Mining Periodic Patterns in Spatio-temporal Sequences at Different Time Granularities*

Sezin Karli †, Yucel Saygin ‡

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 extract patterns of the finest time granularity, and therefore overlooks potential patterns available at coarser time granularities. In this work, we propose two 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.

Keywords: Data Mining, Spatio-temporal Data, Time Granularity, Periodic Pattern

1 Introduction

Our daily lives contain several routines. Some of these routines can be visits made to places such as work, home, favorite restaurant, and so on. These visiting habits are generally available for most of the moving entities. The trajectories

*This work was partially funded by the Information Society Technologies Programme of the European Commission, Future and Emerging Technologies under IST-014915 GeoPKDD project

†E-mail: sezinkarli@su.sabanciuniv.edu

‡Corresponding author. Address: Sabanci University, Faculty of Engineering and Natural Sciences, Orhanli, 34956, Tuzla, Istanbul, TURKEY, E-mail: ysaygin@sabanciuniv.edu, Phone: +90 (216) 483 95 76, Fax: +90 (216) 483 95 50

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, 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 Scholz'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. Consider another person –Brad– who visits his parents living in France at approximately the same time of the year for a week. It is probable that this visit will not contain frequent patterns in finer granularities because, for example, Brad will not visit Notre Dame de Paris for 5 days of the week at the same hour or he will not 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 this pattern 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 using *week* granularity and 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 paper, we work

with the spatio-temporal sequence of a single moving 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 need to 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. During the discretization process, we omit the time information related to finer granularites. This choice has a logical argument behind it. In our daily life, moving to 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 Topkapi Palace, we are concisely saying "Rick and Nielsen visited Topkapi Palace on Monday". This statement does not use the finest granularity although this information was available and it obviously does not contain at which exact time interval they did the visit, because our intention in using the *day* granularity was to disregard details pertaining to the finer granularities.

Two techniques that use different matching schemes are proposed in this paper:

1. MINIM - periodic pattern MINer using exact IMportant places information

2. μ-PIN - periodic pattern Miner Using approximate important Places INformation

Both techniques make use of the "important place" concept which is defined in Section 3. MINIM does exact matching of important place contents while μ-PIN use the similar matching. The reason of allowing this kind of approximation to μ-PIN is the high probability of obtaining patterns of very low support while using MINIM. Experimental results show that the proposed techniques

are accurate and efficient.

Rest of the paper is organized as follows. Section 2 contains the necessary background information and summarizes the related work in the literature. Section 3 provides the overview of our methods, the definitions of time related concepts, preliminary definitions and the problem definition. The mining of periodic patterns at different time granularities is explained in Section 4. Section 5 contains the conducted experiments and the last section concludes the paper.

2 Background and Related Work

2.1 Clustering Techniques Used

Clustering is the task of grouping similar objects such that we obtain high intraclass similarity and high interclass dissimilarity. Two different clustering algorithms are used in our work: DBSCAN [14] and AGNES [23] which are explained in the following subsections.

2.1.1 DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a widely used density-based clustering technique. DBSCAN needs two input parameters: EPS and $MinPTS$ [14].

1. EPS (epsilon) 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 then both clusters will be merged into one cluster.

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 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 clusters. Furthermore, it is powerful in handling arbitrary shapes which is very important in

our task (extraction of important places by the clustering of location measurements). Its time complexity is only $O(n\ logn)$ with a spatial index which we also utilized.

2.1.2 AGNES

AGglomerative NESting is a bottom-up hierarchical clustering algorithm. Initially every single sample is a cluster. Dissimilarities between every pair of clusters are calculated and then the merging phase begins. Most similar clusters are merged and then the new dissimilarities between the newly created cluster and the other clusters are calculated. After that, the merging occurs again. The algorithm continues this operation (dissimilarity calculation and the merging) until the dissimilarity between the most similar clusters becomes larger than the stopping criteria or until the desired number of clusters is reached.

We decided to use AGNES for clustering of bit vectors since it allows the usage of any dissimilarity function without needing an extension to its base algorithm.

Cluster dissimilarities are calculated by a dissimilarity function, but normally clusters contain more than one sample. As the dissimilarity functions offer only the dissimilarity between sample pairs, we will surely need a linkage metric ([26], [27]) to calculate the distance between two clusters.

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

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

where C_i and C_j are two different clusters and $dissimilarity()$ is the dissimilarity 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.

2.2 Related Work

Extensive research has been conducted on periodic pattern mining. Han et. al. [19] propose algorithms for mining partial periodic patterns from time series data. In [20], the ideas proposed in [19] are further extended and an efficient and scalable algorithm that uses a novel tree structure (max-subpattern tree) is proposed.

In [33], a technique for mining periodic patterns from event sequences is proposed which was later extended in [34] for allowing the mining of partial periodic patterns. In [12] and [13], authors propose methods for detecting periodicity in the event sequences by using convolution-like formulas which were not considered in earlier studies.

In a recent work, Cao et al. propose algorithms for mining periodic patterns in spatio-temporal sequences. They describe a discretization method for location information and then use an extended form of the technique proposed in [20] 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 addressing this problem.

In [15], algorithms for mining sequential "important place" patterns with similar time constraints between visits are proposed. Two of the algorithms that are proposed in [15] are extensions of the work in [16]. In our work, we similarly choose to use the concept of important places for the discretization of the location data, but notice that we mine periodic patterns and consider

different time granularities. 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 place. They assume that a signal loss shows that the object is in a building, but they overlook 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 runs low or a disconnection from the satellite occurs). Furthermore, open area important places will be missed with this approach. In [1], 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 [1] is outperformed by the algorithm proposed in [36] where authors apply a density-based clustering algorithm (DJ-Cluster) to the spatial data for the extraction of important places. The importance of [36] is that the authors offer metrics for the evaluation of performance for the extraction of important places. Notice that studies before [36] are not evaluated for their performance of important places discovery. [37] contains experiments conducted on 24 people of different life stages using the technique in [36]. In our work, we use a density-based clustering method and respect periodicity and time information of coarser granularity while extracting important places.

3 Problem Formulation

In this section, we formulate the problem that we chose to address. First, we will concisely give the problem definition and then the overview of our algorithms together with the definitions pertaining to temporal concepts. After that, the preliminary definitions will be provided and later, the problem will be defined in detail.

Our problem is to discover patterns from a spatio-temporal sequence which occur with a frequency greater than a threshold at a given time granularity and period. First, we partition the spatio-temporal sequence to different periods. For instance, if the time granularity is *day* and we are working with a period of

7, then we collect each Monday in a single bucket and continue the collection until there are 7 buckets in total. Later, we process the location measurements and extract important places related to the corresponding position of the period. For instance, we take the location measurements of Mondays and extract important places for Mondays. After that, each element of the granularity in the spatio-temporal sequence is processed such that every element can be represented in terms of important place visits. For instance, the first Monday is represented by a bit vector which depicts existence and non-existence of visits to important places extracted from the Monday bucket. Later, each position of the period is inspected separately and similar representations are grouped together by clustering of bit vectors and they are labeled. For instance, if the first and second Mondays are similar with respect to visited places, then they will be grouped together and tagged with the same label. After that, the initial spatio-temporal sequence can be represented with a smaller sequence that is discretized and then the new sequence can be easily mined with previously proposed techniques.

3.1 Temporal Concept Definitions

In this work, we adopt the temporal concepts defined in [4]. 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 *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 *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 3.2 *A granularity g is finer than a granularity h if for each index i 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 3.3 *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 3.2 *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 3.4 *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 $g(z)$ of the bottom granularity g can be mapped to a unique tick $h(z')$ of one of the granularities h in the time granularity set. $\lceil z \rceil_g^h = z'$ denotes the granularity conversion 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 given, we will now give preliminary definitions that will be needed throughout the paper.

3.2 Preliminary Definitions

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 year 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 [5]. In addition, we will use the expression "coarser granularity" instead of "a new granularity coarser than the bottom granularity".

