MINING PERIODIC PATTERNS IN SPATIO-TEMPORAL SEQUENCES AT DIFFERENT TIME GRANULARITIES

by

Sezin Karlı

Submitted to the Graduate School of Engineering and Natural Science in partial fulfillment of the requirements for the degree of Master of Science
Sabancı University
July 2007

MINING PERIODIC PATTERNS IN SPATIO-TEMPORAL SEQUENCES AT DIFFERENT TIME GRANULARITIES

Sezin Karlı - Computer Science and Engineering, Master Of Science Thesis, 2007

Thesis Supervisor: Asst. Prof. Yücel SAYGIN
Keywords: Data mining, spatio-temporal data, time granularity, periodic pattern

ABSTRACT

With the advancement of technology, it is now easy to collect the location information of mobile users over time. Spatio-temporal data mining techniques were proposed in the literature for the extraction of patterns from spatio-temporal data. However, current techniques can only produce patterns at the finest time granularity, and therefore overlooks potential patterns available at coarser time granularities. In this work, we propose several techniques to allow mining at different time granularities. Experimental results show that the proposed techniques are indeed effective and efficient for mining periodic spatio-temporal patterns at different time granularities.

ZAMAN-MEKAN DİZİLERİ ÜZERİNDE ÇEŞİTLİ ZAMAN TANECİKLİLİKLERİNDEKİ PERİYODİK DESENLERİN ÇIKARTILMASI

Sezin Karlı - Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2007

Tez Danışmanı: Yard. Doç. Yücel SAYGIN
Anahtar Kelimeler: Veri madenciliği, zaman-mekan verileri, zaman tanecikliliği, periyodik desen

ÖZET

Teknolojinin gelişimiyle, hareket eden kişilerin yer bilgilerinin toplanması oldukça kolaylaşmıştır. Literatürdeki zaman-mekan veri madenciliği teknikleri zaman-mekan verilerindeki desenlerin yakalanması için metodlar sunmaktadırlar, ancak işbu teknikler sadece en alt zaman tanecikliliğine ait desenleri bulabilmektedirlerdir. Bu kısıtın sorunu üst zaman tanecikliliklerinde yeralma olasılığı olan desenlerin gözden kaçırılması ve yakalanamamasıdır. Bu tezde çeşitli zaman tanecikliliklerinde desen yakalamayı sağlayacak teknikler sunulmaktadır. Yürüttüğümüz deney sonuçlarına göre önerdiğimiz teknikler desenleri etkin biçimde bulmakta ve bu süreci verimli gerçekleştirmektedirler.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor Dr. Yücel Saygın for his guidance, encouragement throughout this thesis. I am indebted to Dr. Hüsnü Yenigün, Dr. Aytül Erçil, Dr. Tuğkan Batu and Dr. Albert Levi for helping me with their precious advices. I am thankful to Dr. Huiping Cao for sharing his data generator, and for answering my questions patiently and to Saygın Topkaya for his advices on implementation. I am grateful to all of my friends and to my family for their support.

Finally, I would like to express my deepest gratitude to my father for encouraging me to the master program and to Onur Arıkan for his advice in the same matter.

Without them, this thesis was not possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

Our daily lives contain several routines. Some of these routines can be visits done to places such as our favorite restaurant / pub or our work / home and so on. These visiting habits are generally available for most of the moving entities. The trajectories of vehicles or the immigration patterns of animals are examples of these travel routines.

Travel routines generally exhibit periodic behavior. For example, we go to our favorite pub at every Friday night or we come back home from work everyday approximately at the same time or a certain bus visits a bus stop with intervals of half an hour. The natural periodicity of these patterns makes the task of periodic pattern mining interesting and this discovery leads us to an important question: "Can real life situations be modeled with partial periodicity or full periodicity?". As another example, let's consider a single day of Tom who wakes up at 7 o'clock, leaves home at 8 o'clock and arrives at work at 9 o'clock. He sometimes eats at Boston Restaurant, sometimes at Scholtz's Place and sometimes skips lunch and works instead. Tom's pattern occurs most of the days and it is better modeled with a partial periodic pattern as opposed to a fully periodic pattern since he skips lunch once in a while or eats at different restaurants.

In the previous example, we considered patterns based on *hour*, which is an intuitive time granularity. The real life examples show that mining at coarser time granularities (such as "*day*" or "*week*") is also important because mining at coarser granularities can reveal patterns that can not be discovered otherwise. Let's consider another person –Brad– who visits his parents living in France in approximately same time of the year for a week. It is probable that this visit won't contain frequent patterns in finer granularities because, for example, Brad won't visit Notre Dame de Paris for 5 days of the week at the same hour or he won't always eat at the same place at the same hour. Even if there was a frequent periodic pattern in the finer time granularity (such as *hour*), we would miss it because it will occur during only one week of the whole year (i.e. it has a very low frequency). On the other hand, if we did the mining with granularity of *week* and with the optimal period, then we would realize that there is a recurring visit to Paris.

Motivated by examples such as the previous one, we propose techniques that can mine periodic patterns at different time granularities. In this thesis, we work with spatio-temporal sequence of a single object. Moving from a time granularity g_1 to a coarser time granularity g_2 is trivial assuming that conversion from g_1 to g_2 is possible (every time component of g_1 must be contained in a unique time component of g_2): We will map several time components of g_1 to a single time component of g_2. Since there are location measurements associated with each time component of a granularity, the mapping from g_1 to g_2 will force us to a similar many-to-one mapping of locations. We choose to map several locations to a single discretized representation of these locations. Notice that during the discretization process, we are interested only in the spatial information the dataset contains. This choice has a logical argument behind it. In our daily life, moving to the coarser time granularities has the effect of omitting uninteresting details related to the finer time granularities. For instance, when we talk about Rick and Nielsen's visit to Topkapı Palace, we are concisely saying "Rick and Nielsen visited Topkapı Palace on Monday". This statement does not use the finest granularity although, for instance,"*second*" or "*minute*" were available and it obviously does not contain at which exact time interval they did this visit, because our intention in using the *day* granularity was to disregard these details. So we use only the spatial information contained in the dataset.

Five techniques which use different types of discrete representations are proposed in our work:

1. MINOR - periodic pattern MINer using minimum bOunding Rectangles which are generated with important places information
2. PAMUC - Periodic pAttern Miner Using Convex hulls which are generated with important places information
3. MINIM - periodic pattern MINer using exact IMportant places information
4. μ-PIN - periodic pattern Miner Using approximate important Places INformation
5. MAP - periodic pattern Miner using Approximate and numerical important Places information

All proposed techniques make use of the "important place" concept which is defined in Chapter 3. MINOR, PAMUC and MINIM do exact matching of important place contents while μ-PIN and MAP use the "similar" matching.

Experimental results show that the proposed techniques are accurate and efficient. As MINOR, PAMUC and MINIM are designed with the same purpose in mind (exact matching of important place contents), we can compare them without hesitation. Experiments show that MINIM is the best technique that does exact-matching of discrete representations for its superior efficiency, effectiveness and compactness of discrete representations. μPIN and MAP can't be compared with other techniques or with each other because their purpose in the periodic pattern mining differs from other techniques we propose. To best our knowledge, the proposed techniques are the first ones in the domain.

Chapter 2 contains the necessary background information about clustering algorithms and related work in the domain. Chapter 3 contains the definitions of time related concepts, preliminary definitions and the problem definition. The mining of periodic patterns at different time granularities is explained in Chapter 4. Chapter 5 contains the conducted experiments and the last chapter is the conclusion. Appendix contains the metricity proof of our geometry comparison metric.

Chapter 2
BACKGROUND AND RELATED WORK

The purpose of this chapter is to familiarize the reader with the clustering algorithms we used in our work and to give a general idea about why other popular algorithms are not preferred. The taxonomy of clustering algorithms from [18] is adopted. Furthermore, related work in the literature is given in this chapter.

2.1 Clustering Techniques

Clustering is the task of grouping similar objects such that we obtain high intraclass similarity and high interclass dissimilarity. DBSCAN [12] and AGNES [22] are the clustering algorithms used in our techniques. DBSCAN is used during the extraction of important places phase where we discover important places by clustering location measurements and treating the resulting clusters as important places. Notice that as we work on two dimensional Euclidian space, our location measurements are tuples with numeric values. AGNES is used for grouping similar geometries which represent similar visits to important places. AGNES uses our proposed geometry comparison metric (which can be seen in Subsection 4.1.3) for this task. Furthermore, we apply AGNES for grouping similar bit vectors (using our binary dissimilarity measure) and for grouping similar sequences of tuples (using Euclidian distance). Bit vectors and sequences of tuples contain features discovered from important places. A bit vector contains existence or non-existence of visit to important places and a sequence of tuples contains number of readings and visits measured in important places.

In this part of the thesis, we choose to analyze three major families of clustering:

1. Partitioning Methods
2. Hierarchical Methods
3. Density-based Methods

2.1.1 Partitioning Methods

Partitioning methods group data of size n into k ($k < n$) clusters such that every single data object will belong to a single cluster. Two kinds of heuristics are used:

1. K-means which define cluster center as mean

2. K-medoids which define cluster center as a center object

K-means

One of the major problems with K-means [24] is the fact that it can work only with data of numeric attributes. In “extraction of important places” phase our values are numeric, but for instance while comparing geometries it is not obvious what we will use as the cluster center. The most trivial approach is the usage of the mean of geometry centroids as the cluster center but it is easy to realize that for calculating the squared error we need a geometry as the cluster center not a single point. If we use a single point as a cluster center then the distance between a single point and a geometry will be huge using our geometry comparison metric.

Another problem is due to the structure of bit vectors (which are important place contents of different location sets). Finding a straightforward definition for the centers of clusters containing bit vectors is difficult.

Another major problem with K-means is the fact that we should fix a “k” value before running the algorithm which is completely unknown in our cases. Neither in “extraction of important places”, nor in clustering of geometries / bit vectors / sequences of tuples, we have an idea about the number of clusters.

The fact that K-means creates clusters of spherical shape can hurt our performance in “extraction of important places” phase. We doubt that every single building occupies a terrain of a circular shape. Furthermore, clusters of different size is possible in our work and K-means is weak towards that kind of grouping. For instance, Brad is a golf addict and one of his important places is a golf club. Furthermore, his house (which is an another important place) occupies only $120m^2$ of area. Then there are two important places with very much difference in the area they occupy which will give clusters of bad quality with K-means.

The definition of K-means is against our desire of not including noise/outliers into clusters. It tries to include every single data point into a cluster which in result will make our “extraction of important places” impossible. If every point will be included into a cluster, how we will detect noise and outliers?

PAM

PAM (Partitioning Around Medoids [22]) uses the k-medoids approach.

Using a data object as a cluster center will relieve problems of K-means while we cluster geometries or bit vectors. As the cluster centers are objects in K-medoids approach, it will be straightforward to use them instead of extending the current algorithm.

Furthermore, the fact that PAM will include every single object to a cluster will make its use illogical in the "extraction of important place" phase just like in the case of K-means.

The need for a *k* value is still a problem that needs to be solved. Furthermore, the clusters will be of spherical shape just like in the case of K-means which is problematic as previously said.

CLARA

CLARA (Clustering LARge Applications [22]) uses the k-medoids approach.

CLARA includes every single object to a cluster so it is impossible to use it in "extraction of important places", because of the arguments we give above.

The need for a *k* value is a problem just like in the case of K-means and PAM.

A special weakness of CLARA is that if during the sampling phase one or more "good" medoids are omitted, then it will be nearly impossible to obtain a clustering of good quality.

CLARANS

CLARANS (Clustering Large Applications based upon RANdomized Search [27]) is one of the best k-medoids based methods.

Although it won't be an efficient choice, it is possible to run CLARANS for *k* values from 2 to *n* and calculate "silhouette coefficient" [22] for finding the most natural clustering obtainable by CLARANS. So *k* value is not a problem anymore if we accept its inefficiency shortcoming.

Numerous experiments in [12] show that even with the optimal *k* value, the clustering quality is much inferior to the quality of DBSCAN which cancels the possibility of using CLARANS for "extraction of important places". Furthermore, as there is no concept of noise, all data points will be included to a cluster which is not the desired effect.

X-means

X-means [30] is one of the best extensions of K-means which does not need a *k* parameter.

The problem with X-means is that it is usable only for data with numeric features. So it is problematic while comparing geometries / bit vectors of important place contents. Furthermore, it can't be used in "extraction of important places" because as there is no concept of noise, every single data point will be included into a cluster.

2.1.2 Hierarchical Methods

Hierarchical methods partitions the data in a hierarchical way (forming a dendogram). Top-down and bottom-up approaches are possible. Top-down has a single cluster that contains every single data element as the initial clustering and it advances by dividing the large cluster into dissimilar parts. Bottom-up is a progress in the reverse direction.

AGNES

AGglomerative NESting [22] is a bottom-up hierarchical clustering algorithm. The initial clustering is of size N where every single sample has its own cluster. Distances between every pair of cluster is calculated and then the merging phase begins. Similar clusters are merged and the algorithm continues this merge operation until the stopping criteria or the desired cluster number is reached.

The similarity between clusters is calculated by a distance function. The beauty of AGNES is that it allows the usage of any distance function without needing any extension to its algorithm. Any distance metric can be adopted without different complications created by its usage. One of the main reasons AGNES is used in our clustering of geometries / bit vectors is this fact.

Cluster distances are calculated by a distance function, but normally clusters contain more than one sample. As the distance functions offer only the distances between sample pairs, we will surely need a linkage metric ([26], [28]) to calculate the distance between two clusters.

There are three widely used linkage metrics: d_{min}, d_{max} and d_{avg}.

- $d_{min}(C_i, C_j) = min_{x \in C_i, y \in C_j}\{distance(x, y)\}$
- $d_{max}(C_i, C_j) = max_{x \in C_i, y \in C_j}\{distance(x, y)\}$
- $d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} distance(x, y)$

where C_i and C_j are two different clusters and $distance()$ is the distance function we choose.

d_{min} (single linkage) is not generally preferred because of its chaining effect. Single linkage cause clusters to be merged in the presence of only one very similar sample pair which results in this “chaining phenomenon”.

d_{max} (complete linkage) works better than d_{min} generally because of the absence of chaining phenomenon, but it is vulnerable to outliers.

d_{avg} (average linkage) is like a balance between single linkage and complete linkage.

As after the merging phase it is impossible to reverse this process or swap cluster contents, the clustering quality can deteriorate. Our experiments did not show this kind of weakness but it is a possibility to take into consideration.

To sum up, the power of AGNES resides in its ease of using the desired distance function without needing any extension to the original algorithm and its need to a single parameter (stopping criteria). Our experiments show that stopping criteria value can be set intuitively and does not ask for much time for the grid search. Setting it intuitively is not possible in the clustering of bit vectors and there, an algebraic hint is offered to the user to compensate this difficulty.

BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies [39]) is a popular clustering algorithm.

CF tree used in BIRCH is designed to handle numeric data so comparing geometries and bit vectors will be problematic because the lack of trivial centroid definition. Furthermore, because of the CF tree parameters, it is a possibility that we don’t obtain natural clustering which is why BIRCH is not used in “extraction of important places”.

Furthermore, it is known that BIRCH has problems handling non-spherical data which makes it again a bad choice for the “extraction of important places” phase.

CURE

CURE (Clustering Using REpresentatives [13]) is another well-known hierarchical clustering algorithm.

Its robustness on non-spherical shapes makes it a good candidate for the “extraction of important places” phase. But the fact that there are two parameters (shrinking factor and number of representative points) which are not easily set is a negative point. Furthermore, both parameters have a large impact on the final clustering results which in result makes the usage of CURE impossible in any of our techniques.

Chameleon

Chameleon [21] is a powerful clustering algorithm that is proved to offer more natural clustering results than DBSCAN. It can capture arbitrary shaped clusters which is really important in our “extraction of important places” phase.

The problem with Chameleon is that its time complexity is $O(n^2)$ compared to the $O(n \log n)$ of DBSCAN (with a spatial index structure such as R-tree [14] and R*-tree [3]) and as we will use it for a large number of samples it is not a good choice for our work. In our techniques, we apply AGNES which has a complexity of $O(n^2)$ for n objects. But the number of objects AGNES works on is much smaller than the number of objects Chameleon will work on, so we opt for the usage of DBSCAN.

2.1.3 Density-based Methods

Density-based methods treats dense regions in the data space as clusters. Data subsets that are labeled as outlier/noise are found generally between these dense regions and outliers/noise are not included into any cluster.

DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise [12]) is a widely known density-based clustering technique. We need to do few definitions before explaining its working scheme. DBSCAN needs two input parameters: EPS and $MinPTS$.

1. EPS neighborhood of a sample x is defined as
   $N_\epsilon(x) = \{y \in D \mid distance(x, y) \leq \epsilon\}$ where D is the whole data set.

2. A sample x is a “core object” if $|N_\epsilon(x)| > MinPTS$.

3. A sample x is “directly density-reachable” from a sample y if $x \in N_\epsilon(y)$ and y is a core object.

4. A sample x is “density reachable” from a sample y if there is a sequence of samples $p_1, ..., p_q$ such that “p_{i+1} is directly density-reachable from p_i” for $0 < i < q$ and $p_1 = y$, $p_q = x$.

5. A sample x is density-connected to y if there is a sample t ($t \in D$ and $t \neq x, t \neq y$) such that both x and y are density-reachable from t.

6. A density-based cluster C is a set of samples such that

- If $p \in C$ and q is density-reachable from p, then $q \in C$ (where q and p are two samples)
- Every element p of C is density-connected to every element q of C.

Density reachability is symmetric for only the case where x (the first element of the sequence) and y (the last element of the sequence) are both core objects otherwise it is not. On the other hand, density-connectivity relation is symmetric.

DBSCAN takes a single sample x, checks if it is a core object. If it is, then DBSCAN iteratively finds all density-reachable samples from x. This process repeats itself for every $x \in D$. If a previously labeled cluster element is encountered during a new density-reachability search both clusters will be merged into one.

