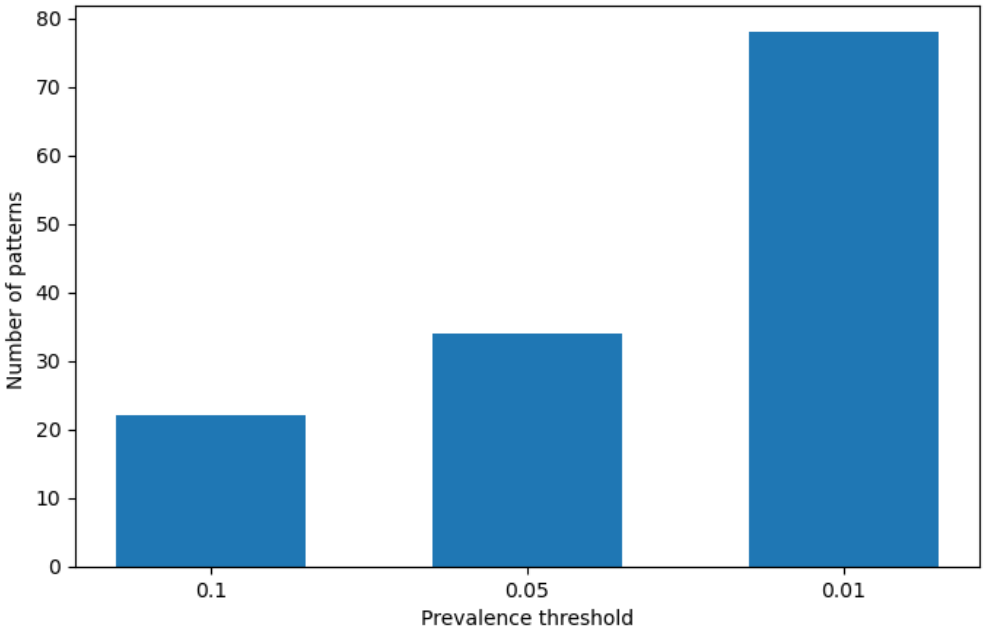


Figure 4.12: Number of pattern found of the People-only algorithm.

Setting the time threshold to 1 second and the distance threshold to 50 meters, Figure 4.11 shows the execution time of the method according to the prevalence threshold that varies from 0.1 to 0.01. The chart shows that the execution time doesn't change. Despite this, Figure 4.12 shows that the number of patterns found changes. In fact, from 0.1 to 0.01 the patterns are quadruplicated. Its behaviour seems at most exponential. The second experiment fixes the time threshold to 1 second and the prevalence threshold to 0.5 and analyzes the number of patterns and execution time as the distance threshold varies. The thresholds used are 50, 100 and 150 meters.

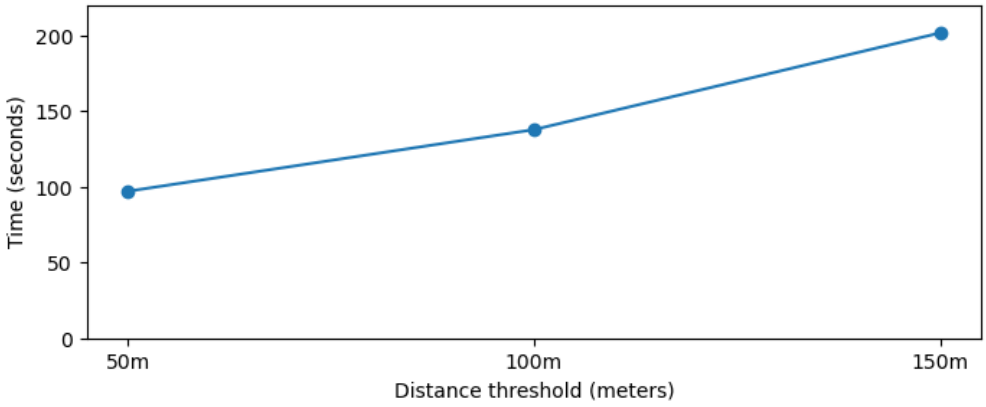


Figure 4.13: Execution time of the People-only algorithm.