Definition 3.5 *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 there is no timestamp $t' \notin TS_k^g$ such that $\lceil t' \rceil^g = k$.*

Example 3.3 *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 in this case. Notice that the index of the time set begins from 0 just like the index in the time granularities.*

We assume that there is a location measurement for each timestamp in S, so it is possible to denote the location measurement corresponding to a timestamp t_i by l_i.

Definition 3.6 *For a coarser time granularity g, the location set of index k is defined as $LS_k^g = \{l_i \mid t_i \in TS_k^g\}$.*

Example 3.4 *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 3.7 *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 3.5 *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 largest 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, two symbols (T, g) will be used throughout the paper 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 will not 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.

Example 3.6 *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 3.8 *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 a location set obtained using the notion of important places. We will briefly explain how this discrete representation describes the data.

Example 3.7 *Assume that we have a location set* LS_0^{day} *which is depicted in Figure 1. Three rectangles in the figure are highlighting the important places. We will obtain a discrete representation from the location set using important places.*

The discrete representations are bit vectors where the contained binary values are separated with "," and delimited by "<" and ">" . This discrete representation is obtained by inspecting the existence and non-existence of visits to important places. Notice that a binary value in the bit vector represents a visit (1) or a lack of visit (0) to the corresponding important place.

Example 3.8 *In Figure 2, we enumerated important places (*$IP0$, $IP1$ *and* $IP2$*). Location set* LS_0^{day} *will be represented with the bit vector* $< 1, 0, 1 >$, *because there is a visit to the zeroth (with index* 0*) and to the second important place (with index* 2*), but the first important place (with index* 1*) is not visited during the zeroth day.*

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 represen-

tation with the same index (i.e. $LS_i^g \rightarrow r_i$). Notice that r is the abbreviation of discrete representation.

Definition 3.9 *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.*

Segments of period of 4 can be seen in Figure 3.

Definition 3.10 *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 "*" which implies the possibility of any value for the discrete representation.*

Example 3.9 *"$r_1\{r_2, r_4\}*$" 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).

Definition 3.11 *In our first technique, a segment s complies with a pattern p if bit vector in s_i is the same with bit vector of p_i or $p_i = *$ (for each $i = 0, 1, 2, ..., T-1$). In the second technique, a segment s complies with a pattern p if bit vector in s_i is similar[1] to bit vector of p_i or $p_i = *$ (for each $i = 0, 1, 2, ..., T-1$).*

Definition 3.12 *A discrete representation set (DRS) is the set that groups all discrete representations of a single position of the period T. Formally, $DRS_z^g = \bigcup_{i\ mod\ T=z} r_i$ for all i available in S^g sequence.*

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

[1]The similarity concept will be defined later

Definition 3.13 *A periodic pattern of length k is called $k-pattern$.*

Definition 3.14 *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 3.10 *If $p = r_1\{r_2, r_3\}*$, then p has six subpatterns such as "$r_1 * *$", "$*r_2*$", "$*r_3*$", "r_1r_2*", "r_1r_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.

4 Mining of the Maximal Frequent Patterns

Mining of the maximal frequent patterns is completed 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 the proposed mining techniques can be seen in Figure 4.

4.1 Finding Frequent 1-Patterns

We will begin this part of the paper by explaining two steps that take place before the discretization process. These two steps are (i) elimination of location measurements belonging to movement with high speed, (ii) extraction of important places.

4.1.1 Extraction of Important Places

As our choice is to describe several location measurements in terms of important places visits, we need this phase which will give us the needed base for the generation of discrete representations in our techniques.

One of the most interesting work for the extraction of important places is [36] as we previously declared. In [36], a density-based clustering algorithm

(DJ-Cluster) is used for clustering the spatial data and clusters are treated as important places. In our work, we partition the spatio-temporal sequence such that the resulting datasets will respect the time information of coarser granularity and periodicity. After that, we apply DBSCAN instead of DJ-Cluster because both algorithms are comparable in performance and DBSCAN is widely used.

Only two parameter values (values for $MinPts$ 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. As our techniques do the mining at different granularities, setting $MinPTS$ to 3 ($MinPTS = 2k-1$ for data with k dimensions as suggested in [31]) does not give the best clustering result in all cases. On the other hand, we claim that EPS can be easily set in our case due to the fact that, in general, we want to find buildings as clusters –which occupies an average amount of area on the map– but value of the $MinPTS$ 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 since 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 consider the time granularity at the same time. For example, Robin visits the shopping mall every Friday night. The shopping mall probably will not 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 that we mine at *day* granularity with a period of 7 which means that we want to find similar Mondays or Tuesdays... etc. In the case of Robin's 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 since 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 *For n objects, the time complexity of DBSCAN is $O(n \log n)$ with a spatial index such as R-tree [17] and R*-tree [2]. In the worst case, we 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 do not 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 erratic 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 not distinguish the difference between the densities of "home" and "traffic light" which means that it will mark both of them as important places. But if we apply our preprocessing step, then the density near the traffic light will get lower which in consequence will help DBSCAN detect that the traffic light is not a region as dense as home.