After the run of the algorithm is complete, there will be;

1. Core objects (essential parts of the density-based clusters)
2. Non-core objects belonging to density-based clusters which are in fact boundaries of these clusters
3. Non-core objects not included in any density-based cluster (tagged as noise / outlier)

The problem with DBSCAN is its usage of global parameters. It is a possibility that different parts of D need different EPS and $MinPTS$ parameters to work with. We did not encounter that type of vulnerability during our "extraction of important places" phase but as we worked on synthetic data, this result is far from surprising.

Another weakness of DBSCAN is its sensitivity towards the parameters EPS and $MinPTS$. We propose a preprocessing technique (elimination of high speed data which can be consulted at Subsection 4.1.2) that can remarkably remove this sensitivity.

DBSCAN's beauty lies on its success in creating natural clusterings. Furthermore, it is powerful in handling arbitrary shapes which is very important in our task. Its time complexity is only $O(n\ logn)$ with a spatial index.

Setting EPS and $MinPTS$ parameters is not counterintuitive, because we work on two dimensional space. On the other hand, it could be difficult to set them while clustering geometries / bit vectors / sequences of tuples which is why AGNES is preferred to DBSCAN for these tasks.

OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure [1]) is an extension of DBSCAN.

We recently talked about the potential difficulty in setting *EPS* and *MinPTS* values in DBSCAN. OPTICS can offer an interactive clustering medium (like "reachability plot") and can make the choice of parameters easier. Reachability plot contains information that is equivalent to DBSCAN run with a large range of parameters.

In our work, it was possible to use OPTICS in the first run and to find the optimal parameters. So, for subsequent runs for coarser granularities, we could find the new parameters by taking optimal parameters of the first run into consideration. We did not need OPTICS because we were working on synthetic data set, and it was easy to find the optimal parameters from the data set.

2.2 Related Work

Extensive research has been conducted on periodic pattern mining. Authors of [29] propose a solution for mining association rules that occur at specific time periods such as first week of every month. Han et. al. [16] propose algorithms for mining partial periodic patterns from time series data. In [17], the ideas proposed in [16] are further extended and an efficient and scalable algorithm that uses a novel tree structure (max-subpattern tree) is proposed. First, the frequent 1-patterns –which are basically patterns found in a single position with frequency of occurrence more than the threshold– are found and then the root of the tree is prepared with this information. After that, the whole discrete sequence is inserted to the max-subpattern tree and the count of the node corresponding to the segment of the tree is incremented. Then the traversal begins and the frequent patterns are extracted from the tree.

In [36], a technique for mining periodic patterns from event sequences is proposed which was later extended in [37] for allowing mining partial periodic patterns. In [10] and [11], authors propose methods to detect periodicity in the event sequences by using convolution-like formulas which were not considered in earlier studies.

The first work in mining periodic patterns in spatio-temporal sequences is [7] where authors offer a discretization method for location information and then use an extended form of the technique proposed in [17] for doing the mining in the finest time granularity. The problem with this approach is that it overlooks patterns at coarser time granularities and to the best of our knowledge our work is the first one attacking this problem.

In our work, we choose to use the concept of "important places" for obtaining a concise representation from several location measurements. The discovery of important places is an ongoing research area. Authors of [25] developed a system specific to GPS data. They infer that a place is important if there are several signal losses in approximately same area. They assume that a signal loss shows that the object is in a building, but they miss the fact that it is possible to have signal losses without a stay in a building (such as in the case where the battery of GPS device is low or a disconnection from the satellite). Furthermore, open area places will be missed with this approach. In [2], the time information is omitted from the spatio-temporal sequence and a variant of k-means is applied to the spatial data. The approach of [2] is outperformed by the approach proposed in [40] where authors apply a density-based clustering algorithm (DJ-Cluster) to the spatial data for the extraction of important places. The importance of [40] is that the authors offer metrics for the evaluation of performance in the extraction of important places. Notice that studies before [40] are not evaluated for their performance of important places discovery. [41] contains experiments conducted on 24 people of different life stages using the system in [40]. In our work, we use a density-based clustering method and respect periodicity and time information of coarser granularity while doing the extraction of important places.

Chapter 3
PROBLEM FORMULATION

In this chapter, we formulate the problem what we choose to address. First, the definitions pertaining to temporal concepts will be defined. Then, the preliminary definitions will be given and after that the problem will be defined.

3.1 Temporal Concept Definitions

In this work, we adopt the temporal concepts defined in [5]. We will use the set *time domain* (denoted as $(R, \leq)$ where R is the set of real numbers and $\leq$ is a total order on R) as the set of primitive temporal entities that is used to define temporal concepts. R is used as the set of time instants in the time domain $(R, \leq)$.

Definition 3.1.1 *A time granularity g is a mapping from the set of non-negative integers (the time ticks) to 2^R (subsets of the time domain) that satisfies the following conditions for all positive integers i, j such that $i < j$:*

1. *If $g(i)$ and $g(j)$ are both non-empty, then each element in $g(i)$ is less than any element in $g(j)$.*
2. *If $g(i)$ is an empty set, then $g(j)$ must be an empty set too.*

Let's see the first property with an example.

Example 3.1.1 *Let's assume that, we are using $year\_since\_1900$ granularity. All elements in $year\_since\_1900(0)$ will be less than $year\_since\_1900(1)$ because every single time element in year 1900 ($year\_since\_1900(0)$) will be less than every single element in year 1901 ($year\_since\_1900(1)$).*

Frequently used (and intuitive) time granularities such as *hour*, *day*, *week*, *month*, *year* all satisfy the above conditions. When working with time granularities, we will need a bottom granularity which requires a temporal relationship such as "finer than".

Definition 3.1.2 *A granularity g is finer than a granularity h if for each index of g, there is an index j such that $g(i) \subseteq h(j)$.*

For example, *hour* is finer than *day* and *month* is finer than *year*. "Finer than" relationship gives us the base for the bottom granularity definition.

Definition 3.1.3 *Given a granularity relation $\prec$ and a set of granularities defined with the same time domain, a granularity g in the set is the bottom granularity with respect to $\prec$, if $g \prec h$ for every granularity h in the set.*

Example 3.1.2 *In $\{minute, hour, day, week, month, year\}$ set and "finer than" being the temporal relationship, we can define minute as the bottom granularity because minute is finer than any time granularity in this set.*

Definition 3.1.4 *A tick of a granularity g is a nonempty subset $g(i)$ where i is its index. The terms "tick of the bottom granularity" and "timestamp" will be used interchangeably.*

Bottom granularity and tick definitions will be useful in granularity conversions. Every tick z of the bottom granularity can be mapped to a unique tick z' of one of the granularities in the time granularity set. $\lceil z \rceil_g^h = z'$ is the conversion operator where g and h are both time granularities and $g(z) \subseteq h(z')$. For instance, $\lceil 2 \rceil_{minute}^{year}$ will return the year value that contains the second minute.

As the time-related definitions are complete, we will now give preliminary definitions that will be needed throughout the thesis.

3.2 Preliminary Definitions

Our task is to mine periodic patterns in spatio-temporal data at different time granularities. In general, this means that we will move to a coarser time granularity than the bottom one and do the mining at this granularity to extract previously unknown patterns.

The sequence of location-timestamp pairs is denoted by $S = \{(l_0, t_0), (l_1, t_1), (l_2, t_2), ..., (l_{n-1}, t_{n-1})\}$ where t_i is the timestamp and l_i is the location component corresponding to the timestamp. For example, this sequence can be Bob's traced movement during 2006; $S_{bob} = \{((1,3),0),\ ((1,4),1),\ ((2,5),2),\ ((3,6),3)...\}$. At the zeroth timestamp, Bob was at location $(1,3)$, at the first one he was at location $(1,4)$ so on.

From now on, we will use S as the abbreviation of the spatio-temporal sequence of the bottom granularity. Furthermore, we will omit the bottom granularity information in $\lceil\rceil$ operator as in [6]. In addition, we will use the expression "coarser granularity" instead of "a new granularity coarser than the bottom granularity".

Definition 3.2.1 *For a coarser time granularity g, the time set with index k is defined as* $TS_k^g = \{t_i, t_{i+1}, ..., t_j\}$ *such that* $\lceil t_i \rceil^g = \lceil t_{i+1} \rceil^g = \lceil t_{i+2} \rceil^g = ... = \lceil t_j \rceil^g = k$ *and it does not exist a timestamp* $t' \notin TS_k^g$ *such that* $\lceil t' \rceil^g = k$.

Example 3.2.1 *If g is day and location measurements are made every hour in S* ($t_{i+1} - t_i = 1$ *where t is timestamp), then* TS_2^{day} *will contain all the timestamps contained in the second day which will be equal to* 24 *timestamps here. Notice that the index of the time set begins from* 0 *just like the index in the time granularities.*

We assume that there are no missing location measurements for timestamps that are in S, so it is possible to denote the location measurement corresponding to a timestamp t_i by l_i.

Definition 3.2.2 *For a coarser time granularity g, the location set of index k is defined as* $LS_k^g = \bigcup_{\forall i \ of \ t_i \in TS_k^g} l_i$.

Example 3.2.2 *If we continue from the previous example, we will have* LS_2^{day} *equal to the set of location measurements belonging to second group of* 24 *timestamps (beginning from timestamp* 48 *and ending in timestamp* 71*) contained in S. Notice that* LS_0^{day} *is the zeroth group of* 24 *timestamps.*

Definition 3.2.3 *Let T be the mining period and g be the coarser granularity. The set that groups all location sets of position p of the period is defined as* $L_p^g = \bigcup LS_i^g$ *for all i such that* $i \bmod T = p$.

Example 3.2.3 *Assuming that the period is* 7, $L_2^g = LS_2^g \cup LS_9^g \cup ... \cup LS_j^g$ *where j is the maximum i of the spatio-temporal sequence that complies with* $i \bmod 7 = 2$.

3.3 Problem Definition

Given a minimum support value, *min_sup* $\in [0, 1]$, a sequence of location-timestamp pairs S, a period T and a time granularity g, our problem is to discover patterns that repeat themselves with the period of T time ticks in time granularity g with a frequency greater than the *min_sup* value. Notice that, three symbols above (S, T, g) will be used throughout the thesis as the abbreviations to their definitions above.

Time information of the bottom granularity will be used for slicing S into location sets. After that, this time information won't be needed. For instance, if we are working on granularity g, we will first build time sets of granularity g from S. Later, we will derive location sets corresponding to these time sets and then S will be turned into a sequence of location sets.

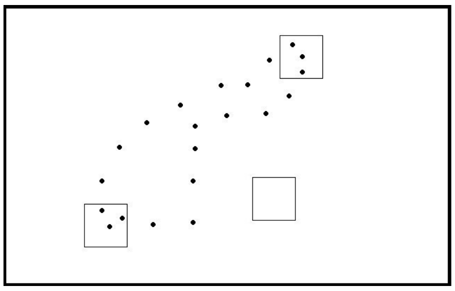

Figure 3.1: Location points of the location set

Example 3.3.1 *Assume that our sequence is* $S = \{((0,0),11),\ ((1,2),23),\ ((2,3),35),\ ((2,4),47),((3,5),59),\ ((3,7),71),\ ((6,7),83),\ ((6,8),95)\}$ *where the bottom granularity is hour (i.e. one location measurement per* 12 *hours). In order to analyze* S *in granularity day, first, we will build time sets for the day granularity.* $TS_0^{day} = \{11,23\}$, $TS_1^{day} = \{35,47\}$, $TS_2^{day} = \{59,71\}$, $TS_3^{day} = \{83,95\}$ *will be obtained.* $LS_0^{day} = \{(0,0),\ (1,2)\}$, $LS_1^{day} = \{(2,3),\ (2,4)\}$, $LS_2^{day} = \{(3,5),\ (3,7)\}$, $LS_3^{day} = \{(6,7),\ (6,8)\}$ *will be extracted. This way, we transform* S *into a sequence of location sets* $LS_0^{day}\ LS_1^{day}\ LS_2^{day}\ LS_3^{day}$.

Definition 3.3.1 *Important places are regions where the traced object visits frequently and spends a fair amount of time.*

A discrete representation is the discretized form of the location sets obtained using the notion of important places. Three discrete representations of the location data are used in the proposed techniques: A discrete representation can be (i) a geometry, (ii) a bit vector, (iii) a sequence of tuples. We will briefly explain how these discrete representations describe the data. The details about the extraction of these discrete representations will be provided in Chapter 4.

Example 3.3.2 *Assume that we have a location set* LS_0^{day} *which is depicted in Figure 3.1. Three rectangles in the figure are highlighting the important places. First, we will omit the points that are not spatially contained in important places. Later, we will build discrete representations from the locations in hand.*

There are three types of discrete representations and the first type is geometric discrete representation. Two types of geometric discrete representations are proposed: (i) a minimum bounding rectangle (as in Figure 3.2) and (ii) a convex hull (as in Figure 3.3).

The second type of discrete representation is a bit vector where the contained binary values are separated with "," and delimited by "<" and ">" . This type of discrete representation is obtained by inspecting the existence and non-existence of visits to important

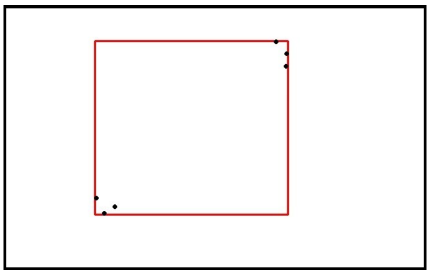
Figure 3.2: A rectangle generated using locations in important places

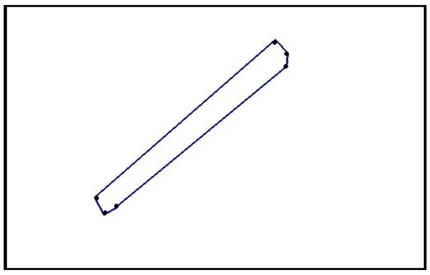
Figure 3.3: A convex hull generated using locations in important places

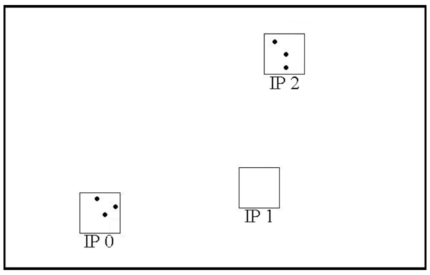

Figure 3.4: Visits to two important places with label 0 and 2

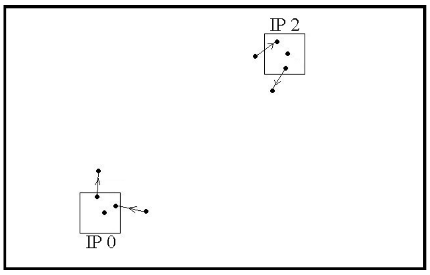

Figure 3.5: Three readings in and a single visit to two important places

places. Notice that every binary value in the bit vector represent a visit (represented by 1) or a lack of visit (represented by 0) to the important place. We will elaborate on this discrete representation in Chapter 4.

Example 3.3.3 *In Figure 3.4, we enumerated important places ($IP0$, $IP1$ and $IP2$). Then location set is represented with the bit vector $< 1, 0, 1 >$, because there is a visit to the zeroth (with index 0) and the second important place (with index 2), but the first important place (with index 1) is not visited.*

The last type of discrete representation is a sequence of tuples which captures the time spent in important places and the number of visits done to these important places. We use the notation $< (r_0, v_0), ..., (r_k, v_k) >$ where all elements in the sequence are integers and each tuple (r_i, v_i) denotes the readings and visits belonging to different important places.

For example, in Figure 3.5, you can see a single visit to the zeroth and the second important place and there are 3 readings in both of these places. The extracted discrete representation will be $< (3, 1),\ (0, 0),\ (3, 1) >$ (i.e. Three readings in the zeroth important place, zero readings in the first and three readings in the second important place. One visit to the zeroth important place, and one visit to the second important place and no visit to the first important place).

Similarity between all these discrete representation types is the fact that they allow

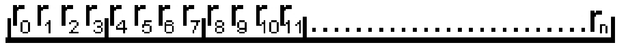
$r_0\, r_1\, r_2\, r_3\, r_4\, r_5\, r_6\, r_7\, r_8\, r_9\, r_{10}\, r_{11}$ .... r_n

Figure 3.6: Segments of S^g of period T

the usage of wildcard "$*$" which implies the possibility of any content for the discrete representation.

Example 3.3.4 *Assuming that we use geometries as discrete representations, "MBR_0 MBR_1 $*$" is a pattern of period* 3 *which presents that the object we are tracing spends time on region MBR_0 then MBR_1 and then "anywhere" on the map.*

We previously explained that S can be considered as a sequence of location sets. After a discrete representation is derived from each location set, a sequence of discrete representations denoted as S^g will be obtained. Notice that for each index in the location set sequence, we have a corresponding discrete representation with the same index (i.e. $LS_i^g = r_i$). Notice that r is the abbreviation of discrete representation.

Definition 3.3.2 *Segments (of S^g) are defined as a sequence "$r_{T\times i}$ $r_{T\times i+1}$ $r_{T\times i+2}$ .. $r_{T\times i+T-1}$" where $i = 0, 1, ..., (\lfloor \frac{j+1}{T} \rfloor - 1)$ assuming that r_j is the discrete representation corresponding to the last location set available for S in granularity g.*

For instance, in Figure 3.6 segments of a period of 4 can be inspected.

Definition 3.3.3 *Periodic pattern with period T is a sequence of T elements where an element can be a single discrete representation, a set of discrete representations, or a wildcard "$*$".*

Example 3.3.5 *For example, "$r_1\{r_2, r_4\}*$" is a periodic pattern of period* 3. *There is a discrete representation in the zeroth position, a set of discrete representations in the first position and "$*$" in the second position of the period.*

For a segment s (pattern p respectively), s_i (p_i respectively) is the ith position of s (p respectively). The meaning of a segment's compliance with a pattern changes with the technique used.

