2013 IEEE 14th International Conference on Mobile Data Management

Identifying Typical Movements Among Indoor Objects—Concepts and Empirical Study

Laura Radaelli* Dovydas Sabonis†
*Department of Computer Science
Aarhus University, Denmark
Email: {radaelli, csj}@cs.au.dk

Hua Lu† Christian S. Jensen*
†Department of Computer Science
Aalborg University, Denmark
Email: dovydas.sabonis@gmail.com, luhua@cs.aau.dk

***Abstract*—With the proliferation of mobile computing, positioning systems are becoming available that enable indoor location-based services. As a result, indoor tracking data is also becoming available. This paper puts focus on one use of such data, namely the identification of typical movement patterns among indoor moving objects. Specifically, the paper presents a method for the identification of movement patterns. Leveraging concepts from sequential pattern mining, the method takes into account the specifics of spatial movement and, in particular, the specifics of tracking data that captures indoor movement. For example, the paper's proposal supports spatial aggregation and utilizes the topology of indoor spaces to achieve better performance. The paper reports on empirical studies with real and synthetic data that offer insights into the functional and computational aspects of its proposal.**

I. Introduction

The identification of movement patterns is an interesting challenge, and both academic and industrial research is ongoing on this topic, which has a wide range of potential real-world applications. For example, frequent movement pattern techniques can be applied to historical records of hurricane tracking data in order to enable the prediction of the movements of emerging hurricanes [1]. Likewise, frequent pattern mining of outdoor trajectories is used in traffic control [2], ecological studies (observing animal movement and migration [3]), homeland security [4], and other location-based services.

With the continued deployment of indoor positioning systems, increasing amounts of indoor tracking data are becoming available; this enables the development of techniques for the identification of indoor movement patterns. Like in outdoor settings, indoor data mining techniques have a wide range of potential applications, covering such areas as targeted advertising, store location planning, resource management, activity monitoring, and security.

Indoor movement data differs from outdoor movement data. First, indoor and outdoor spaces are different. Indoor space is composed of entities such as walls, rooms, and doors that restrict and enable the movement of objects; this makes Euclidian distance inapplicable in indoor settings. Furthermore, indoor locations are typically symbolic, e.g., *room 355*, rather than pairs of coordinate values *(latitude, longitude)*. Second, indoor positioning systems differ from outdoor positioning systems and emit different positioning data. For example, GPS has become the *de facto* standard outdoor positioning system, but is generally not applicable indoors. In contrast, indoor positioning systems are typically based on widely available wireless radio technologies such as Wi-Fi, Bluetooth, and short range proximity sensor technologies such as RFID, or a combination of such infrastructures. Depending on the infrastructure(s) utilized, separate methods [5]–[8] exist that aim to enable indoor positioning.

Fig. 1 shows trajectories of people moving inside a building, as detected by a Bluetooth positioning system. The figure

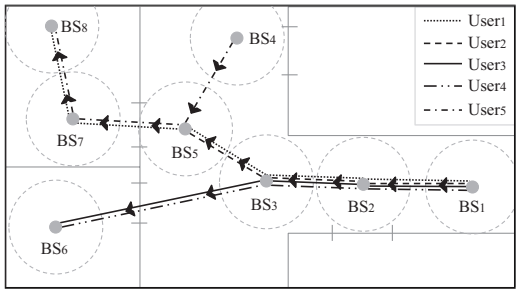


Figure 1. Running Example

depicts four rooms, each connected by a door to a common hallway. Eight Bluetooth base stations (BS_i) are deployed: five in the hallway and three in two of the rooms. A base station is depicted as a dot, representing the actual base station, surrounded by a dashed circle, representing the detection range of the base station. Users are detected by a base station when they are inside its range.

The example encompasses five users, represented by lines with different styles. We can see that the users go to different rooms, but are all passing through the hallway. If we define a path as being frequent when it is used by at least 40% of the users, we can identify these frequent patterns that are used by at least two users: $\langle BS_1, BS_2, BS_3, BS_5\rangle$, $\langle BS_1, BS_2, BS_3, BS_6\rangle$, and $\langle BS_5, BS_7, BS_8\rangle$.