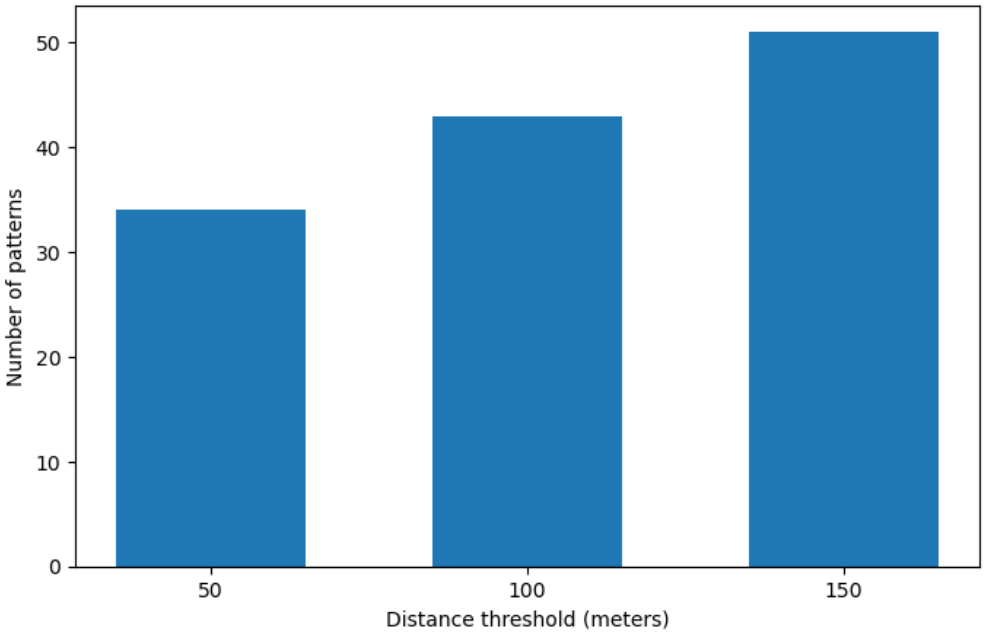


Figure 4.14: Number of pattersn found of the People-only algorithm.

In this case the behaviour is different. The larger the distance threshold is, the longer is the execution time that changes from 100 seconds to 200 seconds. The number of patterns seems to grow linearly with a maximum of 50 patterns. Then, the impact of the introduction of POIs in the dataset is investigated with respect to the performances of the algorithm. In this case each point of interest is considered a unique point that doesn't belong to any category. To clarify, this algorithm finds both correlations among people and correlations between people and POI.

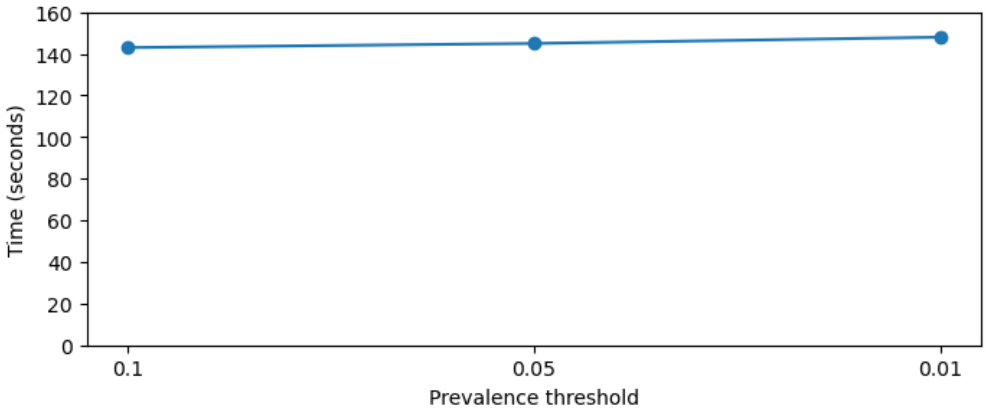


Figure 4.15: Execution time of the People-(unique POIs) algorithm.