Definition 3.3.4 *In techniques that use geometric discrete representations, a segment s complies with a pattern p if the geometry of s_i is spatially contained in the geometry of p_i (for each i from* 0 *to* $T - 1$*).*

Definition 3.3.5 *In the first technique that uses bit vector as discrete representations, a segment s complies with a pattern p if bit vector in s_i is the same with bit vector of p_i for each $i = 0, 1, 2, ..., T - 1$. In the second technique that uses bit vector as discrete representations, a segment s complies with a pattern p if bit vector in s_i is "similar[1]" to bit vector of p_i for each $i = 0, 1, 2, ..., T - 1$.*

Definition 3.3.6 *In our technique that uses sequence of tuples as representation, a segment s complies with a pattern p if every "important place" in s_i has similar feature[2] values with p_i for each i from 0 to $T - 1$.*

Example 3.3.6 *Assume for a period of 2, we have a frequent pattern like "$<(3,2),(0,0),(1,1)><(2,1),(0,0),(0,0)>$". A segment s complies with the above pattern if s_0 has a similar feature set to "$<(3,2),(0,0),(1,1)>$" and s_1 has a similar feature set to "$<(2,1),(0,0),(0,0)>$".*

Definition 3.3.7 *A discrete representation set (DRS) is the set that groups all discrete representations of a single position of the period. Formally, $DRS^g_z = \bigcup_{i \bmod T = z} r_i$ for all i available in S^g sequence.*

Example 3.3.7 *Assume that our discrete representation sequence is $S^g = \{r_0, r_1, ..., r_{99}\}$ and our period (T) is 4. Then, $DRS^g_2 = \{r_2, r_6, r_{10}, ..., r_{98}\}$. Notice that all subscripts of discrete representations give 2 with " mod 4" operation.*

A periodic pattern's length is the count of discrete representations in it. For instance, "$r_t * r_c\, r_h *$" have period equal to 5, and length equal to 3.

Definition 3.3.8 *A periodic pattern of length k is called $k-pattern$.*

Definition 3.3.9 *p' is a subpattern of p, if they both have the same period T and either $p'_i \subseteq p_i$ or $p'_i = *$ for all i such that $0 \leq i < T$.*

Example 3.3.8 *If $p = r_1\{r_2, r_3\}*$, then p has six subpatterns such as "$r_1 * *$", "$*r_2*$", "$*r_3*$", "$r_1 r_2 *$", "$r_1 r_3 *$", "$*\{r_2, r_3\}*$".*

Subpatterns are more general than patterns. So the set of segments in the sequence S that comply with a certain pattern p will be a subset of the set of segments that comply with the subpatterns of p.

[1]The similarity concept will be defined later

[2]Selected features are the time spent on the important place and value of visits to this important place

Chapter 4
MINING OF THE MAXIMAL FREQUENT PATTERNS

In this chapter, we explain the working schema of our techniques. Mining of the maximal frequent patterns is done in two phases; the first phase consists of the mining of frequent 1-patterns and the second phase consists of the construction of a max-subpattern tree and the extraction of frequent nodes (patterns) from the tree. An illustration that describes the essence of all proposed mining techniques can be seen in Figure 4.1.

4.1 Finding Frequent 1-Patterns

As previously stated, patterns consist of discrete representations and these discrete representations change from technique to technique. We will begin this part of the thesis by explaining two steps that take place before the discretization process. These two steps are (i) elimination of points belonging to movement with high speed, (ii) extraction of important places and both of them are initial steps of all techniques.

4.1.1 Extraction of Important Places

We need the concept of important places to obtain better geometric discrete representations that don't contain redundant information such as the trajectories that are rarely taken and noise in the location data. If we don't omit outlier location points, they may have a negative effect on the discrete representation which in consequence will deteriorate our mining performance. For instance, Rick went to Istanbul only once in last 5 years, then this information can change the geometric discrete representation at every time granularity. Another importance of the extraction of important places is that this phase will give us the needed base for the extraction of discrete representations in our techniques which don't use geometric discrete representations.

One of the most interesting work for the extraction of important places is [40] as we previously declared. DJ-Cluster is used for clustering the spatial data and dense regions are treated as "important places". In this work, we partition the spatio-temporal sequence such that the resulting datasets will respect the time information of coarser granularity

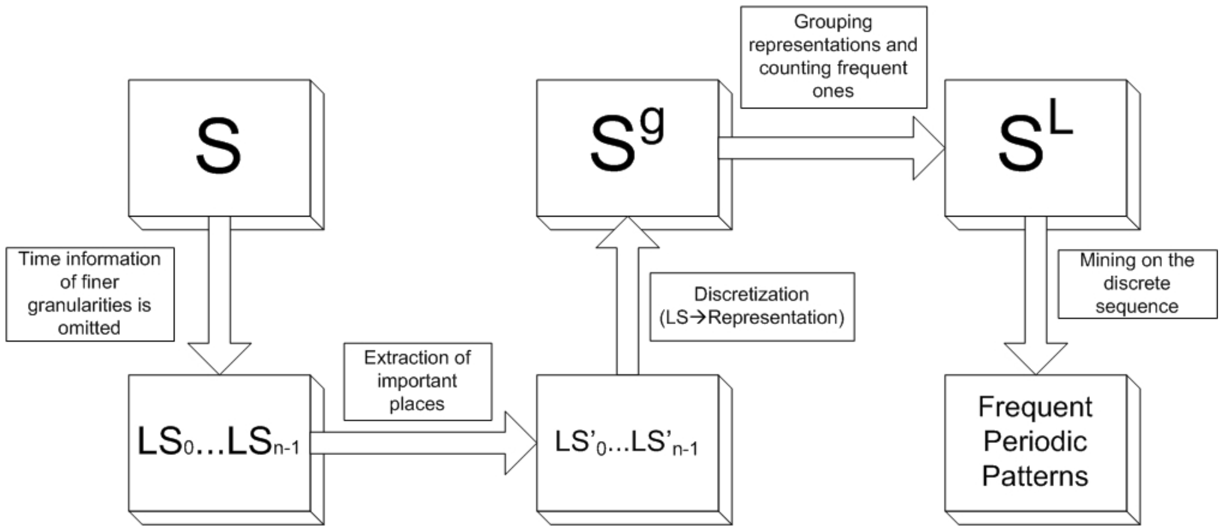
S
Time information of finer granularities is omitted
LS0...LSn-1
Extraction of important places
LS'0...LS'n-1
Discretization (LS→Representation)
Sg
Grouping representations and counting frequent ones
SL
Mining on the discrete sequence
Frequent Periodic Patterns

Figure 4.1: Illustration of the data mining process

and periodicity. After that, we apply DBSCAN instead of DJ-Cluster because there is no evidence in DJ-Cluster's superiority over DBSCAN in terms of clustering quality and the claim that DJ-Cluster is faster than DBSCAN is not proved by any kind of experiment.

Only two parameters ($MinPts$ and EPS) are needed by DBSCAN. $MinPts$ is the minimum number of objects that must be found within EPS distance of an object x for x to be a core object. Remember that the clusters in DBSCAN consist of core objects (such as x) and non-core objects which are reachable from core objects. Although the obtained results in [34] show that $MinPts$ can be easily fixed to $2k - 1$ for a data of k dimensions, that was not the case for us during our experiments. We claim that EPS can be easily set in our application due to the fact that we want to find buildings as clusters –which occupies an average amount of area on the map– but $MinPTS$ parameter has to be experimented. In our work, we propose a preprocessing method that reduces user errors that can occur in the $MinPTS$ selection. Reducing errors in parameter selection is important because DBSCAN is sensitive towards these parameters.

For finding important places, we apply $DBSCAN$ to each of L_i^g separately where $i = 0, ..., T - 1$ since we want to extract important places belonging to different positions of the period and to respect the time granularity. For example, Robin visits the shopping mall every Friday night. The shopping mall probably won't form a dense cluster if we consider all locations in S_{Robin}, because he was in this place for only few hours and for a single day in the whole week. Assume we mine at "day" granularity with a period of 7 which means that we want to find similar Mondays or Tuesdays... etc. In the Robin example, we will surely see that the shopping mall forms a dense cluster if we use L_4^{day} (location measurements of Fridays) to obtain important places, because the shopping mall will form a dense region because it is visited every single Friday. As the shopping mall forms a dense region, it will be treated as an important place just like all other dense regions.

Time Complexity Analysis 4.1.1 *For n objects, the time complexity of DBSCAN is $O(n \log n)$ with a spatial index such as R-tree. In the worst case, we're going to run DBSCAN with $\frac{N}{T}$ location measurements for T times where N is equal to $|S|$ and T is the period. Thus, the worst case complexity of this step is $T \times O(\frac{N}{T} \log \frac{N}{T}) = O(N \log \frac{N}{T})$. Notice that, in reality, we will have less location measurements in L_i^g sets than $\frac{N}{T}$ because there is a preprocessing step that will omit several locations of these sets before DBSCAN phase takes place.*

4.1.2 Elimination of High Speed Movement Data

During the extraction of important places, we use every single location measurement available in S. The problem with this approach is that we don't actually need a large number of location measurements. Eliminating the location measurements belonging to high speed movement and using only the ones with stationary-like tendencies will speed up our techniques and –most importantly– it will work as a safety net for the bad selection of parameters in DBSCAN. Assume that there is a traffic light in the road that the traced person's car frequently follows. He sometimes stops at the traffic light and sometimes does not. With a bad selection of $MinPts$ value, DBSCAN can't distinguish the difference between the densities of "home" and "traffic light" which means that it will mark both as important places. But if we apply our preprocessing, then the density near the traffic light will get low which in consequence can help DBSCAN in detecting that the traffic light is not a location as dense as "home" thus DBSCAN won't treat the traffic light as an important place.

We propose an algorithm for eliminating high speed movement data which basically finds two timestamps t_i and t_{i+2} with a single timestamp t_{i+1} between them and calculate the Euclidian distances $distance(l_i, l_{i+1})$ and $distance(l_{i+1}, l_{i+2})$. If these distances are both bigger than a threshold, then we can omit the location l_{i+1}. The fact that these distance are both bigger than a threshold means that between t_i and t_{i+2}, the object travels with a high enough speed to be eliminated. For instance, a person will not move with more than $40km/h$ speed inside an important place such as "home", "work", "golf club", "pub", so on. The proposed algorithm can be found at Algorithm 1. We define high speed (low speed) movement as a movement with speed larger (smaller) than the input threshold. We will now explain how we obtained the idea used in the algorithm with a case study. Notice that [40] applies a preprocessing step which omits l_{i+1} if $distance(l_i, l_{i+1}) > 0$. Our case study will reveal the problem of omitting l_{i+1} after the inspection of a single distance. Furthermore, using 0 as the threshold can be risky because there is no necessity that the traced object spends some time in an important place without moving. For instance, if Bob spends some time in the park, we can't be sure that he will sit on a bench. Maybe he runs during all his stay in the park.

Algorithm 1 Algorithm for elimination of high speed movement data (Input: The set of all locations D, $Threshold$ / Output: The new set of all locations D')

```
$dist1 \leftarrow distance(l_0, l_1)$
$dist2 \leftarrow distance(l_1, l_2)$
for i from 3 to $n$ do
    if ($dist1 > Threshold \ \wedge \ dist2 > Threshold$) then
        removeFromD($l_{i-2}$)
    end if
    $dist1 \leftarrow dist2$
    $dist2 \leftarrow distance(l_{i-1}, l_i)$
end for
```

After joining every pair of location measurements of consecutive timestamps with a line segment, and with the assumption that the difference between consecutive timestamps is fixed, we can name high speed movement (between consecutive timestamps) as "long segments" and low speed movement (between consecutive timestamps) as "short segments". For simplicity, we use only two segments in our case study where there are 4 different possibilities: (i) short segment after short segment (Figure 4.2), (ii) short segment after long segment (Figure 4.4), (iii) long segment after short segment(Figure 4.5), (iv) long segment after long segment (Figure 4.3).

Without loss of generality, we assume that movement happens from left to right in all cases. We want to omit the location measurement in the middle (denoted by 1) in this case study.

The first case is trivial since, if we omit the point in the middle, it is obvious that we decrease the density of an important place.

In the second case, we should not omit the point in the middle even though this point is part of the high speed movement (part of a long segment). Otherwise, the density of the important place (depicted as a box) can decrease and this decrease can cripple our accuracy. Notice that this is the case where the preprocessing in [40] is problematic.

In the second case, we saw that it is impossible to eliminate the end point of a long segment if there is a short segment after the long one. The third case is similar, but this time we can't eliminate the beginning point of a long segment (again a point in the middle) since it is preceded by a short segment. Otherwise, we can lose some density in

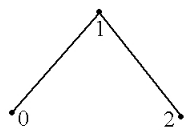

Figure 4.2: Short segment after a short segment

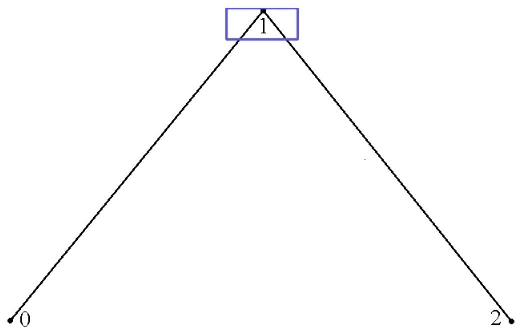

Figure 4.3: Long segment after a long segment

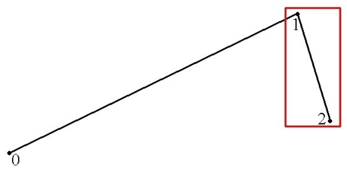

Figure 4.4: Short segment after a long segment

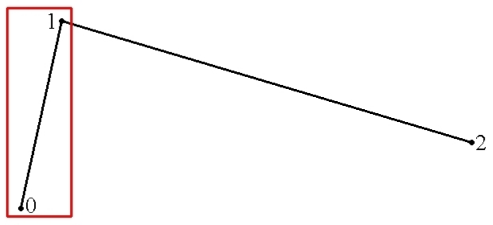

Figure 4.5: Long segment after a short segment

the important place (shown with a box).

In three previous cases, we see that we should not omit the middle point of two segments if at least one of them is a short segment. The fourth case is the only case that allows the elimination of location point in the middle without the risk of losing necessary location measurements. Since this point is not close to any other neighbor point, we know that it can't contribute to the density of an important place.

There is another issue to consider while we think about movements' speeds. In most of the studies about trajectories, it is generally seen that a linear interpolation is applied between two points belonging to two consecutive timestamps which is the shortest path that can be obtained using these points (Figure 4.6). So, the trajectory of the object will be built from location-timestamp pairs. This approach has an essential flaw; it is probable that between two consecutive timestamps a longer road like the one in the Figure 4.7 takes place. With the assumption of linear interpolation between consecutive points, it is possible that our algorithm misses some location points belonging to high speed movement. The reason is that a "short segment" can be in reality a path like the one in Figure 4.7. Two points which delimits this short segment will be treated like a part of movement with low speed and our method won't omit them. On the other hand, it is sure that any

Figure 4.6: The shortest path

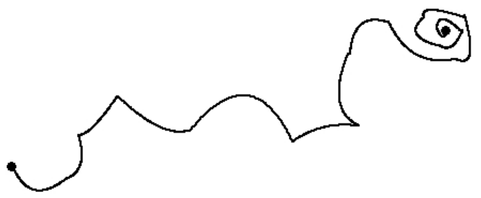
Figure 4.7: A possible (and long) path

point omitted by our method is a location point belonging to a high speed movement. As there is not a shorter path between two points than a straight line joining them, every long segment treated by our algorithm as a part of high speed movement is at least as long as our algorithm considered it in reality which implies that a long segment built with linear approximation always characterizes a "real" high speed movement.

Time Complexity Analysis 4.1.2 *Algorithm 1 has the time complexity of $O(N)$ where N is the number of location points.*

The more time the traced object spends during his travels with high speed, the more this phase is valuable certainly. The proposed preprocessing step won't be useful for mining the spatio-temporal data of someone who spends all his time on campus, because his spatio-temporal sequence will not contain any movement with high speed. We assume that generally the traced objects travel with high speed for a considerable amount of time which justifies our preprocessing step.

Let's note here that this step can be used in all proposed techniques but MAP. The reason is that as MAP has to calculate the visit frequency to important places, it needs location measurements that presents the entrance to an importance place and the exit from that same important place and our preprocessing step can omit these "entrance" and "exit" location measurements.

After the preprocessing step and the extraction of important places by using DB-SCAN on L_i^g set for each $i \in \{0, 1, ..., T-1\}$ separately, we obtain important places belonging to each i position of the period.

We talked about our desire on keeping important information in LS_i^g and omitting redundant location information. For each i of LS_i^g, we will remove all location $l_m \in LS_i^g$ if l_m is not spatially contained in any minimum bounding rectangles depicting the important places of position k such that $i \bmod T = k$. After that, the location sets (with their reduced content) will be ready to be discretized with discrete representations.

4.1.3 MINOR and PAMUC

After (i) turning S into a sequence of location sets ($LS_0^g\ LS_1^g\ ...\ LS_p^g$), (ii) applying preprocessing to the sum of these sets, (iii) omitting locations that are not spatially contained in important places and (iv) building a minimum bounding rectangle, or (iv) a convex hull from points contained in each location set, we will obtain a sequence such as $S^g = \{MBR_0, MBR_1, ...MBR_p\}$ (in MINOR) or $S^g = \{CH_0, CH_1, ...CH_p\}$ (in PAMUC) where each MBR_i is a minimum bounding rectangle that spatially contains LS_i and each CH_i is a convex hull that spatially contains LS_i. Notice that, the purpose of describing a location set by a geometry is that similar geometries imply similar visits to important places.

As we plan to find frequent 1-patterns, we have to count the frequency of discrete representations in S^g sequence. If their frequency is above the threshold min_sup, then they will be accepted as frequent . We have to do the counting on all DRS_i^g separately (i from 0 to $T - 1$) because we are after the frequent 1-patterns, so we have to respect the position of the period.

If we try to do exact matching of geometries during the counting phase, we may not be able to find any frequent 1-patterns since very similar geometries will be treated as if they are different geometries.

For grouping similar geometries, clustering methods can be used. Hierarchical clustering methods (like AGNES [22], DIANA [22]) with a distance metric designed for the comparison of geometries are chosen for this purpose.