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 measurement 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 40 km/h speed inside an important place such as home, work, golf club, pub, and so on. The proposed algorithm can be found as 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 obtain the idea used in the algorithm with a case study. Notice that [36] 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 the traced object may not stop in an important place. For instance, if Bob spends some time in the park, we can not 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 ← distance(l_0, l_1)
dist2 ← distance(l_1, l_2)
for i from 3 to n do
  if (dist1 > Threshold ∧ dist2 > Threshold) then
    removeFromD(l_{i-2})
  end if
  dist1 ← dist2
  dist2 ← distance(l_{i-1}, l_i)
end for
```

After joining every pair of location measurements belonging to 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 5(a)), (ii) short segment after long

segment (Figure 5(b)), (iii) long segment after short segment (Figure 5(c)), (iv) long segment after long segment (Figure 5(d)).

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 l_1) in this case study.

The first case is trivial since, if we omit l_1, it is obvious that we decrease the density of an important place. In the second case, we should not omit l_1 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 [36] is problematic. The third case is similar to the second case, but this time we cannot 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 the important place (depicted again as 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 short. The fourth case is the only case that allows the elimination of l_1 without the risk of losing necessary location measurements. Since this point is not close to any other neighbor point, we know that it cannot contribute to the density of an important place.

There is another issue to consider while we think about speeds of moving objects. 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 5(e)). 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 5(f) 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 5(f). Two points which delimits this short segment will be

treated like a part of movement with low speed and our method will not omit them. On the other hand, it is sure that any 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. Algorithm 1 has the time complexity of $O(N)$ where N is the number of location points.

Our preprocessing is more beneficial when the traced object travels with high speed most of the time. Generally the traced objects travel with high speed for a considerable amount of time which justifies our preprocessing step.

After the preprocessing step and the extraction of important places by applying DBSCAN to 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.

4.1.3 Mining Algorithms

We propose two algorithms 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

There is a chance that exact matching of binary features produces frequent 1-patterns with very low support, thus an approximation (such as we propose in μPIN) can be very useful.

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 will generate the discrete representations from the location sets. 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 generated, 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 extracted from LS_i^g.

Example 4.1 *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 the one with 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 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 above threshold will be labeled with "*". Thus, S^L will be obtained. Notice that label here refers 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 4.2 *MINIM will need two scans over all location measurements for this phase. 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 separate each (i) position of the period by using discrete representation sets (DRS_i^g) and group similar bit vector contents in these sets together. For the grouping step, we use a hierarchical clustering algorithm, 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 [18] with different normalizations (Sokal and Michener [32], Rogers and Tanimoto [28])

2. Normalized inner product with different normalizations (Russell and Rao [29], Jaccard and Needham [21], Dice [10], Kulczynski [24])

3. Correlation similarity measures (Yule and Kendall [35])

Definition 4.1 *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 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 this important place. For instance, "home" is an important place for Tom. We will see that every Monday Tom visits home. So the bit vectors representing Mondays 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. We will see why this approach can be problematic. For example, assume that we mine 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 adapted Jaccard metric (Positive cases are considered insignificant instead of negative cases). 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 do not 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 what if it was "shopping mall" which has 50% chance of visit? This leads us to question the definition of significance of positive and negative cases in our task.

Definition 4.2 *A match between bit vectors with a value in ith offset of the bit vector is insignificant if the value 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"[30] where a word gains more weight if it is seen in less documents.

Example 4.2 *Assuming that we compare different Mondays, it is usual to see the value* 1 *in the "work" offsets. 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 between these days.*

Before the dissimilarity measure, we give the contingency table which can be seen in Table 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 and less importance to insignificant ones.

$Dissim(x,y) = \frac{|b|+|c|}{|b|+|c|+\sum[2(1-\lambda_i)s_i+2\lambda_i t_i]}$

where s_i is 1 only when $x_i = 1$ and $y_i = 1$ (positive case), and t_i is 1 only when $x_i = 0$ and $y_i = 0$ (negative case). λ_i is the weight for ith offset.

It is easy to see that if we set all λ to $1/2$ (i.e. both negative and positive cases are equally important), our measure transforms into Sokal and Michener metric [32] while if we set all λ to 1 (i.e. the positive cases are totally insignificant), it transforms into Dice metric.

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 separately scanned for finding the occurrence rates of 1 in each offset, then these estimates are used as λ values. 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.

Example 4.3 *Assume that we have two Mondays to compare such as* $monday_1 = \{1, 1, 1\}$ *and* $monday_8 = \{1, 0, 1\}$ *and the estimates for the offsets are* $\lambda_0 = 1, \lambda_1 = 0.5, \lambda_2 = 0.4$. *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* $1 \ / \ (1 + 2 \times (1 - 1) + 2 \times (1 - 0.4)) \approx 0.45$. *As you can see, a matching on the zeroth offset (with a* 100% *chance of happening) has no influence on the similarity between bit vectors. On the other hand, the dissimilarity of the measure decreased considerably with a significant match in the second offset, because seeing a positive case in this offset is interesting considering the fact that "1"s has* 40% *chance of happening.*

After λ values are found, we can begin the clustering using AGNES. But first, we will try to come up with a method that could guide the user for stopping criteria selection. Assuming we use complete linkage, setting the optimal stopping value for the clustering task is difficult in adapted Jaccard distance, but that is not the same case with normalized Hamming distance. User can easily set 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 and there are 6 features in bit vectors, then $\frac{2}{6}$ will be the 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 run 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 this task, we will first take the maximum number of difference ($|b+c|$) allowed by the user (as in the normalized Hamming distance case) as input v. Later, using the estimates of every offset (λ values), we will find a lower bound for only one difference ($|b| = 1$ or $|c| = 1$) case and then an upper bound for the v difference case ($|b| + |c| = v$). So,

$[lowerbound, upperbound]$ will be our interval for the grid search.

First, we generate two sequences using λ values. 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 have the minimal effect on the decrease of dissimilarity. Using these weights and input v of the user, we will find the 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 objects available for the input v. The upper bound is calculated with $\frac{v}{v+2sum}$ where sum is the sum of $(k-v)$ smallest weights in min_seq supposing we have k offsets in bit vectors. In this section, we will use k as the total number of offsets in bit vectors (i.e. number of important places).

Now, a similar sequence (max_seq) has to be found for the lower bound. 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 do not match (i.e. if two bit vectors are x and y, $x_i = 0 \wedge y_i = 1$ or $x_i = 1 \wedge y_i = 0$). 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" constraint. 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 4.4 *Assume that* $\lambda_0 = 0.9$, $\lambda_1 = 0.5$, $\lambda_2 = 0.4$, $\lambda_3 = 0.8$, $\lambda_4 = 0.6$ *are the weights. Suppose that the user wants at most* 2 *differences in offsets (i.e. two*