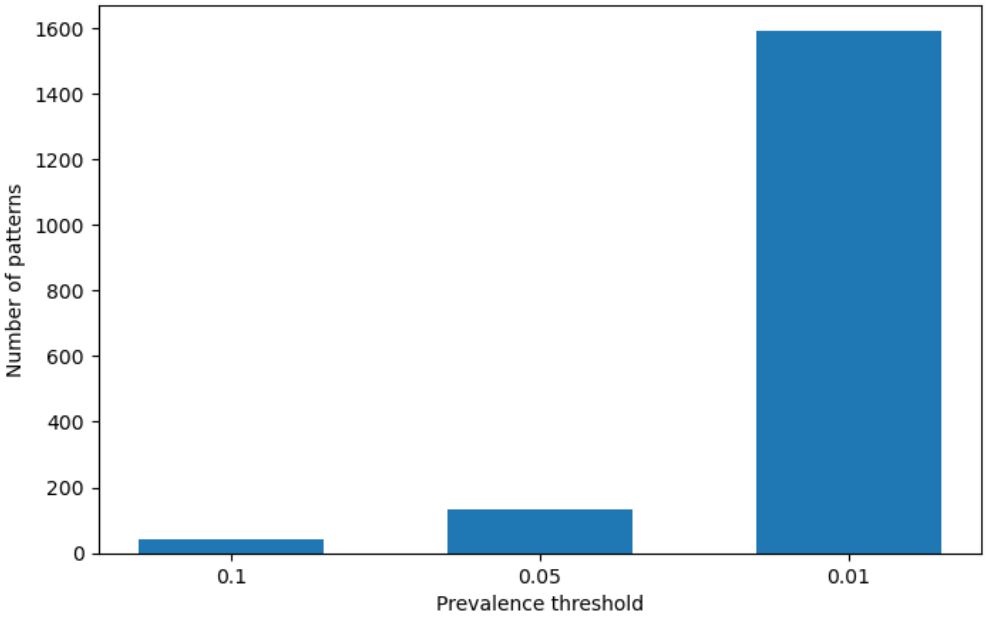


Figure 4.16: Number of pattern found of the People-(unique POIs) algorithm.

Setting the time threshold to 1 second and the distance threshold to 50 meters, Figure 4.15 shows the execution time of the second method. Also in this case the execution time doesn't change and remains at 140 seconds. Instead, the growth of the number of extracted patterns is shown in Figure 4.16. The number of extracted patterns severely increase with a prevalence threshold of 0.1 until at most 1600 patterns. The next experiment fixes the time threshold to 1 second and the prevalence threshold to 0.1 and observes how the time and number of patterns

found change according to the distance threshold. The used thresholds are 50, 100 and 150 meters.

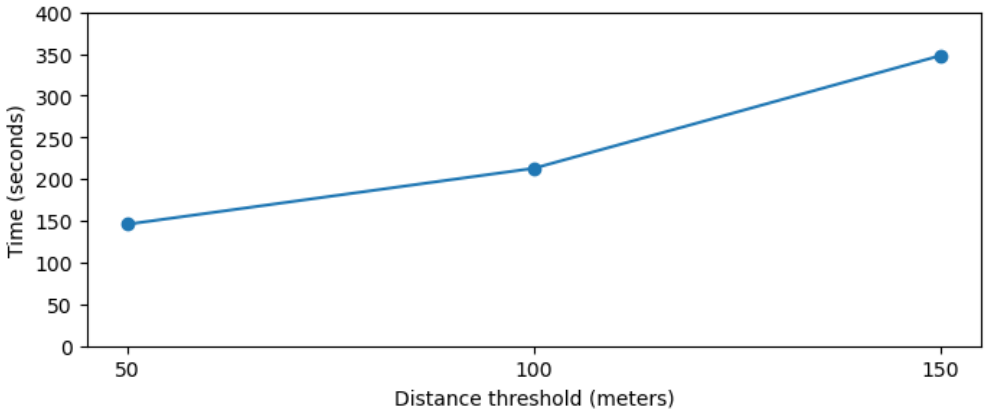


Figure 4.17: Execution time of the People-(unique POIs) algorithm.

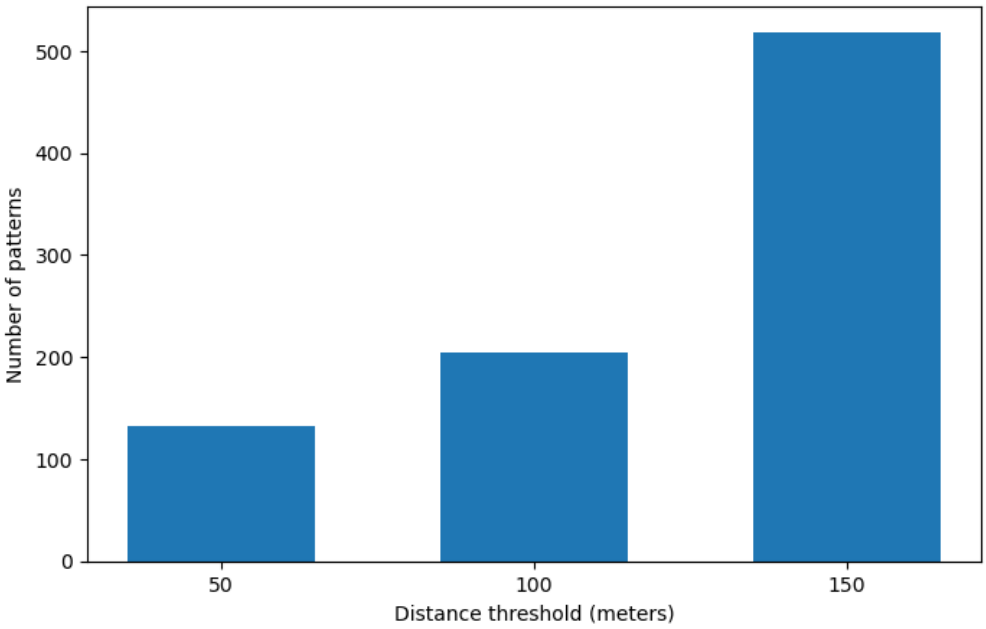


Figure 4.18: Number of pattern found of the People-(unique POIs) algorithm.