Assuming that we have a period T, we will build DRS_i^g sets for each i from 0 to $T - 1$. For instance, if T is 3, then DRS_0^g will be $\{GEO_0, GEO_3, GEO_6, ..., GEO_m\}$ where m is the maximum value that is less than p and such that $m\ mod\ 3 = 0$.

The proposed distance metric is:

$$distance(GEO_i, GEO_j) = 1 - \frac{(Area(GEO_i \cap GEO_j))}{\max\{Area(GEO_i), Area(GEO_j)\}}$$

To best our knowledge, this is the first work proposing this metric. The proof of the metricity for the proposed metric can be found in the appendix.

A threshold value is used as a stopping criteria ($d(GEO_i, GEO_j) > Threshold$) to stop the clustering if the compared discrete representations are not as similar as desired. Threshold value must be in $(0, 1)$ interval. If the threshold is closer to 1, then geometries that are not too similar to each other will be treated as if they are and then they'll be merged. At the end, we will finish up with very few clusters with very large areas. If the threshold is too close to 0, then even very similar geometries will be treated as if they are actually different and we will obtain a large number of clusters with small areas. These facts show us that the choice of the threshold value is crucial. In our experiments, we run a grid search for obtaining the optimal threshold value and we obtain 0.25 and 0.35 as the optimal threshold value for the clustering of minimum bounding rectangles and convex hulls respectively.

Time Complexity Analysis 4.1.3 *For the construction of minimum bounding rectangles from location sets, the time complexity is $\frac{N}{k} \times O(k) = O(N)$ where $N = |S|$ assuming that preprocessing does not omit any location measurements and k is the number of locations contained in LS_i^g. Location sets are of size k, so we obtain $\frac{N}{k}$ sets from the whole dataset. Furthermore, $O(k)$ comes from the fact that for k location measurements, we need a single scan on them to obtain the minimum bounding rectangle.*

For the construction of convex hulls from location sets, the time complexity is $\frac{N}{k} \times O(k \ \log k) = O(N \ \log k) = O(N \ \log \frac{N}{T})$ where $N = |S|$ assuming that preprocessing does not omit any location measurements and k is the number of locations contained in LS_i^g. Notice that $k = O(\frac{N}{T})$ regardless of the bottom granularity and the coarser granularity we work on. Location sets are again of size k, so we obtain $\frac{N}{k}$ sets from the whole dataset. Furthermore, $O(k \ \log k)$ comes from the fact that for k location measurements, convex hull building algorithm works with $O(k \ \log k)$ complexity.

After the construction of geometries from location measurements in location sets, the clustering phase begins. Normally, AGNES runs with $O(n^2)$ for n samples. From the previous step, we know that we built $\frac{N}{k}$ geometries in total. Now, we split these geometries into T parts and apply AGNES separately to each part. So the time complexity of this step is $T \times O\left(\frac{N^2}{k^2T^2}\right) = O\left(\frac{N^2}{k^2T}\right) = O(T)$. Notice that the k parameter is dependent of the granularity of S and the granularity g that we currently work on, but $k = O(\frac{N}{T})$ always holds.

After the AGNES phase that is performed using different DRS_i^g (i from 0 to $T - 1$), we will find clusters with more than $min\_sup \times \left(\lfloor \frac{|S^g|}{T} \rfloor\right)$ geometries. From these largely populated clusters, we can easily find frequent 1-patterns. After we extract largely populated clusters, we have to choose a cluster representative that is similar to all of the elements inside the cluster. In MINOR, we use the minimum bounding rectangle created by merging all rectangles contained in the cluster as the cluster representative. In PAMUC, we use the convex hull created by merging all convex hulls contained in the cluster as the cluster representative. Later, we rename every discrete representation in the S^g sequence with a new label given to their cluster representative. If a geometry is not in a largely populated cluster, then its label is changed to "$*$" because this geometry can't be a frequent one. After this labeling operation, the new sequence S^L is ready to be mined. Notice that S^L is made only of labels which point discrete representations elected as cluster representatives or that have "$*$" as name. Obtaining frequent 1-patterns from the representatives is trivial. For example, for a representative with label l in the first position of the period where $T = 3$, "$* \ l \ *$" is a frequent 1-pattern.

4.1.4 MINIM, μPIN and MAP

There are three techniques based on features obtained from important places;

1. MINIM which does exact matching of the binary features
2. μPIN which does approximate matching of the binary features
3. MAP which does approximate matching of numeric features

MINIM and μPIN

After the preprocessing and the extraction of important places steps, we obtain important places for every i position of the period (i from 0 to $T - 1$). Later, we enumerate important places. The enumeration begins from 0 each time we begin enumerating important places of a new position of the period. After the enumeration of all important places, we generate the discrete representations from the location sets LS_i^g. Each LS_i^g of S will be represented by a bit vector. If there is a location measurement of LS_i^g spatially contained in the important place with label j, then jth offset of the bit vector will be replaced with 1. If there are not any location measurements of LS_i^g contained in the important place with label j, then jth offset of the bit vector will be replaced with 0. Notice that, while these bit vectors are built, only the important places of the corresponding position of the period are used.

We obtain S^g after changing every LS_i^g of S to a bit vector and keeping the order intact. For instance, from $LS_0^g LS_1^g LS_2^g LS_3^g LS_4^g LS_5^g$, a sequence S^g such as $r_0 r_1 r_2 r_3 r_4 r_5$ will be obtained where each r_i is a bit vector obtained from LS_i^g.

Example 4.1.1 *Assume that we have important places with label* 0, 1, 2 *for the zeroth position of the period. After we take a look at* LS_0*, we see that important places with label* 0 *and* 1 *are visited but the important place with label* 2 *is not. Then* LS_0 *will be represented by* $< 1, 1, 0 >$*. If we take a look at* LS_7*, we see that important places with label* 0 *and* 2 *is visited but label* 1 *is not visited. Then* LS_7 *will be represented by* $< 1, 0, 1 >$.

As the counting of frequent 1-patterns begins, the difference between MINIM and μPIN appears.

For MINIM, we will separate each (i) position of the period by using discrete representation sets (DRS_i^g) and group the same bit vector contents in these sets together. We will do the counting in DRS_i^g set for each i from 0 to $T - 1$ separately. Groups of bit vectors with the same content and with more than $min\_sup \times \left(\lfloor \frac{|S^g|}{T} \rfloor\right)$ elements inside will form frequent 1-patterns. All the elements in these largely populated clusters will be labeled with a new label given to the discrete representation they contain while the elements of clusters of size less than the threshold will be labeled with "*". Thus, S^L will be obtained. Notice that label here points to the discrete representation in the cluster. Obtaining frequent 1-patterns from the representatives is trivial. For example, for a representative with label l in the zeroth position of the period where $T = 4$, "$l * * *$" is a frequent 1-pattern.

Time Complexity Analysis 4.1.4 *MINIM will just need two scans over all location measurements for this part. The construction of bit vectors from LS_i^g will be completed first. After this information is obtained, we will need again a single scan to do the counting and to extract frequent 1-patterns which adds $O(N)$ to total time complexity.*

For μPIN, we will separate each (i) position of the period by using discrete representation sets (DRS_i^g) and group approximately same bit vector contents in these sets together. The motive of grouping similar contents is that it offers an approximation instead of an exact matching. There is a chance that exact matching produces frequent 1-patterns with very low support. For the grouping step, we will use AGNES with a binary dissimilarity measure that we tailored for our task.

The motive of designing a binary dissimilarity measure is that previously proposed dissimilarity measures are not fit for our task. Three major families of metrics are taken into consideration;

1. Hamming distance [15] with different normalizations (Sokal and Michener [35], Rogers and Tanimoto [31])

2. Normalized inner product with different normalizations (Russell and Rao [32], Jaccard and Needham [20], Dice [9], Kulczynski [23])

3. Correlation similarity measures (Yule and Kendall [38])

Definition 4.1.1 *x_i will denote the value of bit vector x in its ith offset. Assuming we have two bit vectors x and y, the case where $x_i = 1$ and $y_i = 1$ is the positive case and the case where $x_i = 0$ and $y_i = 0$ is the negative case. Notice that x and y are both discrete representations (such as r_i and r_j) belonging to the same discrete representation set.*

The first and the third type of dissimilarity metric treat both negative and positive matches equally. The problem with this approach is that in our task, it is not always meaningful to have a positive case. As our data is about the presence/absence of important place visits, it is more probable to have a visit to an important place which leads to frequent occurrences of positive cases. For instance, "home" is an important place for Tom. Assuming that we work on *day* granularity with the period of 7, we will see that almost every Monday (contained in DRS_0^{day}) Tom visits home. So the bit vectors representing every Monday will have "1" for the offset of "home". As the first and third type of metric will give the same weight to every feature in bit vectors and as these metrics treat positive and negative cases equally, positive cases which were trivial will be treated as if they are as important as negative cases and they will contribute to similarity as much as any matching case.

An alternative is the second type of metric discussed above. The second type of metric completely ignores the presence of negative cases, because negative cases are supposed to be insignificant. In our task, having a positive case can be considered as insignificant, so it can seem trivial to adapt Jaccard-like metrics to our task. Let's see why this approach will be problematic. For example, assume that we are mining on *day* granularity with $T = 7$. Two discrete representations depicting two Mondays $monday_1 = \{1, 1, 1, 0\}$ and $monday_8 = \{1, 1, 1, 1\}$ will have a dissimilarity equal to 1 (on 1) with the Jaccard metric. If all these "1" values in the zeroth, first and second offset were totally insignificant, then this approach would have the effect that we desire, but we don't have an idea about their level of significance; maybe the zeroth one is "home" which means that the positive case here is totally insignificant, but if it was "shopping mall" that has 50% chance of visit? This leads us to question the definition of significance of positive and negative cases in our task.

Definition 4.1.2 *A value in ith value of the bit vector is insignificant if it is "generally seen" in ith offset of all bit vectors of the same position of the period.*

Notice the similarity of the definition above with the logic of "inverse document frequency"[33] where a word gains more weight if it is seen in less documents.

Example 4.1.2 *Assuming that we compare different Mondays represented with bit vectors, it is usual to see the value "1" in the "work" offsets which implies that it is insignificant. If we were comparing two different bit vectors, a positive case for the "work" feature would give us very little information about the similarity of these bit vectors. Consider the inverse case where on both Mondays, the traced object does not go to work. As it is a rare case, it must have a large influence in the similarity of these days.*

	0	*1*
0	a	b
1	c	d

Table 4.1: Contingency table

Before the dissimilarity measure, we give the contingency table which can be seen in Table 4.1. Notice that the first row and the first column of contingency table are the possible values of x_i and y_i where x and y are both bit vectors.

Our dissimilarity measure gives more importance to significant positive / negative matches in the compared bit vectors.

$Dissim(x, y) = \frac{|b|+|c|}{|b|+|c|+\sum[2(1-\lambda_i)s_i+2\lambda_i t_i]}$
where s_i is 1 only when $x_i = 1$ & $y_i = 1$ (positive case), and t_i is 1 only when $x_i = 0$ & $y_i = 0$ (negative case). $|b|$ here is the number of offsets with $x_i = 0$ and $y_i = 1$ and $|c|$ is the number of offsets with $x_i = 1$ and $y_i = 0$. λ_i is the weight for *i*th offset.

It is easy to see that if we set all $\lambda_i = 1/2$ (i.e. both negative and positive cases are equally important), our measure transforms into "Sokal and Michener" metric [35] while if we set all $\lambda_i = 1$ (i.e. the presence of d is totally insignificant), it transforms into "Dice" metric [9].

If we knew the importance level of each offset, we could easily set λ_i values in our dissimilarity measure. We previously discuss the direct relation between the general presence / absence of positive and negative cases and their impact on the similarity. We use this relation to estimate the λ_i parameters. Offsets (important places) of each bit vector in the discrete representation set DRS_i^g are scanned separately for finding the occurrence rates of "1" in each offset, then these estimates are used. Obviously, this process has to be repeated for each i value from 0 to $T - 1$. For instance, assuming that $T = 5$ and we have 3 discrete representations in DRS_0^g such as $r_0 =< 1, 0, 1 >$, $r_5 =< 1, 1, 1 >$, $r_{10} =< 1, 0, 0 >$, our estimates for each column will be $\lambda_0 = (1 + 1 + 1)/3$ (occurrence rate of "1" in zeroth offsets), $\lambda_1 = (0 + 1 + 0)/3$ (occurrence rate of "1" in first offsets), and $\lambda_2 = (1 + 1 + 0)/3$ (occurrence rate of "1" in second offsets) respectively. After we find λ_i values the clustering phase can begin.

Example 4.1.3 *Assume we have two Mondays to compare such as* $monday_1 = \{1, 1, 0\}$ *and* $monday_8 = \{1, 0, 0\}$ *and the estimates for the offsets are* $\lambda_0 = 0.4, \lambda_1 = 1, \lambda_2 = 0.5$. *If we compare both Mondays, it is immediate to see that only the first offset is different. If we use our dissimilarity measure, the result will be* 1 / (1 +2×(1−0.4)×1 + 2×0.5×1) = 0.31. *As you can see, a similarity on the first offset (with a* 100% *chance of happening) has no influence on the similarity between bit vectors. On the other hand, for the similarity in the* *zeroth offsets, the dissimilarity of the measure decreased considerably because seeing a positive case in this offset is interesting considering the fact that "1"s has* 40% *chance of happening on this offset.*

If we want to use a binary dissimilarity measure with AGNES, then we have to find an easy way for the user for setting optimal stopping criteria values. Setting the optimal stopping criteria value for the clustering task is difficult in Jaccard distance, but that is not the same case with normalized Hamming distance assuming we use complete linkage in AGNES. User can easily fix a stopping value using his maximum allowed number of non-matching columns ($|b + c|$). For instance, if he allows that, at most 2 offsets can be different between two bit vectors and we are working on discrete representations with 6 important places, then $\frac{2}{6}$ will be his stopping criteria.

Setting the optimal value for our dissimilarity measure is not as intuitive as it is in the normalized Hamming distance case, so we should propose a method for helping the user. Grid search can be done for finding this optimal value, but is it really necessary to use all $(0, 1)$ interval during the grid search? From the answer of this question, we gain inspiration for an analytical way to find the most narrow interval for the grid search.

For our task of extraction of the most narrow interval for grid search, we will first take the maximum number of difference ($|b + c|$) allowed by the user (as in the normalized Hamming distance case) as input. Let's name it v. Later, using the estimates of every offset (λ_i values), we will fix a lower bound for only one difference ($|b| = 1$ or $|c| = 1$) case and then an upper bound for the v difference case ($|b| + |c| = v$). The point of giving a lower bound instead of just using 0 is that it will help reducing the interval. It is not logical to fix a low stopping criteria value that we already know that, if we use it, none of the clusters will be merged, so we followed this approach.

For finding the interval, we will use λ_i values as previously stated. First, we will find the upper bound and then the lower bound. As you'll easily realize the order of these phases is trivial.

The sequence *min_seq* is built using the minimum cases for each λ_i from $i = 0$ to the total number of important places belonging to the actual position of the period. i.e. If λ_i is more than 50%, we will use $(1 - \lambda_i)$ else we will use λ_i itself. Notice that the ith element in *min_seq* is λ_i or $1 - \lambda_i$ (i.e. λ order is intact). The weights in *min_seq* are delimited by "<" and ">".

Weights in *min_seq* has the minimal effect on the decrease of dissimilarity. Using these weights and input v of the user, we will find an upper bound. It is clear that we can find it by (i) choosing v different elements with largest values from *min_seq* and (ii) supposing that these v elements are non-matching offsets between two bit vectors. After that, we will use the weights in *min_seq* that are not used in the previous step to obtain the dissimilarity of the most dissimilar object available for the input v. The upper bound is $\frac{v}{v+2\times sum}$ where *sum* is the sum of $(k-v)$ smallest weights in *min_seq* supposing we have k offsets in a bit vector. From now on, we will use k as the total number of offsets in the bit vector (i.e. number of important places).

Now, a similar sequence to *min_seq* has to be found. ith element z in *min_seq* has a counterpart $1-z$ in ith offset of *max_seq*. i.e. *max_seq* is a sequence of weights with the possible largest contribution of similarity. The weights in *max_seq* are delimited by "<" and ">".

We will choose the minimum weight in *max_seq* and suppose that its offset is the only offset where bit vectors don't match (i.e. if two sequences are x and y, $x_i = 0 \land y_i = 1$ or $x_i = 1 \land y_i = 0$). After that, we will use all weights that are not used in *max_seq* in the previous step, and then we will obtain the dissimilarity value for the most similar objects that are bound by the "1 offset of difference". The lower bound is $\frac{1}{1+2sum_2}$ where sum_2 is the sum of all weights in *max_seq* minus the minimum weight in *max_seq*.

Example 4.1.4 *Assume that* $\lambda_0 = 0.9$, $\lambda_1 = 0.5$, $\lambda_2 = 0.4$, $\lambda_3 = 0.8$, $\lambda_4 = 0.6$. *Suppose that the user wants at most* 1 *difference in offsets (i.e. one offset that does not match,* $v = 1$*). Then min_seq* $=< 0.1, 0.5, 0.4, 0.2, 0.4 >$ *and max_seq* $=< 0.9, 0.5, 0.6, 0.8, 0.6 >$. *For the upper bound, we use min_seq and v. We choose v largest elements from min_seq.* 0.5 *is the largest here. The remaining weights will be used in the denominator. The value of the upper bound is* $\frac{1}{1+2(0.1+0.4+0.2+0.4)} = 0.31$. *The lower bound uses max_seq and its value is* $\frac{1}{1+2(0.9+0.6+0.8+0.6)} = 0.15$ *where the denominator contains all weights except the minimum one from max_seq. As you can see,* $(0,1)$ *is reduced to* $[0.15, 0.31]$ *with our approach.*

As v takes values closer to k (number of important places), our approach becomes less beneficial since our upper bound becomes closer to 1. Furthermore, if the λ_i values are generally close to extremities like 0 or 1, then again it is less beneficial to use our approach. Let's note here that if all the lambda values were 0.5, then our approach's upper bound gives the same value with the one that is used as normalized Hamming distance's stopping criteria ($\frac{v}{k}$ where $v = |b| + |c|$).