The paper presents a method for identification of typical movements among objects moving in indoor spaces. The method targets specifically movements in indoor spaces, and it is the first such method that is able to contend with the whole spectrum of object location data that captures movement in symbolic terms; it is therefore independent on the positioning

978-0-7695-4973-6/13 $26.00 © 2013 IEEE
DOI 10.1109/MDM.2013.29

technology used for collecting the movement data. The method leverages concepts from sequential pattern mining [9], [10] and encompasses two new techniques for the generation of candidate frequent movement patterns. These exploit properties of indoor space in order to improve performance. Empirical studies show that when using one of the two techniques, the method is able to efficiently support datasets comprising several millions of records.

The paper covers a series of experiments conducted in order to test the efficiency of each proposed technique. Both simulated and real-world data are used. The real-world data used is obtained from Copenhagen Airport and contains more than 7 million observations from 25 Bluetooth base stations, covering more than 74,000 different users.

We exploit the real-world dataset for not only considering computational efficiency, but also considering the query results obtained. The findings suggest that our techniques are effective in discovering patterns of objects moving in indoor spaces.

A variety of techniques have been proposed for finding frequent patterns in spatio-temporal data and sequential patterns in non-spatial data [9]–[14]. However, these techniques do not take into account the specifics of the indoor setting as described above. To the best of our knowledge, only two studies [15], [16] consider the problem of finding movement patterns in indoor spaces, and the resulting proposals differ substantially from the techniques we propose. First, the existing methods are developed for RFID systems, while our techniques can be used with all indoor positioning systems where the location is represented in symbolic terms. Second, unlike the existing methods, we target large indoor spaces with large populations.

To summarize, this paper proposes three contributions:

- A general method for identifying typical movements among indoor moving objects that supports pattern identification at multiple spatial granularities.
- Two novel candidate frequent movement pattern generation techniques that exploit the indoor setting to achieve improved performance.
- An empirical study with synthetic data as well as a large real-world dataset that elicits design properties of the method and techniques.

The remainder of this paper is organized as follows: Section II presents the problem setting and states the problem; then Sec. III describes our approach and presents the two new techniques for candidate generation. Section IV discusses an optional post-processing that identifies patterns on a coarse spatial granularity. Section V reports on the experimental study. Finally, Sec. VI covers related work, and Sec. VII concludes and offers directions for future work.

II. Preliminaries

A. Indoor Positioning Data

We assume a setting where indoor positioning sensors (e.g., RFID readers or Bluetooth base stations) are deployed at preselected indoor positions. Moving objects are attached with detectable items like RFID tags or mobile phones with Bluetooth interfaces. When an object is within the sensing range of a positioning sensor, its presence is detected and reported by the sensor. Each sensor continuously detects and reports objects with a frequency determined by its sampling rate. A positioning reading is of the form $(sensorID, objectID, time)$, where $sensorID$ identifies a positioning sensor, $objectID$ identifies an object, and $time$ is the detection time.

A set of readings for the running example is shown in Table I, where $readingID$ identifies a reading.

Table I
Raw Positioning Data

readingID	sensorID	objectID	time
r_1	BS_1	$user_1$	t_1
r_2	BS_1	$user_1$	t_2
r_3	BS_1	$user_3$	t_2
r_4	BS_1	$user_1$	t_3
r_5	BS_1	$user_3$	t_3
r_6	BS_2	$user_3$	t_4
r_7	BS_2	$user_3$	t_5
r_8	BS_2	$user_1$	t_7

B. Object Tracking Table

In a pre-processing step [17], the raw readings are consolidated and inserted into an *Object Tracking Table* (OTT) with schema $(recordID, sensorID, objectID, t_s, t_e)$. A record in the table states that the object $objectID$ is continuously observed by the positioning sensor $sensorID$ in the closed period from time t_s to time t_e. In addition, $recordID$ identifies a record.

The content of Table I can be transformed to that in Table II.

Table II
Indoor Object Tracking Table

recordID	sensorID	objectID	t_s	t_e
rd_1	BS_1	$user_1$	t_1	t_3
rd_2	BS_1	$user_3$	t_2	t_3
rd_3	BS_2	$user_3$	t_4	t_5
rd_4	BS_2	$user_1$	t_7	t_8

C. Problem Statement

As a precursor to defining the problem addressed, we define key concepts that are also used throughout the paper.