offsets that does not match, $v = 2$). 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 3 $(v - k)$ smallest elements from min_seq. $\{0.1, 0.2, 0.4\}$ *are the smallest ones which will form sum. The value of the upper bound is* $\frac{2}{2+2(0.1+0.4+0.2)} = 0.59$. *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* $[0.15, 0.59]$ *using 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 $v = |b| + |c|$).

After the new interval for the stopping criteria is obtained, a grid search can be run in this interval. Then, the optimal parameter value can be chosen by using a cluster validation measure such as Dunn's index ([11], [6]) or Davies-Bouldin index ([8]). If the user has difficulties in setting v value, a grid search can be run for the whole $(0, 1)$ interval and the optimal parameter value can be chosen again with the help of cluster validation measures.

Using AGNES with complete linkage as the linkage function and our dissimilarity measure as the distance function, we cluster each DRS_i^g for i from 0 to $T - 1$ separately. After the clustering, 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. 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 4.3 *During the analysis of MINIM, we show that we need a single scan of database for obtaining bit vectors generated from locations sets. The estimation of λ_i values can be completed during this scan. After that, AGNES will be applied. We have $\frac{N}{h}$ bit vectors in total where $|S| = N$ and $|LS_i^g| = h$. We will split the total into T (period) parts and apply AGNES T times. As the complexity of AGNES for n objects is $O(n^2)$, the resulting complexity for the clustering is $T \times O\left(\frac{N^2}{h^2T^2}\right) = O\left(\frac{N^2}{h^2T}\right)$ and from $h = O(\frac{N}{T})$, the total is equal to $O(T)$.*

4.2 Mining of the Frequent Patterns

After frequent 1-patterns are extracted and the labeling is completed, we obtain S^L which is basically a sequence of labels. After that, we will follow the approach proposed in [20] for the construction of a special tree from the sequence S^L and then, we will apply the counting part of the algorithm of [7]. Notice that both of these steps are present in both techniques.

There are a few definitions taken from [20] which will be given here for the sake of completeness, 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 4.3 *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 "$\{a,c\}**$").*

For instance, assume that "$a**$", "$*b*$", "$*c*$" and "$**d$" are frequent 1-patterns. Then, C_{max} is $a\{b,c\}d$.

For holding patterns and their 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 6) 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 reason of this constraint. If the pattern is of length 2 then its direct children will be of length 1 which is not really helpful. Since we know the counts of frequent 1-patterns, we do not need to count them again.

The nodes of the tree has a parent link and child links pointing to the corresponding nodes.

First, the insertion of the label sequence into the max-subpattern tree (as in [20]) will take place. Later, traversal of the tree which extracts the frequent patterns (as in [7]) will be completed.

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 (pattern) that complies with 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 corresponding node is not created yet. So it will be created and initialized with count value equal to 1 and then its ancestor nodes will be added 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 insert the node "$*cd$", we link it to its parent "$*\{b,c\}d$", but "acd" is its parent too.

Definition 4.4 *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 [20].

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 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), both techniques return a set of rectangles depicting locations occupied by visited important places. A pattern of length 3 can be seen in Figure 7. Each subfigure is a single position of the period and each rectangle in subfigures depict visited important places.

5 Experiments

In this section, we present our experiments conducted on synthetic data. In Subsection 5.1, we explain our synthetic data generator. In Subsection 5.2, performance gain of the preprocessing step is analyzed and preprocessing step's safety net behavior (for erratic parameter selections in DBSCAN) is observed. This subsection concludes with our experiment on possible loss of information due to the preprocessing step. In the subsequent subsection, proposed binary dissimilarity measure is compared with two popular distance metrics for their precision and recall in clustering using AGNES. In the subsequent subsection, gain in grid search with our interval narrowing technique is evaluated. In Subsection 5.5, the impact of different representations on the accuracy of important place matching is evaluated. After that, compactness of different representations is evaluated by comparing the total areas of discrete representations. In Subsection 5.7, we evaluate our methods' sensitivity towards their parameters. Subsection 5.8 evaluates the effectiveness of the proposed techniques. The last subsection of this section evaluates the efficiency of the proposed techniques. The overview of all conducted experiments can be seen at Table 2.

All experiments are conducted on a notebook that runs Windows XP and with 1.5 GHz clock speed and 512 MB of RAM. 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 for privacy reasons, 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 update it with "waiting time in important places" and "minimum probability of visiting important places" additions. As in the 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 real life data. With the exception of "home", it is rare to see a visit to an important place in every location set of the same position of the period. The updated 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". Furthermore, positions that contain visits to important places in the bottom granularity are called "periodic slots" while "non-periodic slots" denote positions that does not contain important place visits.

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 our "waiting time in important places" and "minimum probability of visiting important places" extensions. For the sake of completeness, the gener-

ator from [7] is given in detail next to our extensions. 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 is very similar to real life behavior, because we assume that, in general, $\frac{3}{4}$ of our day is spent in important places. Consider two cases; a student which spends 8 hours at dorm, $5-6$ hours at class, and $1-2$ hours at his favorite coffee shop. Assuming these places are important places for him, approximately 75% of his day is spent in important places. Now consider a worker whose days generally consist of 8 hours spent at home and 9 hours spent at the office. Again, approximately 75% of his day is spent in important places.

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, because in daily life, we spend $[2, 7]$ hours in most of the important places such as home (probably $7-8$ hours), class (probably 2 to 6 hours), favorite pub (probably 2 to 4 hours), gym (probably $1-2$ hours) and so on. Although cultural differences exist in this case (such as Turks spending much more time in a coffee shop than Americans, or French spending much more time in a restaurant than Koreans), we choose using the above interval.

After the delay values are set, the pattern centers for periodic slots ($Center_{P_i}$) are created randomly. A pattern center can be considered as the centroid of an important place. The distance 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 (which means 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 number of locations (equal to the delay value of the slot) in α

neighborhood of $Center_{P_i}$ are randomly chosen as location measurements. This part tries to simulate a visit to an important place and does not force an exact location for each visit. In a real life example, it can be thought as your visit of home which contains times spent in the kitchen, living room so on.

- If the actual slot P_j is a non-periodic one, then the object moves towards the pattern center of the next periodic slot (the centroid of the next important place). To obtain 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 step size $ss = \frac{distance(Center_{P_h}, Center_{P_m})}{|t_{P_h} - t_{P_m}|}$ is calculated. ss is the Euclidian distance between previous and next pattern centers divided by the temporal distance between the periodic slots. The object moves towards the next pattern center ($Center_{P_m}$) with step size of ss. For abstaining from generating undesired periodic patterns using the step size and its current angle, we will distort the angle by adding to it a random degree 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. In a real life example, this move can be thought as a travel from home to work.