Using AGNES and our dissimilarity measure, we cluster each DRS_i^g for i from 0 to $T-1$ separately. Then, we will extract clusters with more than $min\_sup \times \left(\lfloor \frac{|S^g|}{T} \rfloor\right)$ elements inside. From these clusters, it is trivial to find frequent 1-patterns as previously explained. For instance, for $T = 4$, and for the zeroth discrete representation set (DRS_0^g), we find a single largely populated cluster C. After we find a single representative discrete representation r (bit vector) for the cluster C, we know that "$r***$" is a frequent 1-pattern.

The dominant bit vector in the cluster will be chosen as the cluster representative. Later, we will do the labeling. Discrete representations (bit vectors) which are elements of a largely populated cluster will be labeled with the new label given to cluster's representative while the other discrete representations will be labeled as "$*$". Notice that label here points to the discrete representation of the cluster representative. After this labeling step, we are ready for the mining phase on the newly obtained sequence S^L.

Time Complexity Analysis 4.1.5 *During the analysis of MINIM, we show that we need a single scan of database for obtaining bit vectors extracted from LS_i^g. During this scan, the estimation of λ_i values can be completed too. After that, AGNES will be applied. We have $\frac{N}{k}$ bit vectors in total. We will split the total into T (period) parts and apply AGNES T times. The resulting complexity for the clustering is $T \times O\left(\frac{N^2}{k^2T^2}\right) = O\left(\frac{N^2}{k^2T}\right) = O(T)$ where $N = |S|$ and k is the number of elements in LS_i^g.*

MAP

After the extraction of important places, we obtain important places for every i position of the period (i from 0 to $T-1$) and we enumerate these important places. The enumeration begins from 0 each time we begin enumerating important places of a new position of the period. Later, we build sequence of tuples from each location set LS_i^g of S. The number of readings on and visits to the important places will be calculated using location sets and each LS_i^g will be described by a sequence of tuples such as $<(reading_0, visit_0), ..., (reading_q, visit_q)>$ where $reading_f$ is the number of readings in the important place with label f and $visit_f$ is the number of visits to the important place with label f in LS_i^g.

Example 4.1.5 *Assume that there are two important places (with label* 0 *and* 1*) and from the content of the LS_3 set we know that important place with label* 0 *is visited one time and there are* 3 *readings in it while label* 1 *is visited two times and there are* 4 *readings in it which means that LS_3's sequence of tuples will be* $<(3,1),(4,2)>$.

After we obtain sequence of tuples from each location set (r_i for LS_i^g), we will build a sequence S^g without disturbing the order of location sets. For instance, from $LS_0^gLS_1^gLS_2^gLS_3^gLS_4^gLS_5^g$, a sequence S^g such as $r_0r_1r_2r_3r_4r_5$ will be obtained where each r_i is a sequence of tuples obtained from LS_i^g.

Later, we will separate each i position of the period by using discrete representation sets (DRS_i^g) and group approximately same sequence of tuples in these sets together. For this grouping step, we will use AGNES with Euclidian distance. Later, the labeling will be done. We will choose clusters with more than $min\_sup \times \left(\lfloor\frac{|S^g|}{T}\rfloor\right)$ elements. For these large clusters, we will treat the cluster mean as the cluster representative. Later, we will label the elements of these clusters with a new label given to the cluster representative. Label here points to the discrete representation of the cluster representative. Again, the elements of clusters with less elements than the threshold will be labeled as "$*$". Then, S^L will be obtained.

Time Complexity Analysis 4.1.6 *A single scan over all location points is required for the extraction of sequence of tuples that MAP makes use of, which adds $O(N)$ complexity to this technique. After that phase, AGNES is applied. AGNES runs with $O(n^2)$ for n samples normally. We separate the whole location measurements (of size N) into sets of size k which gives us $\frac{N}{k}$ parts. All these parts will be used in creating sequence of tuples which will be clustered. But first, we will divide it into T parts and do the clustering separately for each part. This implies $T \times O\left(\frac{N^2}{k^2T^2}\right) = O\left(\frac{N^2}{k^2T}\right) = O(T)$ as complexity.*

4.2 Mining of the Frequent Patterns

After the frequent 1-patterns are extracted and the labeling is done on the sequence S^g, we obtained S^L which is basically a sequence of labels. After that, we will follow the approach proposed in [17] for the construction of a special tree from the sequence S^L and [7] for the counting from the tree. Notice that both of these steps are present in all our techniques.

There are few definitions to be done while explaining the approach of the cited papers but before that let's note that definitions of discrete representations such as "segment", "periodic pattern", "length of a pattern", "subpattern" can be directly used with labels. The sole difference between the usage in discrete representations and labels is that the first uses discrete representations as building blocks while the second uses labels. Notice that we denote labels as alphabetic characters.

Definition 4.2.1 *A candidate max-pattern C_{max} is a pattern generated from frequent 1-patterns by merging them into one if they have non-* values in their different positions (such as "$a**$" and "$*b*$" merged into "$ab*$") and by building a set from their content if these non-* values are found in the same positions in both patterns (such as "$a**$" and "$c**$" merged into "$\{a,c\}**$").*

Example 4.2.1 *"$a**$", "$*b*$", "$*c*$" and "$**d$" are frequent 1-patterns. Then, $C_{max} = a\{b,c\}d$. You can ask yourself about the possibility of frequent 1-pattern such as "$*b*$" and "$*c*$" in the same time. It is only possible if min_sup less than 0.5 is used as minimum support value.*

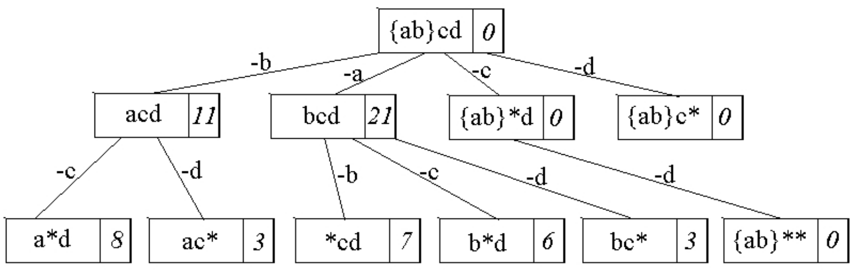

Figure 4.8: A max-subpattern tree

For holding segments and their total count in the sequence S^L efficiently, Han et al. propose a novel data structure called max-subpattern tree (which can be observed at Figure 4.8) which uses C_{max} as its root. Direct children of the root will be subpatterns of the root that has only one difference of non-* value with their parent. For instance, for $C_{max} = a\{b,c\}d$, there are four direct child nodes such as "$*\{b,c\}*$", "acd", "abd" and "$a\{b,c\}*$".

All non-root nodes of the tree will have direct child nodes with the same "loss of a single non-* value" approach. There is a constraint for a node to have a child though; it has to be a pattern of length 3 at least. It is easy to see the purpose of this constraint. If the pattern is of length two then its direct children will be of length one which is not really logical since we know the counts of frequent 1-patterns, we don't need to count them again.

As the purpose of the usage of max-subpattern tree is counting the patterns, there is a counter in every node of the tree. Furthermore, the nodes of the tree has a parent link (a single) and child links pointing to the corresponding nodes.

There are two phases we use;

1. Insertion of the label sequence into the max-subpattern tree (as in [17])
2. Traversal of the tree which extracts the frequent patterns (as in [7])

4.2.1 Insertion Phase

First, the label sequence S^L will be inserted into the max-subpattern tree. Every segment will be processed such as this:

- Beginning from the root, find the node that corresponds to the segment. This will be done by omitting non-* elements of the root in the left-to-right order to obtain the desired node. For instance, if $C_{max} = a\{b,c\}d$, and our segment is "$a * d$" then we will follow first the $\overline{b}$ edge and then the $\overline{c}$ edge to reach the "$a * d$" node.

- If the corresponding node is found, the count associated with it is incremented by one else this means that the node is not created yet, so create it and initialize the node with count value equal to 1 and then add its ancestor nodes recursively until the root node is reached. Notice that the ancestor nodes will be initialized with the count equal to 0.

From the insertion scheme, we know that each node will have only one parent while in reality they can have multiple parents. For instance, if the root is "$a\{b, c\}d$", and we inserted the node "$*cd$", we linked it to its parent "$*\{b, c\}d$" but "acd" is its parent too. This choice is made while preferring a tree structure to a lattice.

Definition 4.2.2 *Every "real" parent of a node (linked to it or not) is called a reachable ancestor.*

Reachable ancestors of a node are not only direct parents of this node. Reachable ancestors of a node z are every single node on the road from the initial node z to the root in the "lattice". The complexity analysis of this step can be found in [17].

4.2.2 Traversal Phase

After the whole sequence S^L is inserted to the max-subpattern tree, we will apply the traversal part of the algorithm STPMINE2v2 (proposed by Cao et al.[7]) which simply counts the occurrences from top to bottom in breadth-first order. It is obvious that if we want the "real" count of a node, we have to use the count of every reachable ancestor it has and sum all these counts. That is due to the fact that child nodes (subpatterns) are more general than parents (patterns) and if a segment complies with the parent, it is sure that it will comply with the child of it.

After we apply the algorithm, all patterns with count more than $min\_sup \times \left(\lfloor \frac{|S^L|}{T} \rfloor\right)$ will be extracted. With the approach of Cao et al., only maximal frequent patterns are mined which means that redundant patterns (frequent patterns that are subpatterns of frequent maximal patterns) are overlooked. The complexity analysis of this step can be found in [7].

After the frequent patterns from S^L are extracted, discrete representations pointed by the labels are fetched. Considering a single non-* position of a pattern (a discrete representation)

- MINOR/PAMUC returns a minimum bounding rectangle/convex hull.

- MINIM and μPIN returns a set of rectangles depicting locations occupied by visited important places.

- MAP returns a table with a set of rectangles depicting the locations of visited important places as the first column and reading / visit values corresponding to these important places as the second column.

Chapter 5
EXPERIMENTS

In this chapter, we present the experiments conducted on synthetic data. In Section 5.1, we explain the working of our synthetic data generator. In Section 5.2, performance gain of the preprocessing step is analyzed and preprocessing step's safety net behavior (for wrong parameter selections in DBSCAN) is shown on an example. This section concludes with our experiment on possible loss of information due to the preprocessing step. In the subsequent section, proposed binary dissimilarity measure is compared with two popular distance metrics for their precision and recall in clustering using AGNES. In Section 5.4, proposed geometry dissimilarity metric's accuracy is evaluated visually. In the subsequent section, gain in grid search with our interval narrowing technique is evaluated. In Section 5.6, the impact of representations on the accuracy of exact matching techniques is evaluated. After that, representation compactness of exact-matching techniques (MINOR, PAMUC, MINIM) is evaluated by comparing the total areas of discrete representations pointed by labels of their patterns. Section 5.8 evaluates the effectiveness of the proposed techniques. The last section of this chapter evaluates the efficiency of the proposed techniques.

All experiments are conducted on a notebook with 1.5 GHz clock speed and 512 MB of RAM which runs Windows XP. Implementation of the techniques are done in Java programming language. PostgreSQL with PostGIS extension is used as RDBMS.

5.1 Data Generator

As finding publicly available spatio-temporal data is difficult, we choose to use a data generator during our experiments as in the case of [7]. We use the source code of the data generator proposed in [7] and rewrite it with "waiting time in important places" and "minimum probability of visiting important places" additions. As in real life, traced objects spend a fair amount of time in important places which implies the need for the first extension. The absence of visits to important places is needed because otherwise our data set will not resemble to real life data. The new data generator generates trajectories with patterns in bottom granularity. Later, these trajectories are grouped such that the whole data will contain a pattern at the coarser granularity. So, the generated data set contains patterns in bottom granularity as well as patterns in the coarser granularity. This way, we were able to generate synthetic data sets that are closer to real life data.

From now on, we will use the expression "delay values" instead of "waiting times in important places". Positions with non-* patterns in the bottom granularity are called "periodic slots" while "non-periodic slots" denote positions with "*" patterns in the bottom granularity.

We will now explain the pattern generation at the bottom granularity. Notice that everything that will be explained about this generation is from [7] excluding "waiting time in important places" and "minimum probability of visiting important places". Interesting parameters of the generator in [7] such as period and ratio of periodic slots are fixed. Period is fixed to 24. Without loss of generality, we assume that the bottom granularity is *hour*, so with a period of 24, we were able to generate a single day with periodic patterns at hour granularity. Ratio of periodic slots is fixed to 75% which's very similar to real life behavior, because we assume that in general $\frac{3}{4}$ of our day is spent in important places. Notice that during the experiment in Section 5.2, we try different values in ratio of periodic slots.

As a beginning step to the data generation, the delay values of periodic slots are randomly chosen from the interval $[MinDelay, MaxDelay]$. We set $MinDelay$ to 2 (hours) and $MaxDelay$ to 7 (hours) to mimic real life. Later, the pattern centers for periodic slots ($Center_{P_i}$) are created randomly. Pattern centers can be considered as the centroids of an important place. The distances between pattern centers are proportional to their difference in time.

After these steps, the generation of trajectories begins:

- If the actual slot P_i is a periodic one (a visit to an important place) then "minimum probability of visiting an important place" is used to decide if this periodic pattern will be visited or not. If it must be visited, a location in α neighborhood of $Center_{P_i}$ is randomly chosen as the location measurement.

- If the actual slot P_j is a non-periodic one, then the object moves towards the pattern center of the next periodic slot (important place). To simulate this effect, the pattern center of the previous periodic time slot ($Center_{P_h}$) and the pattern center of the next periodic time slot ($Center_{P_m}$) are found. Then we calculate step size $ss = \frac{distance(Center_{P_h}, Center_{P_m})}{|t_{P_h} - t_{P_m}|}$ which is the Euclidian distance between previous and next pattern centers divided by the temporal distance between them. The object moves towards the next pattern center ($Center_{P_m}$) with step size of ss. For not generating undesired periodic patterns using the step size and its current angle, we will distort the angle by adding it an angle from $[\frac{-180}{\pi}, \frac{180}{\pi}]$ interval and the step size by multiplying it with a random number from [0.8,1.5] interval.

The parameters pertaining to the coarser granularity are;

1. Time granularity (Day, $Week$, $Month$, $Year$)
2. Number of segments in coarser granularity (nos)
3. Period (T) of the coarser granularity

We generate T types of day content such that, in total, they can form nos segments in coarser granularity. After that, we will do the partitioning of the data and obtain a data set with patterns at coarser granularities. As a time tick of all proposed time granularities can be formed by grouping time ticks of granularity *day* together, it is straightforward to build the content of a coarser granularity's time tick. All we have to do is to group enough day trajectories of the same type such that their time total is equal to a time tick of coarser granularity. For instance, if we work at *week* granularity, we have to group 7 days of generated trajectories of the same type. After the contents of the coarser granularity's time ticks are prepared, they are partitioned such that a time tick corresponding to ith position of the pattern will contain trajectories of type i (i.e. each position of the period T will contain different types of trajectories).

One of the trajectories (generated for a single position of the period) can be seen in Figure 5.1.

5.2 Evaluation of High Speed Movement Data Elimination

Elimination of high speed movement data results in performance gain and acts as a safety net for possible errors that can occur during the selection of parameters for the extraction of important places. To validate our method, we generate 4 segments with period of "3" and with granularity equal to "week". As one location point is generated per hour, we have 24 (hours) $\times 7$ (days in a week) $\times 4$ (segments) $\times 3$ (period) = 2016 location-timestamp pairs in S sequence.

First we will show the safety net behavior by giving wrong parameters to DBSCAN on purpose. EPS parameter of DBSCAN is fixed to 6 and $MinPTS$ is set to 5 although we knew that it is not a good choice (It should be close to 20). In Figure 5.2, the result of the clustering can be inspected. Every single point in this figure is supposed to be the part of a dense cluster.

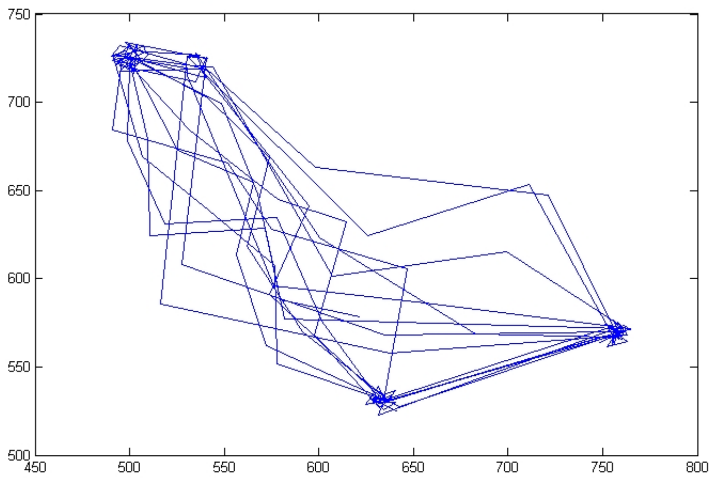

Figure 5.1: A complete trajectory of a single position of the period

As it is obvious, working with these badly formed clusters will be problematic in many ways. But assume that, first we omit points of high speed movement using our method and then apply the clustering. In Figure 5.3, you can witness the results of this approach. Points in circles are elements of the obtained clusters and the clusters match with the important places we generated with the data generator.

There are many cases in real life that has similarity with this discovery. Assuming that we travel with our car, pedestrian crossings and traffic lights will have less denser point regions around them than important places like “home” or “work”. But with a bad selection of *EPS* parameter in DBSCAN, they can be falsely labeled as important places. After we omit location points of high speed movement, regions around pedestrian crossings and traffic lights will lose some of their location points, because we don’t see red light every time we pass by this traffic light or there are not pedestrians every single second in this pedestrian crossing. As these regions will become less denser, it can be possible for DBSCAN to realize that they are not important places in spite of wrongly chosen parameters.

There are two other evaluations that needs to be done. The gain in performance and the loss of accuracy due to this step has to be calculated. We previously stated that the gain in performance totally depends on the percentage of travels in the traced object’s spatio-temporal sequence. So we change the parameter that effects the total time spent on important places and evaluate the results.