Given an object tracking record rd, $\Pi_{sensorID,t_s,t_e}(rd)$ is a *tracking triple* of object $rd.objectID$. An object can have multiple tracking records and has a corresponding tracking triple for each such record. Referring to the running example in Table II, $user_1$'s tracking triple for rd_1 is (BS_1, t_1, t_3).

Given consecutive tracking triples $(sensor_i, t_{si}, t_{ei})$ and $(sensor_j, t_{sj}, t_{ej})$ for an object, the *transition duration*, i.e., the time period between the two tracking records, is $t_{sj} - t_{ei}$. Next, an object's *indoor trajectory* is the time-ordered sequence of the object's tracking triples.

It is possible for an object to have multiple indoor trajectories in an object tracking table OTT. Whether or not two consecutive tracking triples belong to the same or to

two different trajectories depends on the *transition duration*. Specifically, two consecutive tracking triples $(sensor_i, t_{si}, t_{ei})$ and $(sensor_j, t_{sj}, t_{ej})$ for an object are put in the same trajectory if and only if $t_{sj} - t_{ei} \leq T_{split}$, where T_{split} is a *splitting threshold*. Otherwise, $(sensor_i, t_{si}, t_{ei})$ closes a trajectory, and $(sensor_j, t_{sj}, t_{ej})$ starts a new trajectory.

The splitting threshold T_{split} is useful when dealing with cases where moving objects are not observed for a long duration of time. For example, employees usually leave their offices after working hours and return to their offices only the next working day. In such cases, separate trajectories should be formed for different days. We use $\mathcal{TT}(OTT, T_{split})$ to denote the *trajectory table* that results from OTT when using T_{split} as the splitting threshold.

In the running example in Table II, let the splitting threshold T_{split} be 3 time units. As a result, in $\mathcal{TT}(OTT, 3)$ $user_1$ has two trajectories, i.e., $((BS_1, t_1, t_3))$ and $((BS_4, t_7, t_8))$, and $user_3$ has one trajectory, i.e., $((BS_1, t_2, t_3), (BS_2, t_4, t_5))$.

Definition 1: **Indoor Movement Pattern**
An *indoor movement pattern* is a sequence of sensor identifiers, i.e., $\langle sensor_{i_1}, sensor_{i_2}, \ldots, sensor_{i_n} \rangle$, $n \geq 1$.

We refer to an indoor movement pattern of length n as an n-*pattern*.

An indoor trajectory *upholds* an indoor movement pattern $\langle sensor_{i_1}, sensor_{i_2}, \ldots, sensor_{i_n} \rangle$ if and only if there exist n consecutive tracking triples $(sensor_{i_1}, t_{s1}, t_{e1})$, $(sensor_{i_2}, t_{s2}, t_{e2}), \ldots, (sensor_{i_n}, t_{sn}, t_{en})$ in the trajectory. In a single trajectory there can exist more than one of such sequences of n consecutive tracking triples; we let $\mathcal{OCC}(p, t)$ denote the number of occurrences of a pattern p in a trajectory t.

Definition 2: **Support**
Given an object tracking table OTT, a splitting threshold T_{split}, and an indoor pattern p, the *support* of p is

$$\text{support}(p, OTT, T_{split}) = \sum_{t \in \mathcal{TT}(OTT, T_{split})} \mathcal{OCC}(p, t)$$

Thus, the OTT is split according to T_{split} to obtain a set of trajectories. The support of p is then the sum over these trajectories of the number of times each trajectory upholds p.

We are interested in finding all frequent indoor movement patterns from the information captured in an object tracking table.

Problem Definition: Given an indoor object tracking table OTT, a splitting threshold T_{split}, and a support threshold S_{min}, *frequent indoor movement pattern mining* returns all indoor movement patterns p that satisfy $support(p, OTT, T_{split}) \geq S_{min}$.

III. Mining Frequent Indoor Movement Patterns

We proceed to detail our solution to the problem of frequent indoor movement pattern mining.

A. Solution Outline

The outline of our solution is shown in Fig. 2. First, the data is prepared for the mining process. The object tracking table *OTT* is formed from the raw positioning data, and the trajectory table $\mathcal{TT}(OTT, T_{split})$ is built according to the T_{split} set by the user. Second, the actual mining process begins and subsequent iterations of the trajectory table $\mathcal{TT}$ are done in order to discover all the frequent indoor movement patterns. Finally, optional post-processing is performed, to be described in Sec. IV.