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 different types of day content such that, in total, they can form the desired number of segments (nos). 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 (of coarser granularity) 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 8. Densely populated regions are important places for this position of the period.

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 potential errors that can occur during the selection of parameters for DBSCAN. 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 observe the safety net behavior by giving wrong parameter values to DBSCAN on purpose. EPS parameter of DBSCAN is set to 6 and $MinPTS$ is set to values such as 3, 5, 7, 10 although we knew that these $MinPTS$ values are far from the optimal value ($MinPTS$ should be close to 20). As a result, lots of location points which are not related to important places are tagged as if they are. As it is obvious, working with these badly formed clusters will be problematic in many ways. Later, we apply our preprocessing method to the same data set and omit location measurements of high speed movement and then apply the clustering. Even with low $MinPTS$ values such as 3, we see that every single location point tagged as a part of important place is in reality part of an important place which proves our preprocessing method's

safety net behavior.

There are two other evaluations that needs to be done. The gain in performance and the loss of accuracy due to our preprocessing has to be calculated. We previously stated that the gain in performance depends totally on the percentage of high speed travels in the 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 are generated using *day* as the time granularity, 4 as the period value ($|S| = 3072$). Later, we try different values of "total time spent on important places" and calculate the gain in performance in the extraction of important places phase. The results can be seen in the Table 3. 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 with high speed movement as this phase will be beneficial. Two extremities in percentage are 35% and 80%, because less than 35% (less than 8 hours spent in important places; even an 8 hours sleep at home is not possible) and more than 80% of time spent on important places does not seem to mimic real life behavior. In Subsection 5.1, we argued that 75% is the probable percentage for most people.

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 that are spatially contained in important places. These location measurements will be used as the testset. Later, we apply our preprocessing and do the extraction of important places on the previous data set. Then we check if the results after the extraction match with the testset. Comparison of the result with the testset shows no loss in the location measurements. 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 difficulty with this evaluation is that 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.

In our data generator, the overall frequency of 1s in a single offset depends on the input value and all offsets are supposed to be uncorrelated. That is, we give frequency values to different offsets (important places) as input and these frequencies do not have an effect on each other.

The bit vector data is generated. Later, AGNES is run with two popular binary metrics and our dissimilarity measure. After that, the results will be inspected with the right clustering that is done manually. To simplify manual clustering step, we generated a small dataset of 20 bit vectors. Four columns with different 1 frequencies are generated. The frequency of 1s is 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 "ith 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 clustered them by hand. During this process, we used the information of the dominant value in each offset instead of the complete estimation for its 1 frequency. 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 suppose that 0 (1) is dominant in this offset. We group bit vectors with less number of $|c| + |b|$ and with 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 1). For instance, if the dominants of the offsets are "$1, 1, 1, 0$" in the respective order and we have $monday_1 = \{1, 0, 0, 0\}$ and $monday_8 = \{0, 0, 0, 0\}$, then we can tell that these Mondays are really similar.

That is because they only got one non-matching offset and the offsets that match have a large influence to the similarity 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 the logic of single linkage, because single linkage will help the manual clustering with its chaining phenomenon.

Adapted Jaccard distance ($\frac{|b|+|c|}{|a|+|b|+|c|}$) and normalized Hamming distance are chosen for the evaluation task as they are the most popular metrics among two families of binary dissimilarity measures. AGNES is used as the clustering algorithm. The plan was to observe the final clustering results and compare them to the "right" clustering that we did manually. For the evaluation purpose, we use micro-average-precision and micro-average-recall measures ([9]).

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

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

FN_i: 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 [3]. First, we give cluster labels to the clustering results. During this labeling, we choose a labeling that, as a result, will increase the TP value as much as possible. It is done 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.

Precision and recall values of three dissimilarity measures can be observed

at Table 4. Normalized Hamming distance and adapted Jaccard distance suffer from the facts that are told previously, so their less successful result in recall is not surprising.

5.4 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 $\lambda_i \in [0.4, 0.9]$ are randomly generated for k important places. $[0.4, 0.9]$ interval is chosen to mimic real life; we assume that visits to important places will probably not happen with frequency more than 90% (close to 100% for home but it is very difficult to find a similar example) 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. The first column of the table is k values and the first row is v values. Notice that the gain percentages are calculated by $\frac{(upperbound - lowerbound)}{(1-0)} \times 100$ where the denominator is the length of initial interval $[0, 1]$ and $v \leq k \times \frac{3}{5}$ because it is not logical otherwise.

5.5 The Impact of Different Representations on the Accuracy

In this subsection, we measure the performance of different representations 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 and the extraction of important places are performed. After that, the generation of different representations takes place.

Our initial periodic pattern mining techniques were based on geometric rep-

resentations. After location points which are not spatially contained in the important places are omitted (due to the fact that they represent redundant points because they are related to rarely taken trajectories), our techniques were generating a minimum bounding rectangle or a convex hull from elements of each location set. So it was possible to represent a location set with a geometry which depicts the place within the traced object travels. As exact matching of generated geometries during the counting phase would cripple our accuracy, we chose to group similar geometries by applying clustering with the geometry comparison metric we propose. The details of previous techniques can be found in [22].

After different discrete representations are generated by different techniques, 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 the technique which uses minimum bounding rectangles as discrete representations sometimes matches minimum bounding rectangles whose contents in terms of important places are different. Although false positives can be seen in this technique, false negatives are never encountered. On the other hand, the technique which uses convex hulls as geometric representations has both false positives and false negatives. 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 the technique which uses convex hulls has both kinds of errors, experimental results show that its precision and recall are better compared to the results of the technique which uses minimum bounding rectangles. That is due to the larger approximation error of minimum bounding rectangles compared to convex hulls.

Table 6 summarizes our techniques' precision and recall results of important

place content matching. 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. same bit vectors are grouped together which forms the right clustering. Using convex hulls is better than using minimum bounding rectangles in terms of precision and recall while using bit vectors (such as in MINIM) outperforms both of them. MINIM has 100% success in precision and recall, because it 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 in this experiment.

5.6 Compactness of Different Discrete Representations