In the chart 4.17, a higher increase in execution time is observed passing from 100 to 150 meters with respect to the 50m to 100m case. Similarly, also the number of patterns found grows rapidly when the threshold is set to 150 meters. As it is

seen in the 4.18 the minimum number of patterns is 132 with 50 meters of threshold and the maximum is 518 with 150 meters of threshold.

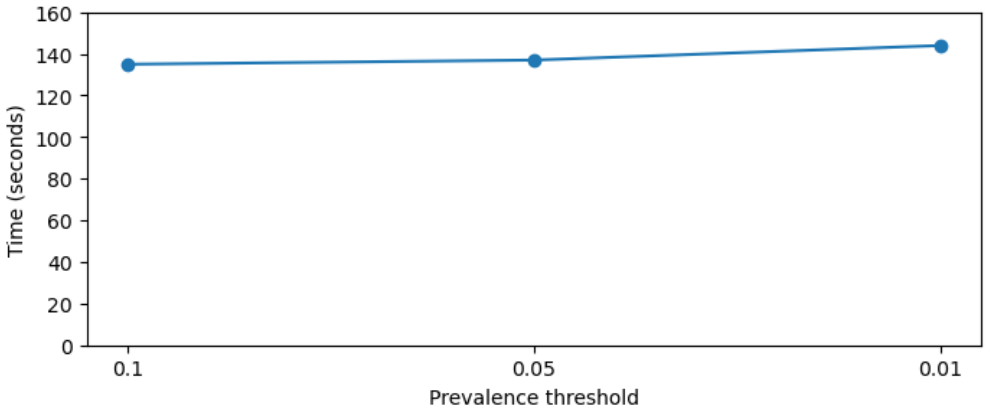


Figure 4.19: Execution time of the People-(category POIs) algorithm.

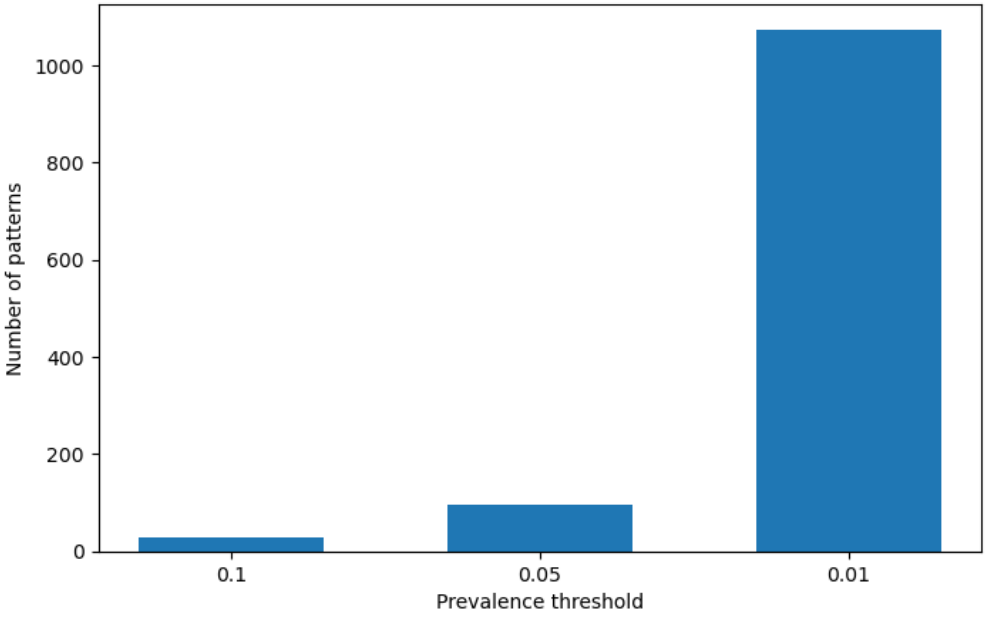


Figure 4.20: Number of pattern found of the People-(category POIs) algorithm.