32 segments of data is generated using *day* as the time granularity, 4 as period value ($|S| = 3072$). Later, we try different values of total time spent on important places and calculate the gain in performance in “extraction of important places” phase. The results

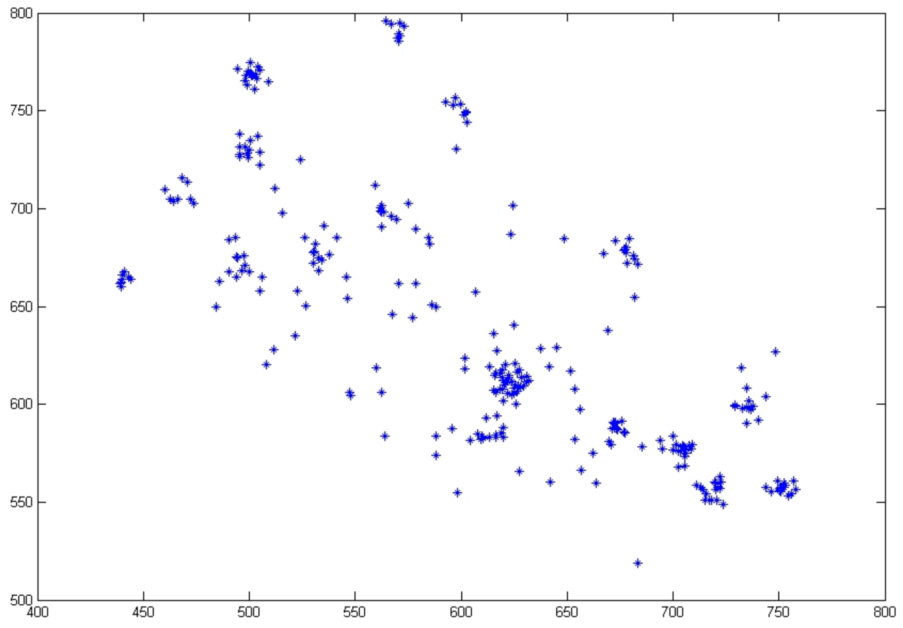

Figure 5.2: Clustering with EPS=5

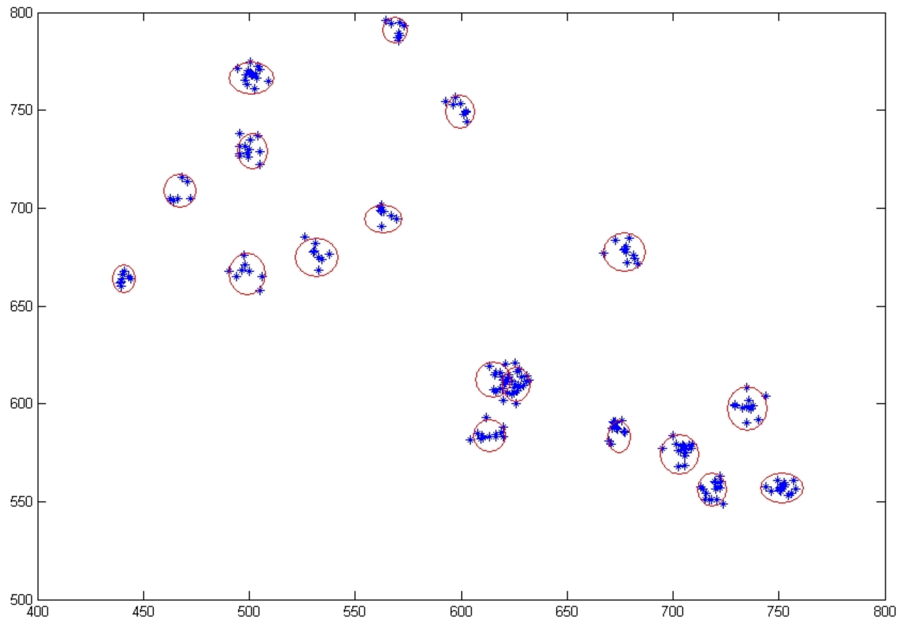

Figure 5.3: Clustering with EPS=5 after the preprocessing

min. % of time spent in important places	*35%*	*50%*	*65%*	*80%*
Gain in performance	34%	17%	10%	6%

Table 5.1: Gain in performance with the proposed preprocessing method

can be seen in the Table 5.1. Percentage values in the first row are minimum percentage of time spent on important places to total time. The performance gain is calculated by

$gain = \frac{time_{\overline{p}} - time_p}{time_p} \times 100$ where $time_p$ is the total time spent on the DBSCAN phase after preprocessing and $time_{\overline{p}}$ is the total time spent on the DBSCAN phase without preprocessing.

Obtained results prove that as the object spends more time traveling (like in the case of 35% of total time spent on important places and the rest spent traveling) as this phase is beneficial. Two extremities in percentage are 35% and 80% because less than 35% and more than 80% of time spent on important places does not seem to mimic real life behavior.

The other experiment to be conducted is the evaluation pertaining to the loss of necessary location points due to our preprocessing method. For this task, we first extract the location measurements spatially contained in the important places which will be used as the testset. Notice that we don't want our preprocessing step to omit location measurements from the testset. Later, we apply our preprocessing and do the extraction of important places. Then we check if the results after the extraction match with the testset. The inspection shows 0% of loss in the location points data. The dataset used for the experiment has *day* as the time granularity, 7 as the period, 50% chance of visiting an important place. 16 segments with these parameters were generated ($|S| = 2688$).

5.3 Evaluation of the Binary Dissimilarity Measure

The problem with the evaluation step is that a similar dataset to the desired one –a dataset with bit vectors depicting important place visits– has to be used and no such dataset is publicly available. So we had to design a binary data generator.

For our data generator, the overall "1" frequency in a single offset (frequency of "1" of a single offset for the whole DRS_i^g of the same position of pattern) is dependent of the input parameter and all the columns are supposed to be uncorrelated. That is, we give frequency percentages to different offsets (important places) and these percentage of occurrence don't have an effect on each other.

The bit vector generator generates the desired data and the clustering is done using two popular binary metrics and our dissimilarity measure. Later, the results of the clusterings are inspected with the "true" clustering that is done manually. To simplify manual clustering step, we generate a small dataset of 20 objects. Four columns with different "1" frequencies are generated. The frequency of "1"s are 90% for the zeroth column, 90% for the first, 60% for the second and 40% for the last column. The zeroth and the first column can be thought as "home" and "'work" for instance. Notice that "*i*th column" means x_i of all bit vectors x such that all x are elements of the same discrete representation set.

After the generation of bit vectors, we cluster them by hand. During this process, we used the information of the dominant value in an offset instead of the complete estimation for its 1 frequency. As previously stated if 0 (1) is more frequent than 1 (0) in an offset (considering all bit vectors of DRS_i^g of the same position of period) then we say that 0 (1) is dominant in this offset. We group bit vectors that contain less dissimilarity. By "less dissimilarity", we mean less number of $|c| + |b|$ and high number of matching offsets (high $|a| + |d|$) with the non-dominant values in the corresponding offsets (a, b, c, d were defined in Table 4.1). For instance, if the dominants of the offsets are "$1, 1, 1, 0$" in the respective order and we have $monday_1 = \{1, 0, 0, 0\}$, $monday_8 = \{0, 0, 0, 0\}$, then these Mondays are really similar because they only got one non-matching offset. Furthermore, the offsets that match are really important (because most of the values they contain are not dominant). After the dissimilarity matrix is populated with results S (Similar) and D (Dissimilar), we use the most straightforward approach and built clusters by hand by using "single linkage", because it will help the clustering phase with its chaining phenomenon. Jaccard distance and normalized Hamming distance are chosen for the evaluation task as they are the most popular metrics among two families of binary distance metrics. AGNES with these popular metrics is used for the clustering phase. The plan was to observe the final clustering results and compare them to the "right" clustering. For the evaluation purpose, we use micro-average-precision and micro-average-recall measures ([8]).

Micro-average-precision: $\frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)}$
Micro-average-recall: $\frac{\sum_i TP_i}{\sum_i (TP_i + FN_i)}$ where

TP: True positive (Sample belongs to cluster C_i, clustering result gives the same information.)

FP: False positive (Sample does not belong to cluster C_i, clustering result tells that it does.)

FN: False negative (Sample belongs to cluster C_i and clustering result tells that it does not.)

While using micro-average-precision and micro-average-recall, we use the same approach with [4]: We will give cluster labels to the clustering results. During this labeling,

	Precision	*Recall*
N. Hamming Dist.	100	70
Jaccard Dist.	100	70
Proposed Func.	**100**	**100**

Table 5.2: Precision and recall for dissimilarity functions

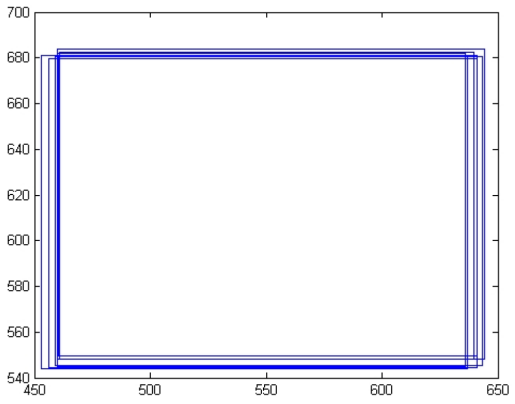

Figure 5.4: Cluster 1 containing 7 rectangles

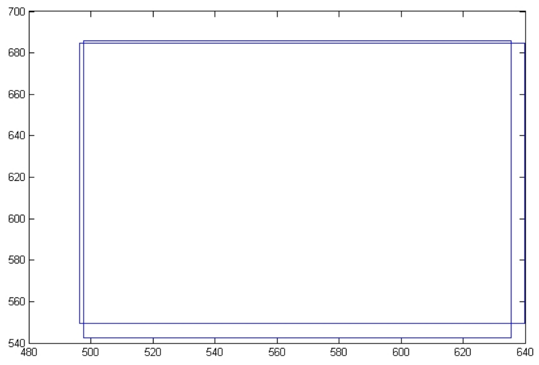

Figure 5.5: Cluster 2 containing 2 rectangles

we choose a labeling that as a result will increase the TP value as much as possible. We are doing this by using the most dominant object (the bit vector with the largest percentage of occurrence in the cluster) in the result clusters and relating these result clusters to the "right" clusters with the similar dominant object. For instance, assume that the right clustering is $A = \{1, 1, 1, 2\}$ and $B = \{3, 3, 3, 4\}$ where 1, 2, 3 and 4 are different types of objects and A and B are two clusters. Assume that the resulting clusters are $X = \{1, 1, 1\}$, $Y = \{3, 3, 3\}$, $Z = \{2, 4\}$. We can easily see that 1 is dominant in X and 3 is dominant in Y. We will link the resulting clusters to the right clusters by relating them with their dominant value. Thus X is linked to A and Y is linked to B.

Precision and recall values of three dissimilarity measures can be observed at the Table 5.2. N. Hamming distance and Jaccard distance suffer from the facts that are told previously, so their less successful results in recall is not really surprising.

5.4 Evaluation of the Dissimilarity Metric for Comparing Geometries

In this section, we evaluate the clustering results visually. 10 segments are generated with our data generator. AGNES finds three clusters with threshold of 0.25 as stopping criteria.

The first cluster has 7 rectangles, the second has 2 rectangles, and the third has only 1 rectangle. The first cluster (in Figure 5.4) succeeds in grouping all similar rectangles

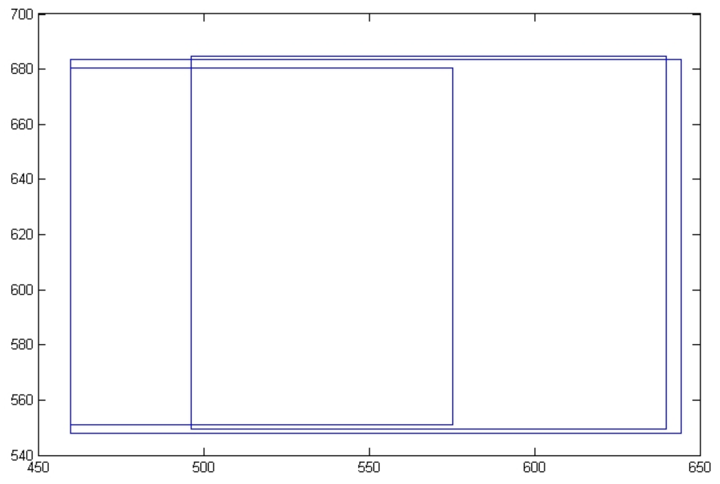

Figure 5.6: Three elements of different clusters

and is a good approximation compared to an exact matching of geometries. The second cluster (in Figure 5.5) groups similar rectangles successfully. Figure 5.6 contains the single element of the third cluster and one representative from the first and second cluster. We can see that all three minimum bounding rectangles present different important place visit contents, so they are in different clusters.

Later, we generate again 10 segments and cluster convex hulls with $threshold = 0.35$ as the stopping criteria of AGNES, then we visually evaluate the robustness of the proposed metric on grouping similar convex hulls. The first cluster (Figure 5.7) contains two similar convex hulls while the second cluster's (Figure 5.8) content may not seem as similar as the content of the first cluster; that is due to a higher dissimilarity between convex hulls of the second cluster.

Let's note that the stopping criteria for AGNES during the clustering of convex hulls is set to a higher value than the one we use for clustering of minimum bounding rectangles, so it is logical that elements in the minimum bounding rectangle clusters have a higher visual similarity than the elements of convex hull clusters. Both stopping criteria values (0.25 and 0.35) are found to be optimal in the accuracy experiments (in Section 5.6) which explains why we work with them during the visual inspection experiment.

5.5 Gain in Grid Search by Proposed Analytical Method

In this experiment, we inspect the gain in grid search with the usage of our interval narrowing method. First, three sets of weights with $\lambda_i \in [0.4, 0.9]$ is randomly generated for k important places. The $[0.4, 0.9]$ interval is chosen to mimic real life. We assume that

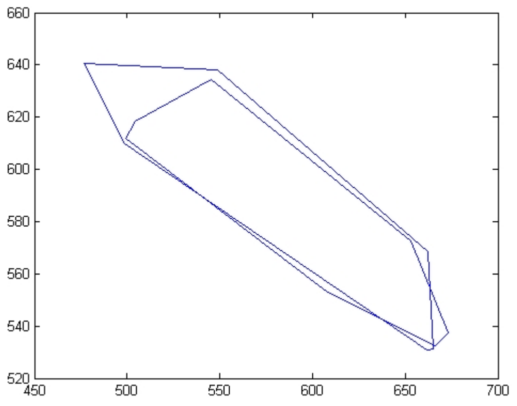

Figure 5.7: Cluster 1 containing 2 convex hulls

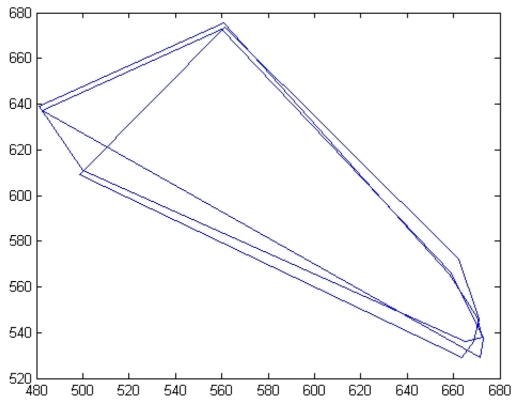

Figure 5.8: Cluster 2 containing 3 convex hulls

	1	2	*3*	*4*	*5*	*6*
2	79.17	-	-	-	-	-
3	87.6	-	-	-	-	-
4	80.77	50.63	-	-	-	-
5	86.8	61	38.85	-	-	-
6	85.7	61.66	41.74	-	-	-
7	87.44	66.12	47.08	32.85	-	-
8	90.8	73.05	56.41	43.2	-	-
9	84.28	64.03	47.8	35.12	25.62	-
10	**90.93**	74.85	59.14	46.61	35.99	27

Table 5.3: The percentage of gain obtained by our analytical method

visits to important places won't probably happen with frequency more than 90% or less than 40%. After the random generation of weights, we change the v value (the number of non-matching offsets between two bit vectors) and experiment the change in gain. Later, a new group of sets of weights is generated for a larger k value and the experiment is again conducted just like we explain above. The results of the evaluation can be seen in Table 5.3. The first column of the table is k values and the first row is v values. Notice that the gain percentages are calculated by $\frac{(upperbound-lowerbound)}{1} \times 100$ where 1 is the length of initial interval $(0, 1)$ and v values are always less than $k \times \frac{3}{5}$ because it is not logical otherwise.

5.6 The Impact of Different Representations to the Accuracy of Exact-matching Techniques

In this section, we measure the impact of different representations to the accuracy of exact-matching techniques (MINOR, PAMUC, MINIM) for their matching of important place content. For this evaluation, we generate 16 segments with the minimum probability of visiting an important place equal to 50%, period equal to 7 and *day* as the time granularity ($|S| = 2688$). Then preprocessing, extraction of important places, generation of discrete representations from location sets are performed. After that, either AGNES is used with our geometry comparison metric (in MINOR and PAMUC) or exact matching of bit vectors is used (in MINIM). Then, the accuracy of clustering/matching is evaluated with micro-average-precision and micro-average-recall by comparing the techniques' success in grouping same important places contents together.

Experiments show that MINOR sometimes matches minimum bounding rectangles whose contents in terms of important places are different. An error of that kind can be seen in Figure 5.9 where stars and triangles are location measurements belonging to different LS_i^{day} such that $i \ mod \ 7 = k$. As it can be easily seen, both location sets have the same minimum bounding rectangle although their important place visits were different.