Techniques that use geometric discrete representations (which are briefly explained in Subsection 5.5) and MINIM are to be compared for their compactness in their discrete representations' occupied area. For this purpose, 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 resulting discrete representations are summed. As discrete representations are directly related to frequent patterns, the summation of their area gives an idea about the compactness of the offered pattern information. The evaluation results are shown in Table 7. Patterns based on convex hulls are more compact than the patterns based on minimum bounding rectangles 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 the technique which uses convex hulls is not surprising too. As important places are not necessarily close to each other, a convex hull that tries to spatially contain them occupies

a much larger area than the sum of areas that the important places occupy.

5.7 Parameter Sensitivity of our Techniques

In this subsection, we conduct experiments about the sensitivity of our techniques. Two of proposed algorithms' parameters are observed: the stopping criteria and $MinPTS$. Other parameters which are EPS and min_sup are intuitive to set, therefore, they are not observed in this experiment.

5.7.1 The Stopping Criteria

This subsection contains two experiments regarding the stopping criteria parameter of our system. First, the results of the cluster validation indices with different choices of stopping criteria for AGNES are given. Dunn's index ([11], [6]) and Davies-Bouldin index [8] are used as cluster validation indices. Second, the change in mined patterns by our algorithm (μ-PIN) with the change of stopping criteria value is inspected.

For the first experiment, we use our bit vector generator (explained in Subsection 5.3) for generating 200 bit vectors with $\lambda_i \in [0.4, 0.9]$. With this approach 9 data sets are generated. Each of them contains a different number of important places (from 2 to 10). Notice the similarity of parameter values for the data generator with the values used in Subsection 5.4. After the generation of data, AGNES is applied with different stopping criteria values and the clustering quality is calculated using two indices.

For a partition U and the ith cluster of this partition (X_i),
Dunn's validation index:

$$D(U) = \min_{1 \leq i \leq c} \left\{ \min_{1 \leq j \leq c \ and \ j \neq i} \left\{ \frac{\delta(X_i, X_j)}{\max_{1 \leq k \leq c} \{\Delta(X_k)\}} \right\} \right\}$$

Davies-Bouldin validation index:

$$DB(U) = \frac{1}{c} \sum_{i=1}^{c} \max_{i \neq j} \left\{ \frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i, X_j)} \right\}$$

Δ is the intracluster dissimilarity and δ is the intercluster dissimilarity. For the intracluster dissimilarity we use the complete diameter ($\Delta(X_i) = max\{dissimilarity(y, z)\}$ where $y, z \in X_i$) and for the intercluster dissimilarity we use complete linkage (because we used complete linkage in AGNES). Table 8 and 9 contain results calculated with Dunn and Davies-Bouldin indices respectively. In both tables, the first column is the number of important places and the first row is the stopping criteria. The bold values in tables highlight the optimal stopping criteria for μ-PIN. In Dunn's index, large values are better, while in the case of Davies-Bouldin index it is the contrary. During the usage of μ-PIN, we recommend either using the v value (the maximum number of different offsets allowed by the user) and doing a grid search in the reduced interval. After that, finding the optimal stopping criteria will be trivial if we use a cluster validation measure. Another possibility is to directly begin the grid search and again choose the optimal stopping criteria by considering results of cluster validation measures.

For the second experiment, we use our trajectory data generator. We generate 8 segments with 50% as the minimum probability of visit for important places, $T = 3$, and *week* as time granularity ($|S| = 4032$). In μ-PIN, EPS is set to 6, $MinPTS$ to 33, and $min\_sup = 0.4$. μ-PIN is run with different stopping criteria and Dunn's index is calculated for each run. After that, extracted patterns are inspected. Table 10 contains the values of Dunn's index for each run of μ-PIN. The first column of the table is different positions of the period and the first row is different values of stopping criteria. After a table such as this is obtained, it is easy to set the stopping criteria value for each position of the period.

After each run of μ-PIN, we inspect the extracted patterns.
For $stopping\ criteria = 0.2$, μ-PIN could not find any patterns because, as the stopping criteria is low, μ-PIN's "grouping of similar bit vector" phase did an exact matching of bit vectors instead of an approximate matching. Thus, the support of patterns in the data becomes really low (lower than 0.4). That

example supports our idea on MINIM's potential shortcomings and the necessity of μ-PIN. For *stopping criteria* = 0.4, μ-PIN could not find any patterns. Although there are patterns with higher support compared to the first run, they are still less than the threshold. For *stopping criteria* = 0.6, μ-PIN finds a 2-pattern with 50% support. For *stopping criteria* = 0.8, μ-PIN finds two 2-patterns with 50% support each. But as the results in Table 10 suggests, it is not wise to use 0.8 as the parameter value. After we inspect the clustering of bit vectors, we realize that the intracluster dissimilarities are much higher than the ones obtained after the run with *stopping criteria* = 0.6. i.e., very dissimilar bit vectors depicting important place visits are grouped together.

5.7.2 MinPTS Parameter

This subsection contains the experiments pertaining to $MinPTS$ parameter of our methods. In Subsection 5.2, we explain our experiments on $MinPTS$ values diverging from the optimal value by decreasing. We observe that our preprocessing method acts as a safety net in these cases and that all important places are extracted even though $MinPTS$ values much lower than the optimal one are used. In this subsection, our algorithms are inspected for their behavior in extraction of important places phase for increasing $MinPTS$ values (always taking larger values than the optimal one). First, the change in the number of points (labeled as visits to important places) and later the change in the number of important places is observed. After that, the change in extracted patterns is inspected.

For experiments with MINIM, we generate trajectory data with our data generator. 4 segments with period of 3, *week* as the granularity, 100% as the minimum probability of visiting important places are generated. Later, MINIM is run with $EPS = 6$, $min\_sup = 0.8$ and different $MinPTS$ values. We know that the optimal value of $MinPTS$ for this data set is 30 and the number of extracted important places should be 12 (4 for each position of the period). Table 11 contains the change in $MinPTS$ parameter and the re-

sults of extraction of important place phase. As EPS is fixed, larger $MinPTS$ values make $DBSCAN$ more selective for the extraction of important places. Therefore, each time $MinPTS$ increases, less important places are found. At $MinPTS = 130$, no important place is found. After this experiment, we observe the pattern extraction performance of MINIM on this data set. For values of $MinPTS \in \{30, 50, 70, 90, 110\}$, MINIM extracts a pattern of length 3 with 100% support. In each increase of parameter value, there is a decrease in the number of important places. Therefore, each extracted pattern contains a lesser number of important places which hints a loss of information. Only important places of large density (such as home, work) are left and patterns using these important places are extracted. At the run with $MinPTS = 130$, MINIM cannot find any patterns since no important place is extracted.

For the experiments with μ-PIN, we generate a similar data set to MINIM's with the difference in "minimum probability of visiting an important place" which is set to 75%. Later, MINIM is run with $EPS = 6$, $min\_sup = 0.4$, stopping criteria=0.6 and different $MinPTS$ values. We know that the optimal value for $MinPTS$ for this data set is 25 and the number of extracted important places should be the same with the previous experiment (12). Table 12 contains the change in $MinPTS$ parameter and the results of important place extraction. Again, as $MinPTS$ increases, less number of important places is extracted. After that, we observe the change in patterns of μ-PIN. In runs with $MinPTS = 25$ and $MinPTS = 45$, μ-PIN finds a pattern of length 2 with 50% support. But, as there are less important places in the case of $MinPTS = 45$, the pattern content of the second run is less informative. For $MinPTS = 65$, two 2-patterns are found: one with 50% support, and the other with 75%. This case has more patterns than the case of $MinPTS = 45$, because as number of important place decreases, it becomes easier to find similar bit vector contents because they are now more general. For the run with $MinPTS = 85$, a 3-pattern with 75% support is found. Again, it contains less information than patterns of the previous run. For the run with $MinPTS = 105$, a 2-pattern with 75% support

is extracted. In this run, it is impossible to obtain a 3-pattern even though the *min_sup* value is set really low. That is because, in one position of the period, no important place is found which disables the possibility of obtaining a pattern in this position. Similar to a case of MINIM, if a large value is given to $MinPTS$, no important place is extracted by μ-PIN and thus no pattern is mined.