The next series of analyses are conducted on the third algorithm. It finds correlations between people and POIs, but in this case each point of interest is considered belonging to a category. The first experiments conducted, shown in Figures 4.19 and 4.20, fixes the time threshold always equal to 1 second and the

distance to 50 meters, and experiments fix to analyze how the execution time and the number of patterns found change when the prevalence threshold is modified. The results obtained are very similar with respect to the results previously shown in the case of POIs as unique entities. The main difference is in the maximum number of patterns found with prevalence threshold equal to 0.01. In this case, they are 1073, less than previous case. A possible explanation can be taken as an example of restaurants: a person will tend to go frequently to a specific place to eat, if he likes it, compared to going frequently to several different restaurants. Hence, it is easier to find correlations by considering the unique POI.

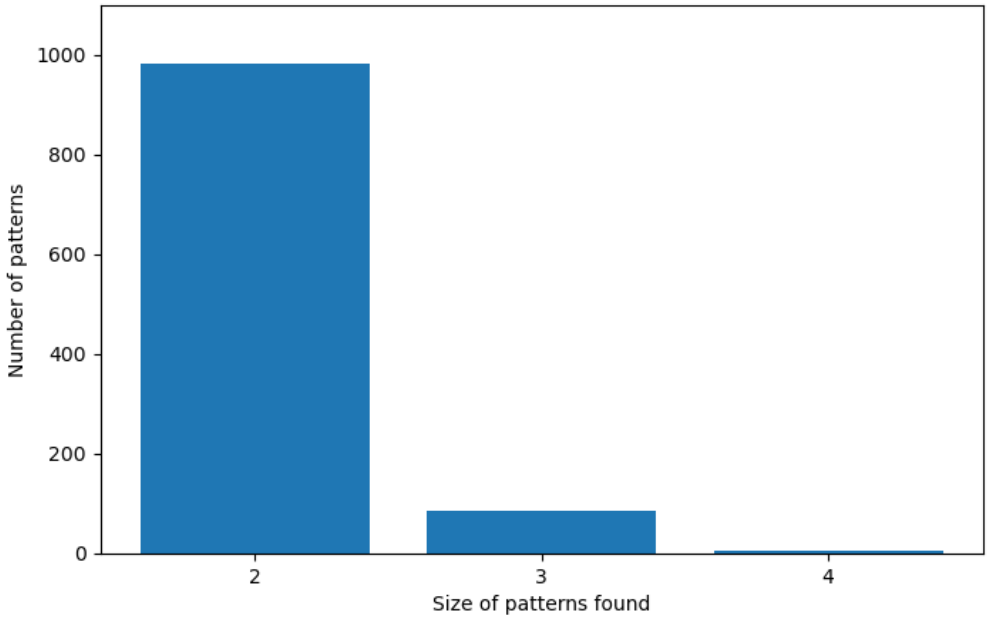


Figure 4.21: Patterns distribution according to their size.

To further investigate the patterns found with prevalence threshold equal to 0.01, Figure 4.21 shows the distribution of the patterns according their size. The distribution displays that most of patterns found have size equal to 2. Next, the performance of the algorithm is analyzed with respect to the value of the distance threshold. Setting the time threshold to 1 second and the distance equal to 50 meters, the experiment wants to understand how the time and the number of patterns found change when the distance threshold changes. Like the previous algorithm, the threshold used are 50, 100 and 150 meters.

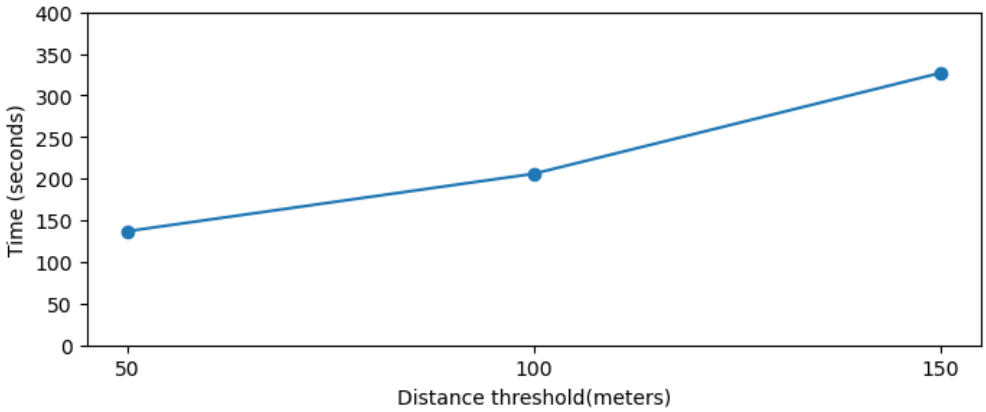


Figure 4.22: Execution time of the People-(category POIs) algorithm.