Although false positives can be seen in MINOR, false negatives are never encountered. On the other hand, PAMUC has both false positives and false negatives which means that PAMUC sometimes considers convex hulls with different contents "similar" (see Figure 5.11) and convex hulls with the same content "different" (see Figure 5.12). In Figure 5.11, the dissimilarity of the geometries is really close to the stopping criteria value of AGNES, so such an error did happen. In Figure 5.12, you can see that both location sets (again, one location set's elements are depicted with stars, while the other set's elements are depicted with triangles) have the same important places content, but little differences between their location measurements changed the shape of the convex hull and AGNES could not catch their similarity. Empirical studies show that false negatives of convex hull clustering are generally seen when all important places are located as if they are positioned on a single line. When this happens, little changes in the location measurements make large differences in shape which can result in false negatives.

Although PAMUC has both kinds of errors, experimental results show that PAMUC's precision and recall is better than MINOR's. That fact is due to the larger approximation error of minimum bounding rectangles. An example supporting this claim can be observed at Figure 5.9 and Figure 5.10.

The precision and recall results of AGNES step of MINOR and PAMUC can be observed at Table 5.4. These results are calculated by comparing their clustering results with the "'right" clustering results obtained by the exact matching of important place contents, i.e. discrete representations with same bit vectors are grouped together which forms the "right" clustering.

MINIM is evaluated together with MINOR and PAMUC whose objectives are the

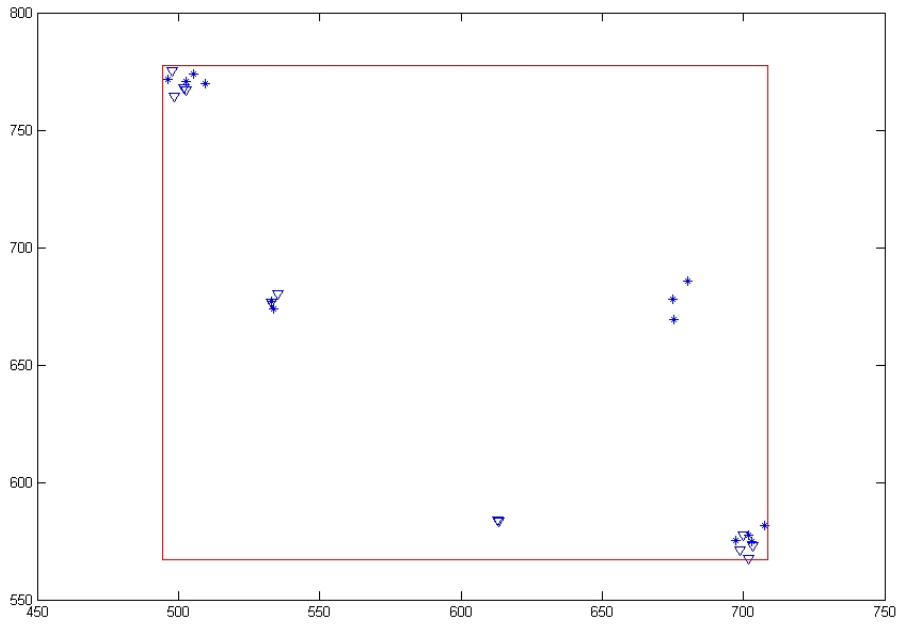

Figure 5.9: Different important place contents with the same minimum bounding rectangle

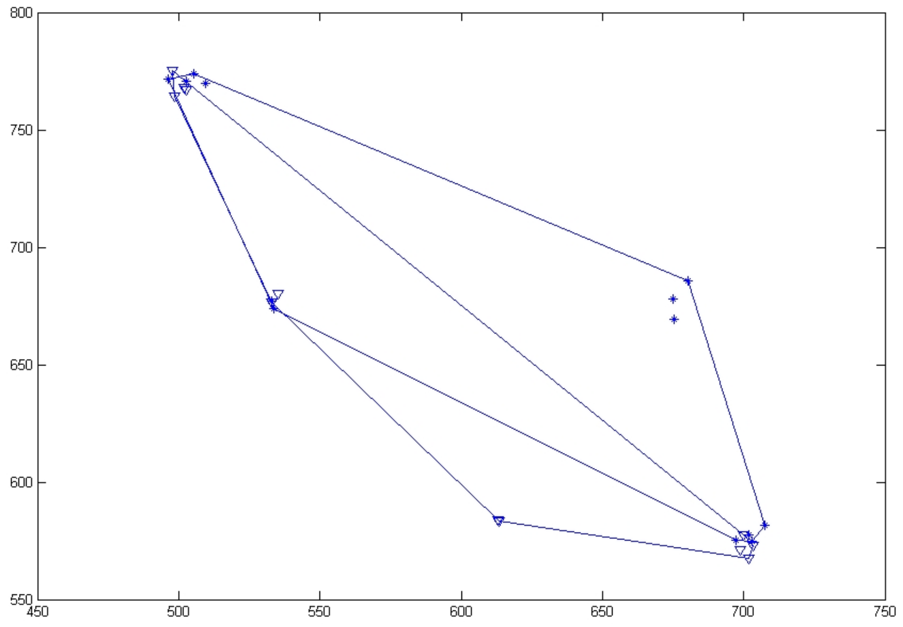

Figure 5.10: Different important place contents with different convex hulls

.

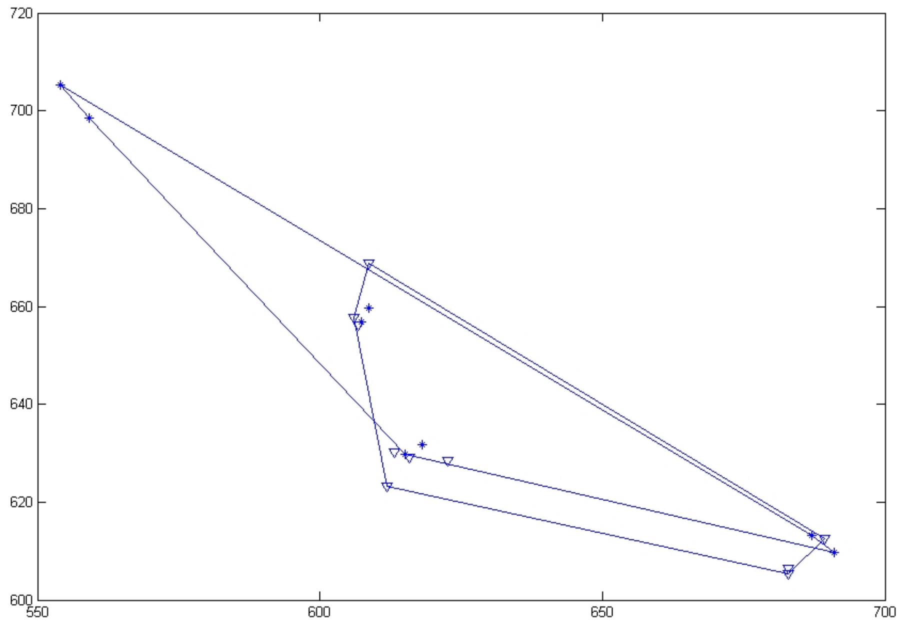

Figure 5.11: False positive in PAMUC

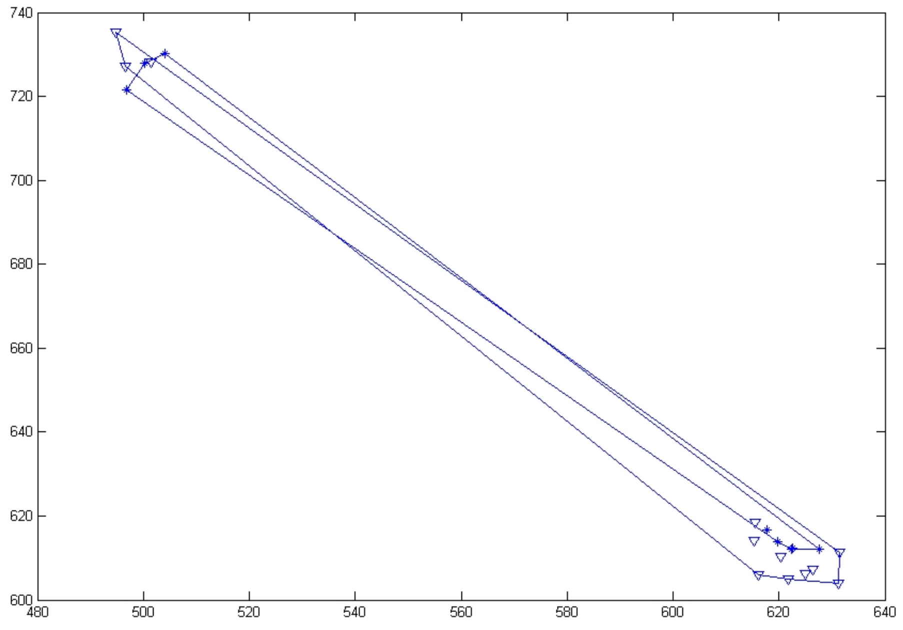

Figure 5.12: False negative in PAMUC

percent	*Precision*	*Recall*
MINOR	74	72
PAMUC	82	77
MINIM	100	100

Table 5.4: Precision and recall results for three techniques

MINOR	*PAMUC*	*MINIM*
4856045	1681503	7297

Table 5.5: Sum of representation areas of three exact-matching techniques

same (exact matching of important place contents). PAMUC is better than MINOR in precision and recall while MINIM outperforms both of them. MINIM has 100% success in precision and recall, because this technique uses representations based on important places visits and then does an exact matching of these representations. So, MINIM's precision and recall is 100% by definition.

5.7 Compactness of Representations for Exact Matching-based Techniques

MINOR, PAMUC and MINIM are to be compared for their compactness in their discrete representations' occupied area.

16 segments of data are generated with the period of 7 and *day* as the time granularity ($|S| = 2688$). 50% is used as minimum probability of visiting an important place.

Preprocessing, extraction of important places, generation of geometries/bit vectors, the grouping of similar geometries or grouping of exact bit vectors are all completed and the areas of the discrete representations are summed. As the discrete representations are directly related to the frequent patterns, the summation of their area gives an idea about how compact is the offered pattern information. The evaluation results are shown in Table 5.5. Patterns of PAMUC are more specific than patterns of MINOR and the reason is easy to realize; convex hull is by definition the convex shape with the minimal area. So as minimum bounding rectangles are convex shapes, convex hulls will occupy less area than rectangles. The large superiority of the patterns of MINIM over the patterns of PAMUC is not surprising too. As important places are locations like "home", "work", they occupy very limited areas which is why MINIM's results have very small areas compared to PAMUC. As these important places are not necessarily close to each other, convex hulls that try to spatially contain them all occupy a much larger area than the sum of areas that the important places occupy.

5.8 Effectiveness of the Techniques

Previous evaluations showed that the best of proposed mining techniques with exact-matching behavior on important places content is MINIM. So we use MINIM as the representative of the exact-matching techniques. μPIN and MAP are also evaluated in this section. Furthermore, the data mining algorithm that can work on bottom granularity (STPMINE2v2 proposed in [7]) is evaluated. Notice that, we generally follow the same logic with the work in [7] during effectiveness evaluations.

16 segments of data with period equal to 7, *day* as the time granularity and 100% as the minimum probability of visiting an important place are generated ($|S| = 2688$). So, we generate a pattern of length 7 with 100% support.

We then set the parameters of the MINIM. $MinPTS$ is set to 5 and EPS to 6. Notice that we easily find the optimal values for EPS and $MinPTS$, because the data generator's parameters help us. The min_sup value is set to 0.9. MINIM finds a single pattern of length 7 with 100% support. Extracted pattern's content is compared with the generated pattern's important place content and we see that they completely match. This result shows that MINIM is accurate. (i.e. It finds the whole pattern with right bit vectors in each position of the period)

After the evaluation of MINIM, we do the evaluation of STPMINE2v2 with the same dataset. STPMINE2v2 finds 7 patterns when it is run with a low support value such as $min\_sup = 0.14$. Notice that, such a low min_sup value is normally problematic because the fact that it can cause the mining of redundant patterns. Each position of the extracted patterns of STPMINE2v2 are the important places that are included in the patterns of *day* granularity, but obviously there are no further resemblance between the patterns of different granularities because STPMINE2v2 does not have a clue about the pattern of the *day* granularity. Let's note that, the important places our techniques use are not bounded to patterns of bottom granularity. i.e. it is possible that important places are formed without the need of existing patterns in bottom granularity. If our data generator was not generating patterns in bottom granularity (together with coarser granularity), then STPMINE2v2 could not find any patterns in the bottom granularity obviously.

Later, a small dataset (8 segments) is generated with period equal to 3, *day* as the time granularity and 75% as the minimum probability of visiting an important place ($|S| = 576$).

For the evaluation of μPIN, first, the extraction of important places is completed. Later, the clustering of bit vectors is done with the same logic as explained in Section 5.3. After that, discrete representations in the same cluster are given the same label and then the label sequence is mined for frequent patterns. 3 patterns with a support more than 50% is found. These three patterns will be our testset. After that, μPIN is run with 0.4 as *min_sup* value, $MinPTS = 5$ and $EPS = 6$. Remember that we easily find the optimal clustering parameters because the data generator's parameters such as α helps us. We chose to use a high threshold value as stopping criteria for AGNES (such as 0.7. i.e. 70% of dissimilarity at most), because otherwise we could have the same pattern results with the MINIM technique using the generated data set. The results of μPIN totally match with the testset which implies that this technique is accurate.

After the test of effectiveness for μPIN, we use the same dataset to evaluate the effectiveness of MAP. First, the extraction of important places is completed and the sequence of tuples depicting readings and visits to important places are built. After these steps, the clustering of sequence of tuples is performed as in Section 5.3 and discrete representations with the same cluster are labeled the same. After the labeling, we mine patterns from the label sequence and find only one pattern with support more than 50%. This single pattern will be our testset. Later, MAP is run with the same dataset and with parameters *min_sup* = 0.4, $MinPTS = 5$ and $EPS = 6$. The result obtained from the technique is the same with the testset. MAP finds a single pattern with support 50%. Again, we show that the proposed system is indeed effective.

After the evaluation of μPIN and MAP, we do the evaluation of STPMINE2v2 with the same dataset. STPMINE2v2 can't find any patterns until *min_sup* is chosen as low as 0.2. We previously argued the problem of using such a low minimum support value. Later, STPMINE2v2 is run and several patterns are extracted. Each position of the extracted patterns of STPMINE2v2 are again the important places that are included in the patterns of *day* granularity. It is obviously not possible for STPMINE2v2 to extract the patterns of *day* granularity; only the important places which our techniques use are found by STPMINE2v2 and that is due to the fact that our data generator generates patterns in bottom and *day* granularity together.

5.9 Efficiency of the Techniques

Different number of segments with granularity *day*, $T = 7$ are generated and the cost of our techniques with the increasing number of segments is evaluated. 7 is chosen as the period value because we assume that it is a moderate value while 4 (which is generally used with *week* granularity) is assumed to be the low and 12 (which is generally used with *month* granularity) is assumed to be the high value for period.

In Figure 5.13, the cost of exact-matching techniques can be seen. MINIM is faster than MINOR because as MINIM uses bit vectors as representations and does an exact