5.8 Effectiveness of the Techniques

In this subsection, MINIM and μPIN are evaluated for their effectiveness on synthetic and real world data sets. Furthermore, the periodic pattern mining algorithm that can work on bottom granularity (STPMINE2v2 proposed in [7]) is inspected for its effectiveness in the same data sets.

5.8.1 Effectiveness on Synthetic Data Set

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 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 such as α point to the optimal parameters. *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 it is seen that they totally match. This result shows that MINIM is accurate.

After the evaluation of MINIM, we do the mining with STPMINE2v2 using 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 it can cause the mining of redundant patterns. Each position of the extracted patterns of STPMINE2v2 are important places that are included in 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 our important places are formed without the need of existing patterns in the bottom granularity. If our data generator was not generating patterns in bottom granularity (together with coarser granularity), then STPMINE2v2 could not find any patterns 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 completed as explained in Subsection 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 $EPS = 6$. In μ-PIN's bit vector grouping phase, we chose to use complete linkage as the linkage function. The results of μPIN completely match with the testset which implies that this technique is accurate.

After the evaluation of μPIN, we mine using STPMINE2v2 with the same dataset. STPMINE2v2 cannot find any patterns until min_sup is chosen as low as 0.2. 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 the reason of this is previously given.

5.8.2 Effectiveness on Real World Data Set

We also performed some preliminary experiments on a real data set[2]. The data set consists of the GPS location measurements of an anonymous student collected (with his consent) under the GeoPKDD project[3]. The time granularity of the data set is *second*. The data set contains 814962 location-timestamp pairs.

We used μ-PIN for mining patterns from the real world data set, since we only have 26 days of measurements (with a total of 7 days of gap between them and with nearly 60% of hours in those 26 days completely missing). For this limited data set, it is not possible to find patterns with MINIM which does exact matching. For the same reason, we extracted patterns of the *day* granularity in our experiments. Using 7 as the period (in conjunction with *day* granularity) would be intuitive, but there will be only 5 segments if we used this period value. So, we prefer to use $T = 3$ which give more than twice segments compared to $T = 7$.

We begin by setting the parameters of μ-PIN. EPS is set to 0.01 (which is close to 10 meters). We previously tell that EPS can be chosen rather easily, because we can find it by considering the average area that buildings occupy. After that, we try few values for $MinPTS$ and then set it to 70. This value is chosen, because after the extraction of important places phase, we see that $\frac{2}{3}$ of total readings were labeled as visits of important places and the number of extracted important places were close to what we predict considering the routines of our lives. After the extraction of important places is complete, we inspect these places. The cluster with largest spread (important place with largest area) has a diameter close to 90 meters which can be the work place of the traced person for instance.

After the extraction of important places is complete, we run μ-PIN with $min\_sup = 0.25$ (25%) and choose the results obtained by using the optimal stopping criteria value. We obtain the optimal value by considering values of

[2]the data set is available upon request
[3]http://www.geopkdd.eu

Dunn's and Davies-Bouldin indices. The values of cluster validation indices always pointed the same stopping criteria value during our experiment. After the run is complete, we obtain 4 patterns. A 3-pattern with 27% support, two 2-patterns with 27% support and, a 2-pattern with 45% support. The supports of the patterns may not seem very high, but considering how small our real world data set is, and the fact that we use 3 as period (work days are sometimes in the same position of the period with holidays) this outcome was quite predictable.

The following experiment is conducted using the same data set (which is processed such that there will be a single measurement per hour). STPMINE2v2's optimal parameters are chosen with grid search for each parameter and period is set to 24. Its minimum support is set to 20% which can seem low, but we previously explained the limitations of the data set in hand. STPMINE2v2 finds a 4-pattern and a 3-pattern with 21% support. After we observe the visited places of each pattern, we see that each non-* position in the patterns contain a visit to the same place. Later, we compare important places that our algorithm extracted to the place that STPMINE2v2 found and realize that one of our important places (out of 12) is matching with STPMINE2v2's place.

Experiments on the real world data set show that important places which cannot be extracted at finer granularities can actually be found at coarser granularities with our method. Important places extracted at coarser granularities are then used for finding patterns which cannot be observed at finer granularities. Therefore, we can safely say that μ-PIN does not need patterns in bottom granularity for mining patterns at coarser granularities. Furthermore, the experiment shows that a periodic pattern miner working on bottom granularity misses the existing patterns of coarser granularities.

5.9 Efficiency of the Techniques

Several 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 a low and 12 (which is generally used with *month* granularity) is a high value for period.

In Figure 9, the cost of techniques which use different discrete representations can be seen. MINIM is faster than the technique using minimum bounding rectangles because as MINIM uses bit vectors as representations and does an exact matching on these representations, it does not need a "grouping of similar geometries" step. The technique which uses minimum bounding rectangles is slightly faster than the one which uses convex hulls, because the convex hull building algorithm is more costly than minimum bounding rectangle building algorithm ($O(N \log k)$ versus $O(N)$ to be exact). Our experiments show similar costs for the phases after the extraction of frequent 1-patterns in all techniques which is the reason why we do not comment on their addition to the total cost. In Figure 10, the cost of μPIN can be inspected. Notice that MINIM's and μPIN's cost are less than quadratic. Both MINIM and μPIN are indeed efficient and scalable.

6 Conclusion

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

First, important places are extracted by the application of a density-based clustering technique. As the second step, remaining location points are used for obtaining concise discrete representations. 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 proposed in [20] and [7] for the extraction of the periodic patterns from the discretized sequence.

During our experiments, we use both synthetic and real world data sets. 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 using our preprocessing. Furthermore, our experiments show that none of the necessary location measurements are omitted by our 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 on the accuracy of our techniques is evaluated by the inspection of their matching accuracy of important place contents. Compactness of different representations are then evaluated. Later, the sensitivity of of our methods to their parameters are evaluated. The subsequent section evaluates the effectiveness of the proposed techniques. Both techniques are perfectly doing the desired discretization process and they are accurate in extracting the frequent periodic patterns from both synthetic and real world data sets. The last section contains a cost analysis for the proposed techniques. All our techniques are shown to be efficient and scalable.

As a future direction of research, an extensive study on several socioeconomic layers of society for obtaining an improved method for the extraction of important places should be conducted. Furthermore, the extension of our techniques with an effective method for the detection of optimal periodicity and granularity is needed.

References

[1] *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*

[2] 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*

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

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

[5] 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

[6] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity , IEEE Transactions on Systems, 1998

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

[8] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Recognition and Machine Intelligence, 1979

[9] 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

[10] L. R. Dice, Measures of the Amount of Ecologic Association Between Species, Ecology, 1945