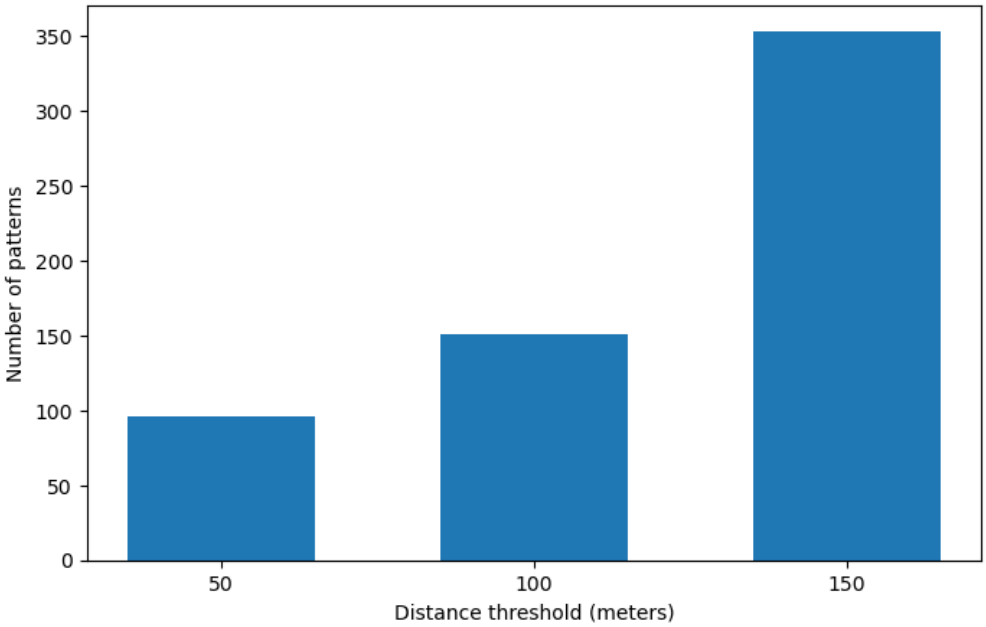


Figure 4.23: Number of pattern found of the People-(category POIs) algorithm.

The results obtained in the plot 4.22 and 4.23 are similar to charts 4.19 and 4.20 obtained with the second algorithm.

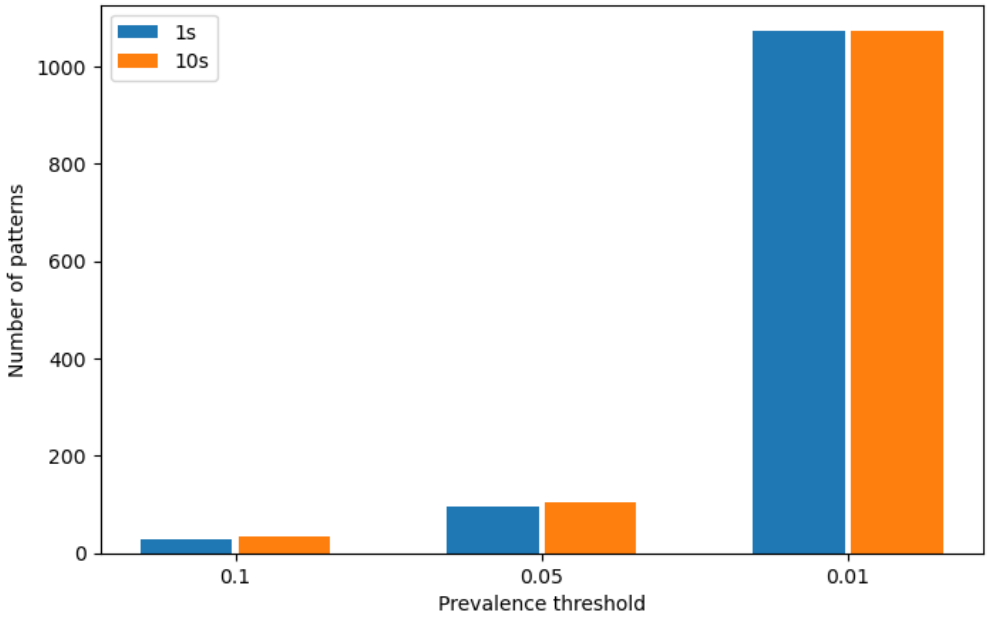


Figure 4.24: Comparison of patterns found using two different time thresholds

The last experiment conducted is related to the time threshold used. The experiment shown in Figure 4.20 has been repeated with a time threshold equal to 10 seconds. The comparison is displayed in Figure 4.24 in which it is seen that for each prevalence threshold the number of patterns found is approximately the same. Also, in terms of execution time, no significant difference is observed.