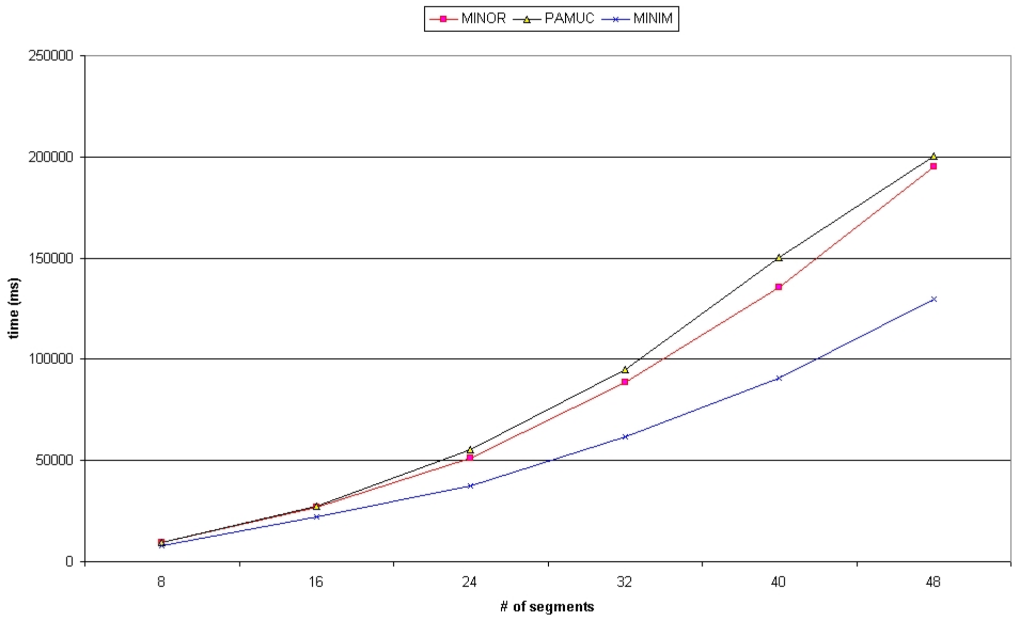

Figure 5.13: Cost of exact-maching techniques versus the number of segments

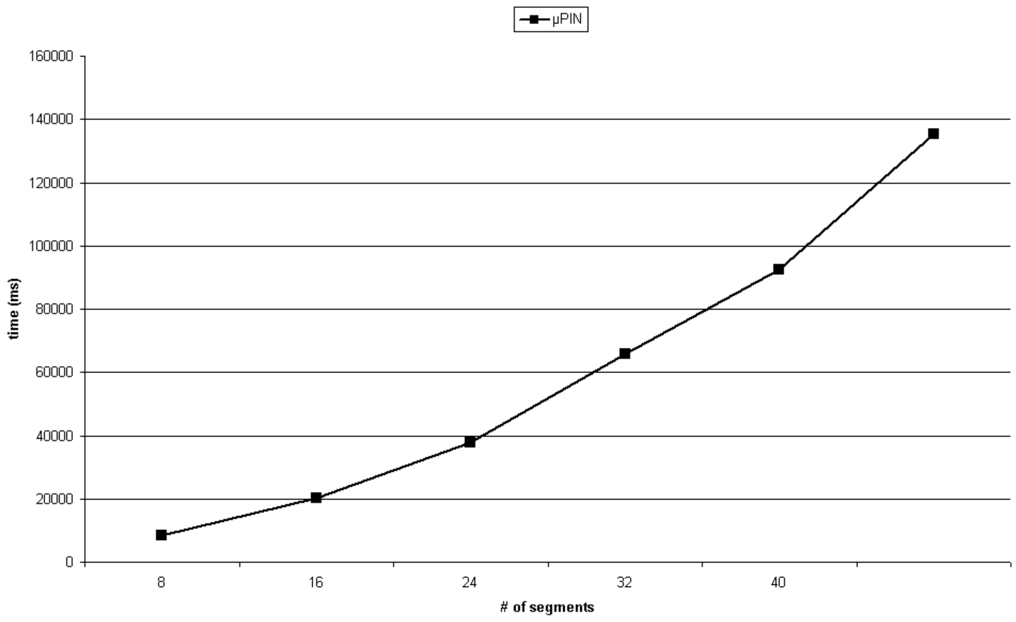

Figure 5.14: Cost of μPIN versus the number of segments

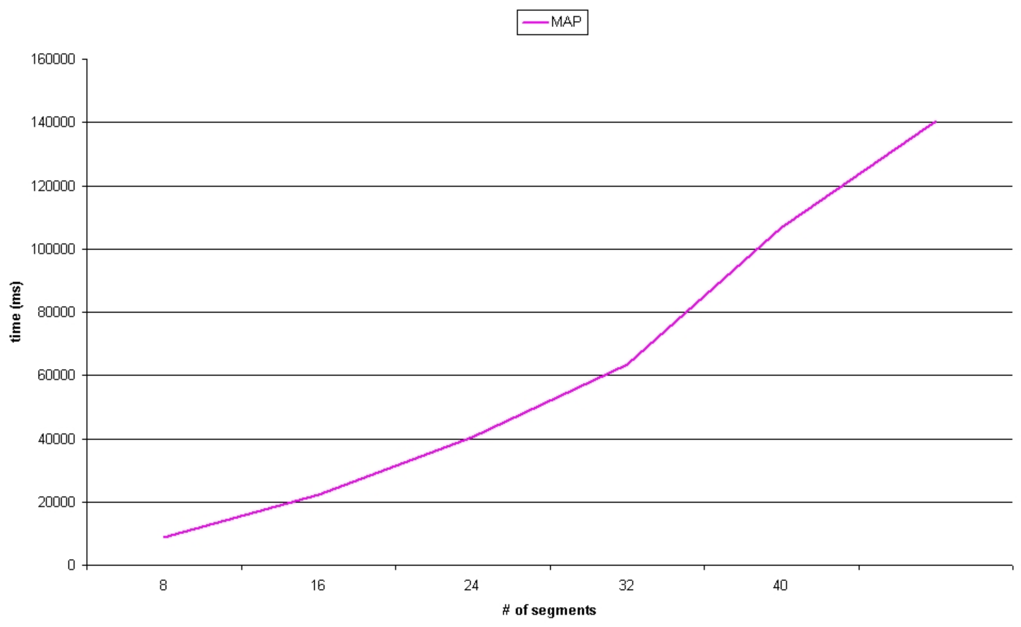

Figure 5.15: Cost of MAP versus the number of segments

matching on these representations, it does not need a "grouping of similar geometries" step as it is in the case of MINOR. MINOR is slightly faster than PAMUC because the discrete representation building step of PAMUC is more costly than MINOR's same step ($O(N \log k)$ versus $O(N)$ to be exact). Our experiments show similar costs for the phases after the extraction of frequent 1-patterns in exact-matching systems which is the reason why we don't comment on their addition to the total cost. In Figure 5.14, the cost of μPIN can be inspected. Notice that μPIN's cost is less than quadratic. In Figure 5.15, the cost of MAP can be seen. All proposed techniques are indeed efficient and scalable.

Chapter 6
CONCLUSION

In this thesis, we propose methods for mining periodic patterns at different time granularities from the spatio-temporal sequence of a single moving object.

The important places are extracted by a density-based clustering technique. As the first step, these important places are used for omitting redundant location points. As the second step, remaining location points are used for obtaining concise discrete representations. Two proposed techniques build geometries containing the location points. Two other proposed techniques use bit vectors depicting the visits/non-visits to the important places. The last proposed technique builds a sequence of tuples depicting the number of readings and visits observed in the important places. The subsequent phase can be divided into two categories: (i) Exact matching of important place contents, (ii) Similar matching of important place contents. After the matching phase, the initial spatio-temporal sequence becomes discretized and the frequent 1-patterns are obtained. After that, it is the straightforward application of techniques offered in [17] and [7] for the extraction of the periodic patterns from the discretized sequence.

During our experiments, we opted for the usage of a data generator due to the lack of publicly available spatio-temporal data containing periodic patterns. Different parts of the proposed techniques are evaluated. The preprocessing step (elimination of high speed data) is evaluated for its contribution to the gain of performance and for its behavior as a safety net for wrong parameter choices in the extraction of important places. We show that up to 34% gain of performance is possible with our technique. Furthermore, our experiments show that none of the necessary location measurements are omitted by this preprocessing step. After that, the proposed binary dissimilarity measure is compared with two popular distance metrics of different binary metric families where our measure outperforms the popular metrics during the usage in conjunction with AGNES. Later, the gain in grid search by proposed analytical method for finding the most narrow interval for the stopping criteria is evaluated. After that, the impact of different representations to the accuracy of our techniques (using exact matching of important place contents) is evaluated by the inspection of their matching accuracy on important place contents. Compactness of representations for exact matching-based techniques are then evaluated and the subsequent section evaluates the effectiveness of all techniques. All proposed techniques are perfectly doing the desired discretization process and they are accurate in extracting the frequent periodic patterns. The last section contains a cost analysis of the proposed techniques. All our techniques are shown to be efficient and scalable.

As a future direction of research, we are planning to conduct an extensive study on several socioeconomic layers of society for obtaining an improved method for the extraction of important places. Furthermore, the extension of our techniques with an effective method for the detection of optimal periodicity seems to be a fascinating idea.

APPENDIX

In this chapter, the proof of metricity for our geometric dissimilarity metric will be done. There are three properties needed for a function d to be metric:

1. Positive definiteness:
   $d(e_1, e_2) > 0$ for the case $e_1 \neq e_2$
   $d(e_1, e_2) = 0 \Leftrightarrow e_1 = e_2$

2. Symmetry:
   $d(e_1, e_2) = d(e_2, e_1)$

3. Triangle inequality:
   $d(e_1, e_2) + d(e_2, e_3) \geq d(e_1, e_3)$

Claim 1 *$distance(GEO_i, GEO_j) = 1 - \frac{(Area(GEO_i \cap GEO_j))}{\max\{Area(GEO_i), Area(GEO_j)\}}$ function is a distance metric.*

Proof 1 • *Positive definiteness:*
If $GEO_1 \neq GEO_2$, then $Area(GEO_1 \cap GEO_2) \geq 0$. We know that in any possible case $\max\{Area(GEO_1), Area(GEO_2)\} > 0$ and it always will be more than $Area(GEO_1 \cap GEO_2)$. If $Area(GEO_1 \cap GEO_2) = 0$ then

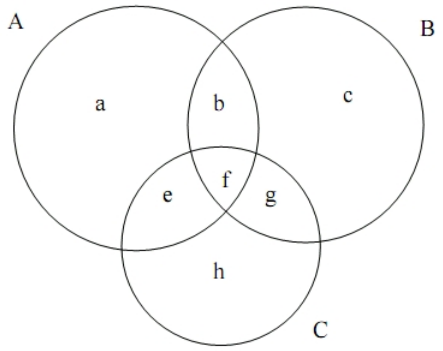

Figure 6.1: Venn diagrams for the case study

$distance(GEO_1, GEO_2) = 1 > 0$ else $distance(GEO_1, GEO_2) = 1 - Z$ where $0 < Z < 1$ thus again $distance(GEO_1, GEO_2) > 0$.

If $GEO_1 = GEO_2$, then $Area(GEO_1 \cap GEO_2) = Area(GEO_1) = Area(GEO_2) = K$. We know that $\max\{Area(GEO_1), Area(GEO_2)\} = Area(GEO_1)$ $= Area(GEO_2) = K$, thus $distance(GEO_1, GEO_2) = 1 - \frac{K}{K} = 0$.

- *Symmetry:*
  $(GEO_1 \cap GEO_2)$ and $(GEO_2 \cap GEO_1)$ will define the same geometry, thus we can say that $Area(GEO_1 \cap GEO_2) = Area(GEO_2 \cap GEO_1)$. Furthermore, we know that the order of the elements in the max *set is trivial for its outcome which implies* $\max\{Area(GEO_1), Area(GEO_2)\} = \max\{Area(GEO_2), Area(GEO_1)\}$.
  Thus, $distance(GEO_1, GEO_2) = 1 - X = distance(GEO_2, GEO_1)$ where X is $\frac{(Area(GEO_1 \cap GEO_2))}{\max\{Area(GEO_1), Area(GEO_2)\}}$ *or* $\frac{(Area(GEO_2 \cap GEO_1))}{\max\{Area(GEO_2), Area(GEO_1)\}}$.

- *Triangle inequality:*
  We will do a case study considering three geometries (GEO_1, GEO_2, GEO_3) depicted by three venn diagrams (A, B, C) in Figure 6.1. Without loss of generality, we can say that $Area(A) \geq Area(B) \geq Area(C)$. The lower case letters in the figure denotes areas. Our measure will be denoted by $D()$.

  From the symmetry relation and the relation of area between A, B, C, we know that
  $D(A, B) = D(B, A) = (a + e)/area(A)$
  $D(A, C) = D(C, A) = (a + b)/area(A)$
  $D(B, C) = D(C, B) = (b + c)/area(B)$
  $a + e \geq c + g$
  $a + b \geq g + h$
  $b + c \geq e + h$

  Now, we will show that three inequalities satisfy the conditions:

  1. $D(A, B) + D(B, C) \geq D(A, C)$
     $D(A, B) + D(B, C) = \frac{a+e}{area(A)} + \frac{b+c}{area(B)} = \frac{(a+e)area(B)}{area(A)area(B)} + \frac{(b+c)area(A)}{area(B)area(A)} \geq \frac{a\ area(B)}{area(A)area(B)} + \frac{b\ area(A)}{area(B)area(A)} \geq \frac{(a+b)area(B)}{area(A)area(B)} = D(A, C)$
  2. $D(B, A) + D(A, C) \geq D(B, C)$
     $D(B, A) + D(A, C) = \frac{a+e}{area(A)} + \frac{a+b}{area(A)} = \frac{(a+e)area(B)}{area(A)area(B)} + \frac{(a+b)area(B)}{area(A)area(B)} = \frac{(a+e)(b+c+f+g)}{area(A)area(B)} + \frac{(a+b)(b+c+f+g)}{area(A)area(B)} = \frac{(2ab+2ac+2af+2ag+eb+ec+ef+eg)+(b^2+bc+bf+bg)}{area(A)area(B)} \geq \frac{b(a+b+e+f)+c(a+b+e)+(a+e)f}{area(A)area(B)} \geq$

$\frac{(b+c)(a+b+e+f)}{area(B)area(A)}$ *(because* $a + e \geq c$*)*
$= \frac{(b+c)area(A)}{area(B)area(A)} = D(B, C)$

3. $D(A, C) + D(C, B) \geq D(A, B)$
   $D(A, C) + D(C, B) = \frac{a+b}{area(A)} + \frac{b+c}{area(B)} = \frac{(a+b)area(B)}{area(A)area(B)} + \frac{(b+c)area(A)}{area(B)area(A)} \geq \frac{(a+b+c)area(B)}{area(B)area(A)}$
   (because $area(A) \geq area(B)$*)*
   $\geq \frac{(a+e)area(B)}{area(B)area(A)} = D(A, B)$ *(because* $b + c \geq e$*)* □

Bibliography

[1] *M. Ankerst, M. Breunig, and H.-P. Kriegel, and J. Sander - OPTICS: Ordering Points to Identify the Clustering Structure, In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999*

[2] D. Ashbrook and T. Starner - Learning Significant Locations and Predicting User Movement with GPS, In Proceedings of IEEE Sixth International Symposium on Wearable Computing, 2002

[3] N. Beckmann, H. P. Kriegel, R. Schneider, B. Seeger - The R-Tree: An Efficient and Robust Access Method for Points and Rectangles, SIGMOD Conference, 1990*

[4] R. Bekkerman and M.Sahami - Semi-supervised Clustering using Combinatorial MRFs, In Proceedings of ICML 2006 Workshop on Learning in Structured Output Spaces, 2006

[5] Claudio Bettini, S. Jajodia, Sean X. Wang - Time Granularities in Databases, Data Mining, and Temporal Reasoning, Springer

[6] Claudio Bettini, Sean X. Wang, S. Jajodia, Jia-Ling Lin - Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences, IEEE Transactions on Knowledge and Data Engineering, 1998

[7] H. Cao, N. Mamoulis, and D. W. Cheung - Discovery of Periodic Patterns in Spatiotemporal Sequences, IEEE Transactions on Knowledge and Data Engineering (TKDE), 2007

[8] Inderjit S. Dhillon, Subramanyam Mallela, Dharmendra S. Modha - Information-Theoretic Co-clustering, Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003

[9] L. R. Dice - Measures of the amount of ecologic association between species, Ecology, 1945

[10] Mohamed G. Elfeky, Walid G. Aref, and Ahmed K. Elmagarmid - Using Convolution to Mine Obscure Periodic Patterns in One Pass, EDBT, 2004

[11] Mohamed G. Elfeky, Walid G. Aref, and Ahmed K. Elmagarmid - Periodicity Detection in Time Series Databases, IEEE TKDE, 2005

[12] *Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu - A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, 1996*

[13] *S. Guha, R. Rastogi, K. SHIM - CURE: An efficient clustering algorithm for large databases, In Proceedings of the ACM SIGMOD Conference, 1998*

[14] *Antonin Guttman - R-Trees: A Dynamic Index Structure for Spatial Searching, Proc. of ACM SIGMOD International Conference on Management of Data, 1984*

[15] *R. V. Hamming - Error detecting and error correcting codes, Bell Sys. Tech. Journal, 1950*

[16] *J.Han, W.Gong, Y.Yin - Mining segment-wise periodic patterns in time-related databases, In Proc. of Intl. Conf. on Knowledge Discovery and Data Mining, 1998*

[17] *J.Han, G.Dong, Y.Yin. - Efficient mining of partial periodic patterns in time series database, In Proc. of International Conference on Data Engineering, 1999*

[18] *J.Han, M. Kamber - Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 2001*

[19] *P.Indyk, N.Koudas, S.Muthukrishnan - Identifying representative trends in massive time series data sets using sketches, VLDB, 2000*

[20] *P. Jaccard - Nouvelles recherches sur la distribution florale, Bulletin de la Societe Vaudoise de Science Naturelle, 1908*

[21] *G. Karypis, E.H. Han, and V. Kumar - CHAMELEON: A hierarchical clustering algorithm using dynamic modeling, IEEE Computer 32, 1999*

[22] *Leonard Kaufman, Peter J. Rousseeuw - Finding Groups in Data: An Introduction to Cluster Analysis, Wiley-Interscience*

[23] *S. Kulczynski - Die pflanzenassoziationen der pieninen, Bulletin International de l'Academie Polonaise des Sciences et des Lettres, 1927*

[24] *J. B. MacQueen - Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967*

[25] *N. Marmasse and C. Schmandt - Location-aware information delivery with com-Motion, HUC, 2000*

[26] *F. Murtagh - Multidimensional Clustering Algorithms, Physica-Verlag, 1985*

[27] *R. Ng , J. Han - Efficient and effective clustering methods for spatial data mining, In Proceedings of the 20th Conference on VLDB, 1994*

[28] *C. Olson - Parallel algorithms for hierarchical clustering, Parallel Computing, 1995*

[29] *Banu Özden, Sridhar Ramaswamy, Abraham Silberschatz - Cyclic Association Rules, In Proc. of International Conference on Data Engineering, 1998*

[30] *D. Pelleg and A. Moore - X-means: Extending K-means with Efficient Estimation of the Number of Clusters , In Proceedings of ICML, 2000*

[31] *D. J. Rogers and T. T. Tanimoto - A computer program for classifying plants, Science, 1960*

[32] *P. F. Russell and T. R. Rao - On habitat and association of species of anopheline larvae in southeastern Madras, J. Malar. Inst. India, 1940*

[33] *G. Salton - On the Use of Term Associations in Automatic Information Retrieval, Proceedings of COLING-86, 1986*

[34] *Jorg Sander, Martin Ester, Hans-Peter Kriegel, Xiaowei Xu - Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications, Data Mining and Knowledge Discovery, 1998*

[35] *R. R. Sokal and C. D. Michener - Statistical method for evaluating systematic relationships, University of Kansas Scientific Bulletin, 1958*

[36] *J.Yang, W.Wang, P.S.Yu - Infominer:Mining surprising periodic patterns, In Proc. of 7th Intl. Conf. on Knowledge Discovery and Data Mining, 2001*

[37] *J.Yang, W.Wang, P.S.Yu - Infominer+: Mining partial periodic patterns with gap penalties, In Proc. of the 2nd IEEE Intl. Conf. on Data Mining, 2002*

[38] *G. U. Yule and M. G. Kendall - An Introduction to the Theory of Statistics, 14th ed. Hafner, 1950*

[39] *Tian Zhang, Raghu Ramakrishnan, Miron Livny - BIRCH: an efficient data clustering method for very large databases, Proceedings of ACM SIGMOD international conference on Management of data, 1996*

[40] *Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, Loren Terveen - Discovering Personal Gazetteers: An Interactive Clustering Approach , Proceedings of the 12th ACM Intl. Symp. on Advances in Geographic Information Systems, 2004*

[41] *Changqing Zhou, P. Ludford, D. Frankowski, and L. Terveen - An experiment in discovering personally meaningful places from location data, In Proc. CHI, 2005*