[11] J. Dunn, Well separated clusters and optimal fuzzy partitions, J.Cybernetics Vol. 4, 1974

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

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

[14] 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

[15] Fosca Giannotti, Mirco Nanni, Dino Pedreschi, Fabio Pinelli, Trajectory Pattern Mining, KDD 2007, 2007

[16] Fosca Giannotti, Mirco Nanni, Dino Pedreschi, Efficient Mining of Sequences with Temporal Annotations, In Proceedings of SIAM Conference on Data Mining, 2006

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

[18] R. V. Hamming, Error Detecting and Error Correcting Codes, Bell Sys. Tech. Journal, 1950

[19] 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

[20] 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

[21] P. Jaccard, Nouvelles Recherches Sur La Distribution Florale, Bulletin de la Societe Vaudoise de Science Naturelle, 1908

[22] Sezin Karli, Mining Periodic Patterns in Spatio-Temporal Sequences at Different Time Granularities, Master of Science Dissertation, 2007

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

[24] S. Kulczynski, Die Pflanzenassoziationen Der Pieninen, Bulletin International de l'Academie Polonaise des Sciences et des Lettres, 1927

[25] N. Marmasse and C. Schmandt, Location-aware Information Delivery with ComMotion, HUC, 2000

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

[27] C. Olson, Parallel algorithms for hierarchical clustering, Parallel Computing, 1995

[28] D. J. Rogers and T. T. Tanimoto, A Computer Program for Classifying Plants, Science, 1960

[29] P. F. Russell and T. R. Rao, On Habitat and Association of Species of Anopheline Larvae in Southeastern Madras, J. Malar. Inst. India, 1940

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

[31] 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

[32] R. R. Sokal and C. D. Michener, Statistical Method for Evaluating Systematic Relationships, University of Kansas Scientific Bulletin, 1958

[33] 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

[34] 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

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

[36] 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

[37] Changqing Zhou, P. Ludford, D. Frankowski, and L. Terveen, An Experiment in Discovering Personally Meaningful Places from Location Data, In Proc. CHI, 2005

	0	*1*
0	a	b
1	c	d

Table 1: Contingency table

Subsection	*Content*	*Related part(s)*	*Bottom line*
5.2	Safety net for wrong MinPTS choices, Performance gain, Loss of accuracy	Subsection 4.1.2	Our preprocessing nullifies the sensitivity of our methods towards low values of MinPTS. Performance gain is up to 34%. No loss in necessary points is observed.
5.3	Bit vector clustering performance of different dissimilarity measures	Subsection 4.1.3	Proposed dissimilarity measure is performing better than two widely used measures on our type of data.
5.4	Gain in grid search by proposed method	Subsection 4.1.3	Our analytical method can result up to 90.93% of gain by narrowing stopping criteria's grid search interval.
5.5	Impact of different representations on the accuracy of important place content matching	Subsection 4.1.3 and [22]	Bit vector representation has better accuracy than geometric representations.
5.6	Compactness of different representations	Subsection 4.1.3 and [22]	Bit vector representation is more compact than geometric representations.
5.7	Sensitivity of our techniques to the parameters	Subsection 4.1.3	Giving a low value to stopping criteria can decrease the support of potential patterns and therefore several patterns can be missed. In the contrary case, our algorithm extracts patterns with very dissimilar bit vectors in the same non-* element of the pattern. If $MinPTS$ parameter is given a high value, then $DBSCAN$ becomes more selective and therefore extracts less important places. It is possible that our methods do not find any important places and thus any patterns.
5.8	Effectiveness of our algorithms and a periodic pattern miner which works at bottom granularity on synthetic and real world data sets	Section 4 and [7]	Our algorithms find the generated patterns of the synthetic data set while miner at the bottom granularity misses these patterns. In experiments with real world data set, our algorithm finds patterns which are again overlooked by the miner at bottom granularity.
5.9	Efficiency of our algorithms	Section 4 and [22]	Proposed algorithms' cost is less than quadratic.

Table 2: Overview of experiments

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

Table 3: Gain in performance with the proposed preprocessing method

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

Table 4: Precision and recall for dissimilarity functions

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

Table 5: The percentage of gain obtained by our analytical method

	Precision	*Recall*
Technique using mbrs	74%	72%
Technique using convex hulls	82%	77%
MINIM	100%	100%

Table 6: Precision and recall results for three techniques

MBR	*Convex hull*	*Bit vector*
4856045 units	1681503 units	7297 units

Table 7: Sum of areas of three different discrete representations

	s. criteria=0.1	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
imp. places=2	–	–	–	**2.15**	2.15	2.15	2.15	1.27	1.27
3	–	–	–	**1.73**	1.73	1.73	1.11	1.42	1.42
4	–	–	1.08	**1.33**	1.20	1.11	1.32	1.32	1.15
5	–	1.01	1.23	1.16	**1.27**	1.27	1.05	1.17	1.18
6	–	1.03	**1.31**	1.12	1.14	1.04	1.07	1.14	1.12
7	–	1.07	1.10	1.02	1.07	1.11	1.03	**1.16**	1.06
8	–	**1.29**	1.04	1.01	1.19	1.03	1.02	1.09	1.03
9	**1.14**	1.13	1.08	1.05	1.03	1.08	1.12	1.08	1.10
10	1.02	1	1.03	**1.09**	1.04	1.01	1.04	1.08	1.02

Table 8: Dunn's index values for several numbers of important places and different stopping criteria values

	s. criteria=0.1	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
imp. places=2	–	–	–	**0.37**	0.37	0.37	0.37	1.15	1.15
3	–	–	–	**1**	1	1	1	1.34	1.34
4	–	–	**0.9**	1.02	1.11	1.21	1.31	1.31	1.53
5	–	**0.56**	0.89	0.88	1.2	1.2	1.32	1.38	1.54
6	–	**0.61**	0.69	1.04	1.11	1.37	1.35	1.5	1.61
7	–	**0.76**	0.86	1.02	1.28	1.31	1.37	1.47	1.56
8	–	**0.84**	0.86	1.05	1.26	1.31	1.45	1.54	1.72
9	**0.17**	0.74	0.97	1.22	1.37	1.5	1.47	1.63	1.77
10	**0.25**	0.68	0.94	1.3	1.36	1.44	1.53	1.69	1.77

Table 9: Davies-Bouldin index values for several numbers of important places and different stopping criteria values

	s.criteria=0.2	*0.4*	*0.6*	*0.8*
Zeroth position	-	1.33	**1.5**	1.2
First position	-	-	-	**1.5**
Second position	-	**2.2**	**2.2**	1.25

Table 10: Dunn's index values for different stopping criteria values in different positions of the period

	MinPTS=30	*50*	*70*	*90*	*110*	*130*
important places	**12**	10	6	4	3	0
locations	**1435**	1267	868	651	448	0

Table 11: Results of (MINIM's) important place extraction for different $MinPTS$ values

	MinPTS=25	*45*	*65*	*85*	*105*	*115*
important places	**12**	10	7	3	2	0
locations	**1225**	1036	798	427	322	0

Table 12: Results of (μ-PIN's) important place extraction for different $MinPTS$ values

Figure Captions

1. Location points of the location set
2. Visits to two important places with label 0 and 2
3. Segments of S^g of period T
4. Illustration of the data mining process
5. Case study with two segments and possible outcomes of the linear interpolation between consecutive points:
   - Short segment after a short segment
   - Short segment after a long segment
   - Long segment after a short segment
   - Long segment after a long segment
   - The shortest path
   - A possible (and long) path
6. A max-subpattern tree
7. A frequent 3-pattern:
   - Zeroth position
   - First position
   - Second position
8. A complete trajectory of a single position of the period
9. Cost of techniques versus the number of segments
10. Cost of μPIN versus the number of segments

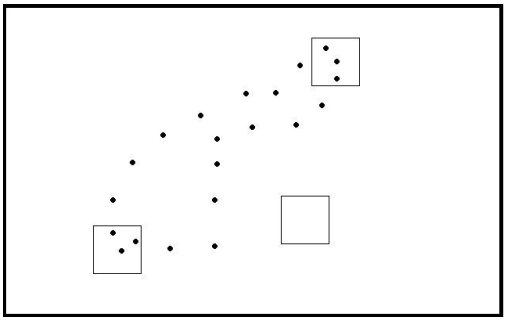


Fig. 1: Location points of the location set

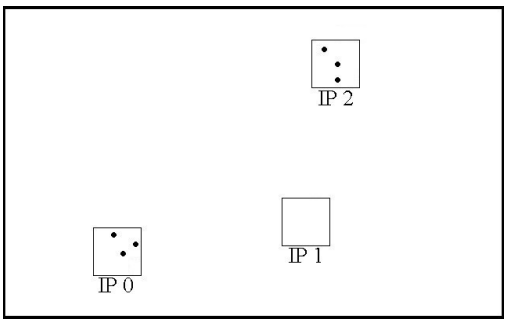


Fig. 2: Visits to two important places with label 0 and 2

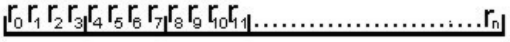


Fig. 3: Segments of S^g of period T

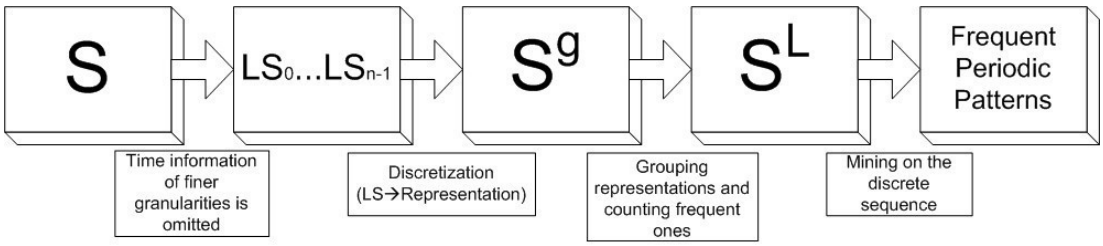
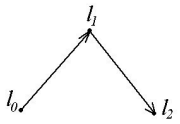
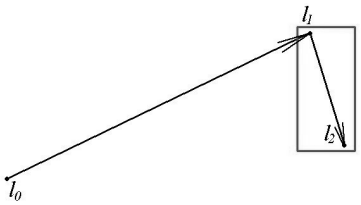


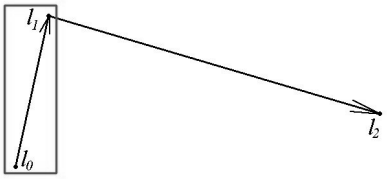
Fig. 4: Illustration of the data mining process



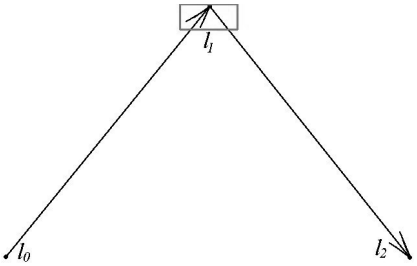
(a) Short segment after a short segment



(b) Short segment after a long segment

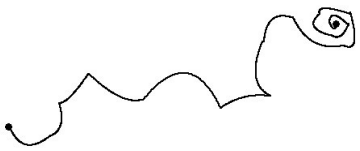


(c) Long segment after a short segment



(d) Long segment after a long segment

(e) The shortest path



(f) A possible (and long) path

Fig. 5: Case study with two segments and possible outcomes of the linear interpolation between consecutive points

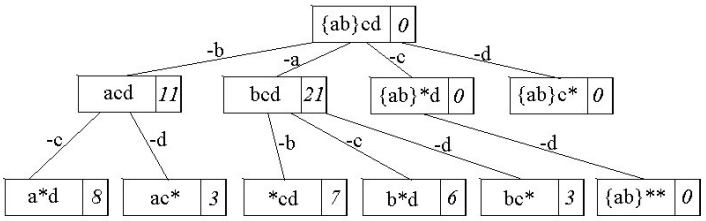
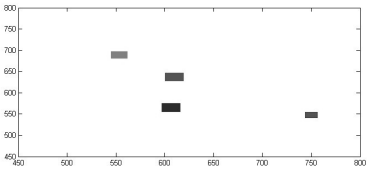
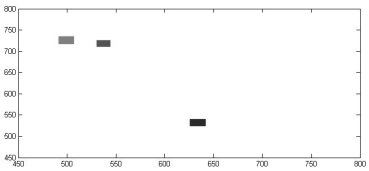


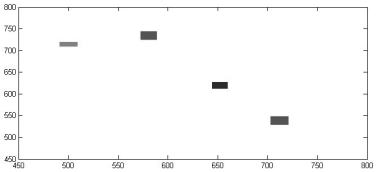
Fig. 6: A max-subpattern tree



(a) Zeroth position



(b) First position



(c) Second position

Fig. 7: A frequent 3-pattern

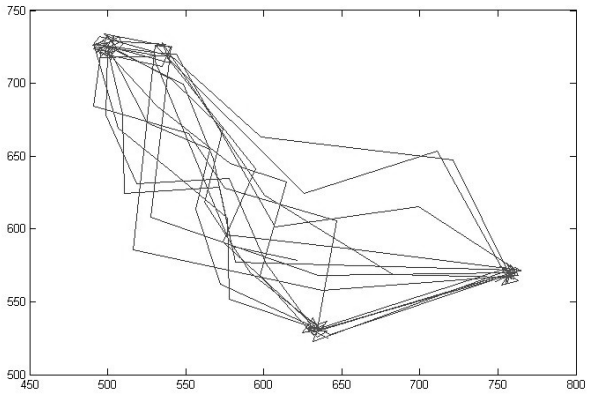


Fig. 8: A complete trajectory of a single position of the period

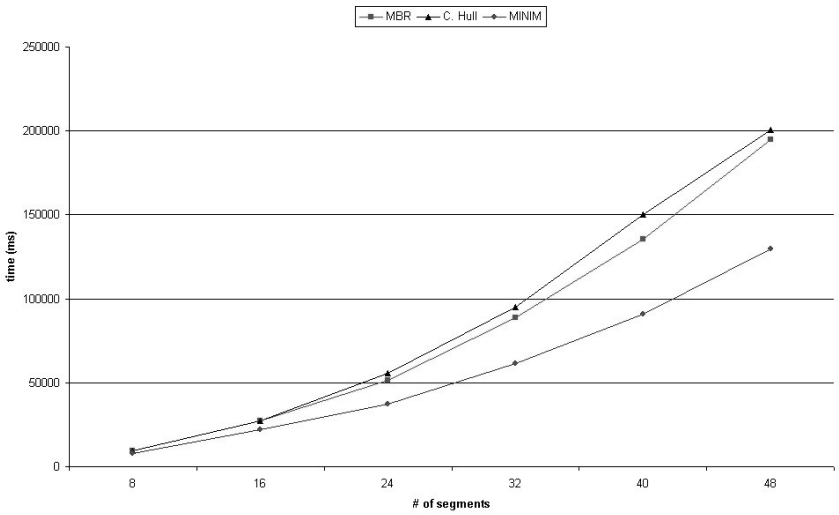


Fig. 9: Cost of techniques versus the number of segments

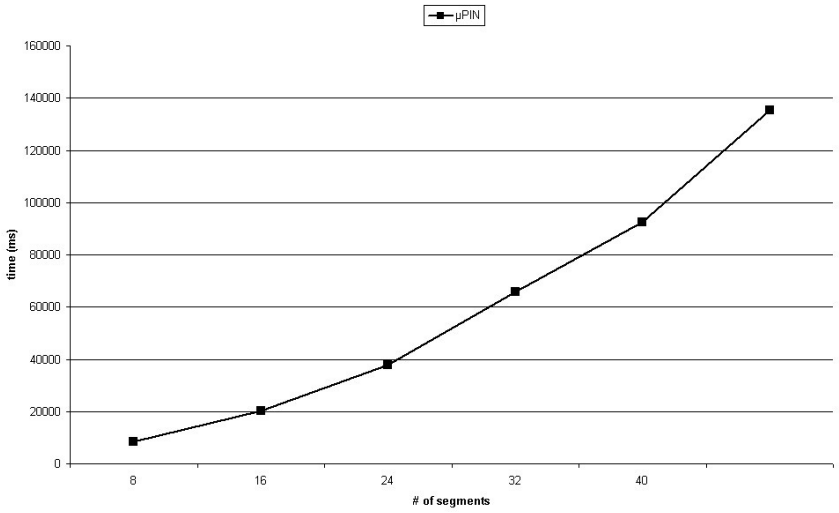


Fig. 10: Cost of μPIN versus the number of segments