Chapter 5

Conclusion

In this master thesis, we have investigated co-location patterns: two or more categories of entities that appear frequently close to each other. Their importance is undeniable, from the urban field to the logistics and the medicine, finding these kinds of patterns is very useful. Therefore, understanding the multitude of algorithms invented to find them, will be needed, if we want to discover prevalent patterns. Obviously, the algorithm to use changes according our needs like the size the dataset used, the typology of the correlation that we want to find or if we have constraints on the time in which the output should be ready. Spatial instances could retrieve tons of information but the question is: which information are we interested in? Do we want to know if two categories are correlated in space or over time? Are we interested in understanding how the entities frequently occurs? When they appear together, is a category the prevail on the other? These kinds of information are obtained through different algorithms and so it is necessary to understand which is the most suitable algorithm depending on the task. There is an important category of patterns called co-location patterns that reveals underlying correlations between entities in the space. To find co-location patterns, several algorithms have been implemented through different approaches. A step forward has been done by the designing of parallel algorithms run on specific hardware such as GPU and cluster. These technologies speed up the discovery of patterns and allow the usage of huge dataset. However, the state of the art algorithms that use the parallel approach are stuck because of two problems. The first problem is related to the independence of the partitions that will be run in parallel. Finding co-location patterns is a sequential process and so it is difficult to build a method that takes full advantage of the parallelization. The second problem is a technical one, to find a co-location pattern is necessary to build the instance table: a huge table that contains all combinations of instances belonging to different categories. The novel algorithm re-implemented in the master thesis tackles these two problems obtaning competitive performances. The proposed algorithm is based on two main parts.

The first one is related to how to divide the instances to take advantage of the parallelization. The novel solution is based on the concept of "neighbor-dependency partition" and is able to divide the entire dataset, through the prefix strategy, into subset which are independent to each other, containing all the required information to compute the participation ratio. This solution tackles the independence problem. For the second step of the work re-implemented, it introduces a novel algorithm called "Column Calculation" that allows the identification of all instances that take part in the pattern without building the huge table previously mentioned. The column calculation will be used on each partition for each candidate pattern found. Using The developed algorithm in combination, state-of-the-art results are obtained. The algorithm reimplemented in the context of this thesis was compared with the performance declared by the original authors (which did not disclose the source code), demonstrating similar performances. The second part of the master thesis introduces a novel pipeline by extending the aforementioned algorithm by adapting the spatial co-location mining algorithm to the spatio-temporal domain adding a new algorithm. The new algorithm defines a new neighborhood not just geographically, but also temporally. In a nutshell, two entities are close if and only if they are in the same place at the same time. Based on this assumption, the proposed algorithm maps the instances into instance pairs, in a parallel way, only if they respect the time and space constraint. The output of proposed algorithm is used as input of the re-implemented one and in combination are able to return spatial-temporal co-location patterns, in particular we can extract people behaviour finding correlation among people and between people and points of interest. The great advantage of the method is the ability to handle large datasets with limited execution time as as seen in the experimental analysis conducted on a dataset of 3.5 millions of time-stamped points. There are, however, various potential directions for expanding and improving on this study. To begin, one possible future research route would be to expand the spatial-temporal correlation analysis to points of interest (POIs) with more complicated shapes, such as polygons. Because it takes into consideration the exact form and position of individual POIs, this would allow for a more precise and nuanced understanding of the association between people and points of interest. Second, another feasible enhancement would be to investigate other temporal correlations between events. For example, rather than restricting the analysis to precise co-presence of individuals, one may look into the relationship between events that happened many days or even weeks apart. This would offer a more complete picture of complex events.

Bibliography

[1] Peizhong Yang, Lizhen Wang, Xiaoxuan Wang, Lihua Zhou, and Hongmei Chen. «Parallel Co-location Pattern Mining based on Neighbor-Dependency Partition and Column Calculation». In: Nov. 2021, pp. 365–374. DOI: 10.1145/3474717.3483984 (cit. on pp. 5, 21, 28).

[2] S. Shekhar and Y. Huang. «Co-location rules mining: a summary of results». In: Proceedings of SSTD (2021), pp. 236–256 (cit. on pp. 6, 9, 10).

[3] Y. Huang, S. Shekhar, and H. Xiong. «Discovering colocation patterns from spatial data sets: a general approach». In: *IEEE Transactions on Knowledge and Data Engineering* 16.12 (2004), pp. 1472–1485. DOI: 10.1109/TKDE.2004.90 (cit. on p. 11).

[4] Jin Soung Yoo and S. Shekhar. «A Joinless Approach for Mining Spatial Colocation Patterns». In: *IEEE Transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1323–1337. DOI: 10.1109/TKDE.2006.150 (cit. on pp. 11, 16).