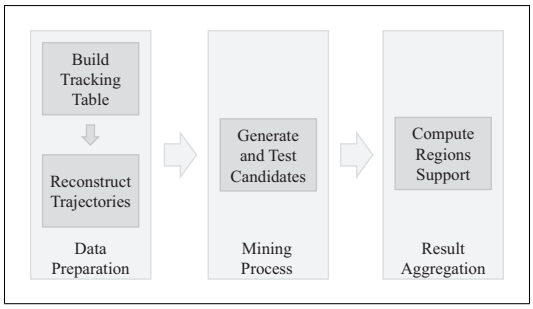


Figure 2. Method Outline

The actual mining process consists of the three steps shown in Fig. 3. In the first step, we obtain the set of individual sensors (1-patterns) and the set of transitions (2-patterns) that satisfy the minimum support requirement. In the second step, multiple passes over the intermediate data are performed in order to discover longer patterns that meet the minimum support. In each pass, the frequent pattern results calculated in previous passes are combined to generate longer candidate patterns. In the third step, the aggregate support of each candidate pattern is computed, and only the qualifying candidates are kept for use in subsequent passes. The process terminates when no candidate patterns exist that meet the minimum support. In the third step, final results (including all frequent patterns) are returned.

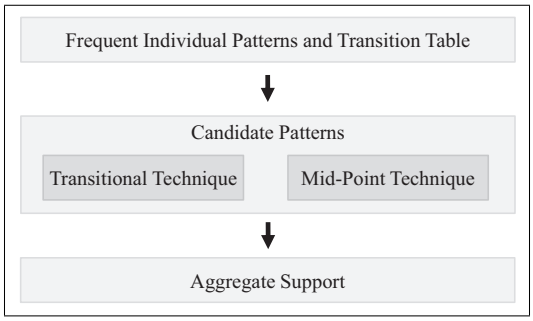


Figure 3. Mining Process

B. Frequent Individual Patterns and Transition Table

The first step calculates the supports for sensors and transitions, i.e., 1-patterns and 2-patterns, respectively. The frequent individual sensors are needed in subsequent steps because every movement pattern that meets the minimum support must be composed of a set of 1-patterns that each meet the minimum support. Furthermore, finding all frequent transitions provides us with additional information on the indoor space that can be exploited during candidate generation.

The set of frequent n-patterns with their support is stored in a table called *n-pattern table* with the schema $(pattern, support)$. In the rest of the paper, we use PT_n to denote an n-pattern table.

The pseudo-code presented in Algorithm 1 shows how to find individual frequent sensors and frequent transitions from a trajectory table. Trajectory table $\mathcal{TT}$ is iterated once (lines 1–13), and each trajectory is scanned element by element. For each element of a trajectory, the support of the corresponding 1-pattern is incremented by 1 (lines 4–7), and also the support of the 2-pattern composed by the element itself and the next element (line 8) is incremented by 1 (lines 9–12). Having

Algorithm 1 compute1and2Patterns($\mathcal{TT}, S_{min}$)

```
1: for t ∈ TT do
2:     while t ≠ ⊥ do
3:         p ← head(t).sensorID;
4:         if PT1.contains(p) then
5:             PT1.incrementSupport(p, 1);
6:         else
7:             PT1.add(p, 1);
8:         p ← p.concat(head(tail(t)).sensorID);
9:         if PT2.contains(p) then
10:            PT2.incrementSupport(p, 1);
11:        else
12:            PT2.add(p, 1);
13:        t ← tail(t);
14: for e ∈ PT1 do
15:    if e.support < Smin then
16:        PT1.remove(e);
17: for e ∈ PT2 do
18:    if e.support < Smin then
19:        PT2.remove(e);
20: return PT1, PT2;
```

iterated through the whole trajectory table, infrequent 1-patterns and 2-patterns are removed from the respective pattern tables (lines 14–19). The results are returned in a 1-pattern table and a 2-pattern table (line 20).

C. Candidate Patterns Generation

The mining process proceeds by performing a series of passes over the data in order to discover longer patterns that meet the minimum support. In each pass, candidate patterns are generated based on the result of the previous pass, and the support of each pattern is computed.