[5] Lizhen Wang, Yuzhen Bao, and Zhongyu Lu. «Efficient Discovery of Spatial CoLocation Patterns Using the iCPI-tree». In: *The Open Information Systems Journal* 3 (Sept. 2009), pp. 69–80. DOI: 10.2174/1874133900903020069 (cit. on pp. 11, 16).

[6] Jin Soung Yoo, Douglas Boulware, and David Kimmey. «A Parallel Spatial Co-location Mining Algorithm Based on MapReduce». In: (2014), pp. 25–31. DOI: 10.1109/BigData.Congress.2014.14 (cit. on pp. 11, 18).

[7] Jin Soung Yoo, Douglas Boulware, and David Kimmey. «Parallel Co-Location Mining with MapReduce and NoSQL Systems». In: *Knowl. Inf. Syst.* 62.4 (Apr. 2020), pp. 1433–1463. ISSN: 0219-1377 (cit. on pp. 11, 20).

[8] Peizhong Yang, Lizhen Wang, and Xiaoxuan Wang. «A MapReduce approach for spatial co-location pattern mining via ordered-clique-growth». In: *Distributed and Parallel Databases* 38 (June 2020). DOI: 10.1007/s10619-019-07278-7 (cit. on pp. 11, 20).

[9] Witold Andrzejewski and Pawel Boinski. «Efficient Spatial Co-location Pattern Mining on Multiple GPUs». In: *Expert Systems with Applications* 93 (Oct. 2017). DOI: 10.1016/j.eswa.2017.10.025 (cit. on pp. 11, 17).

[10] Arpan Man Sainju, Danial Aghajarian, Zhe Jiang, and Sushil Prasad. «Parallel Grid-Based Colocation Mining Algorithms on GPUs for Big Spatial Event Data». In: *IEEE Transactions on Big Data* 6.1 (2020), pp. 107–118. DOI: 10.1109/TBDATA.2018.2871062 (cit. on pp. 11, 17, 18).

[11] Yong Ge, Zijun Yao, and Huayu Li. «Computing Co-Location Patterns in Spatial Data with Extended Objects: A Scalable Buffer-Based Approach». In: *IEEE Transactions on Knowledge and Data Engineering* 33.2 (2021), pp. 401–414. DOI: 10.1109/TKDE.2019.2930598 (cit. on pp. 11, 12).

[12] Xiaojing Yao, Ling Peng, and Tianhe Chi. «A co-location pattern-mining algorithm with a density-weighted distance thresholding consideration». In: *Information Sciences* 396 (Feb. 2017). DOI: 10.1016/j.ins.2017.02.040 (cit. on p. 13).

[13] Peizhong Yang, Lizhen Wang, Xiaoxuan Wang, and Lihua Zhou. «Efficient discovery of co-location patterns from massive spatial datasets with or without rare features». In: *Knowledge and Information Systems* 63 (June 2021). DOI: 10.1007/s10115-021-01559-3 (cit. on p. 14).

[14] Harry Kai-Ho Chan, Cheng Long, Da Yan, and Raymond Chi-Wing Wong. «Fraction-Score: A New Support Measure for Co-location Pattern Mining». In: (2019), pp. 1514–1525. DOI: 10.1109/ICDE.2019.00136 (cit. on p. 14).

[15] Peizhong Yang, Lizhen Wang, Xiaoxuan Wang, and Lihua Zhou. «SCPM-CR: A Novel Method for Spatial Co-Location Pattern Mining With Coupling Relation Consideration». In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2022), pp. 5979–5992. DOI: 10.1109/TKDE.2021.3060119 (cit. on p. 15).

[16] Xuguang Bao, Gu Tianlong, Liang Chang, Zhoubo Xu, and Long Li. «Knowledge-Based Interactive Postmining of User-Preferred Co-Location Patterns Using Ontologies». In: *IEEE Transactions on Cybernetics* PP (Mar. 2021), pp. 1–14. DOI: 10.1109/TCYB.2021.3054923 (cit. on p. 16).

[17] Xiangye Xiao, Xing Xie, Qiong Luo, and Wei-Ying Ma. «Density based co-location pattern discovery». In: Nov. 2008, p. 29. DOI: 10.1145/1463434.1463471 (cit. on p. 17).

[18] M. Prasad and Venkata Jarugumalli. «Performance Evaluation: Ball-Tree and KD-Tree in the Context of MST». In: vol. 62. Oct. 2012. ISBN: 978-3-642-32572-4. DOI: 10.1007/978-3-642-32573-1_38 (cit. on p. 30).

[19] https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940. In: (cit. on p. 31).