To determine whether a candidate pattern actually meets the minimum support, an iteration of the whole trajectory table is needed. Hence, it is very important to reduce the number of candidates as much as possible while ensuring that all frequent patterns are found.

A naive approach to generate potentially frequent candidate patterns is to consider all combinations of frequent patterns found previously. This approach then considers all combinations of sensors from the 1-pattern table, meaning that it generates many candidates with support 0. This is because the indoor topology dictates no path exists that directly connects many pairs of sensors.

A better idea is to utilize information contained in the previously formed transition table in order to prune irrelevant candidates. In general, we exploit all the information provided by the results of previous passes in order to prune irrelevant candidate patterns.

In the next two sections, we design two topology-based techniques for generating potentially frequent candidate patterns. Both techniques exploit the indoor topology as well as the data gathered in the previous passes to eliminate infrequent candidates quickly. In each technique, we generate longer candidates by appending a $seedPattern$ to a $basePattern$. Patterns p_i and p_j can be combined if $last(p_i) = head(p_j)$, i.e., the last element of the base pattern must be equal to the first element of the seed pattern. We refer to this as the *Combination Rule*.

1) Mid-Point Technique: The first technique for candidate patterns generation exploits as much information as possible from the frequent patterns found in previous passes. The length of a seed pattern is adjusted to be as close to the length of the base pattern as possible. Hence, both base and seed patterns contribute to the new pattern with at most half. In order to achieve this, the length of base patterns is set to $\lfloor \frac{n}{2} + 1 \rfloor$ and the length of seed patterns is set to $\lceil \frac{n}{2} \rceil$. Therefore, base and seed patterns differ at most by 1 in length.

This *Mid-point Technique* is described in Algorithm 2. The algorithm takes as input a value n, the length of the patterns to obtain, and it returns an n-pattern table. Base and seed patterns

Algorithm 2 midPointTechnique(n)

```
1: bases ← PT⌊n/2+1⌋.pattern;
2: seeds ← PT⌈n/2⌉.pattern;
3: candidates ← generateCandidates(bases, seeds);
4: for p ∈ candidates do
5:     s ← calculateSupport(p);
6:     if s ≥ Smin then
7:         PTn.add(p, s);
8: return PTn;
```

are retrieved from the pattern tables corresponding to half the length of n (lines 1–2). A set of candidates is generated using the algorithm described in Algorithm 3 (line 3). After that, the support of each n-pattern candidate is calculated by function calculateSupport($\cdot$) (line 5), which scans the trajectory table and counts pattern occurrences according to Def. 2, and the frequent n-patterns are added to the n-pattern table (lines 6–7) to be returned (line 8).

Algorithm 3 generates candidates using base and seed patterns. It takes as input two sets (base and seed patterns) and outputs a set of candidate patterns. For each base pattern all the seed patterns are checked (lines 2–3), and the seed patterns that satisfy the Combination Rule (line 4) are appended to the base pattern with the common element removed, i.e., the first element of the seed (line 5). These combined patterns are the candidates (line 6) that are then returned.

Table III shows how 4-patterns are combined with 3-patterns to generate candidate 6-patterns for the running example.

Algorithm 3 generateCandidates($baseList$, $seedList$)

```
1: candidates ← ∅;
2: for base ∈ baseList do
3:    for seed ∈ seedList do
4:       if last(base) = head(seed) then
5:          p ← base.concat(tail(seed));
6:          candidates ← candidates ∪ {p};
7: return candidates;
```

Table III
GENERATING CANDIDATE 6-PATTERNS

4-patterns		3-patterns	
$\langle BS_1, BS_2, BS_3, BS_5\rangle$	×	$\langle BS_1, BS_2, BS_3\rangle$	⇒
$\langle BS_1, BS_2, BS_3, BS_6\rangle$		$\langle BS_2, BS_3, BS_5\rangle$	
		$\langle BS_2, BS_3, BS_6\rangle$	
		$\langle BS_5, BS_7, BS_8\rangle$	

Candidate 6-patterns
$\langle BS_1, BS_2, BS_3, BS_5, BS_7, BS_8\rangle$

2) Transitional Technique: This technique exploits the constraints of the indoor movement transitions in order to reduce the number of candidates. Such constraints are captured in the frequent transitions. Indeed, the 2-pattern table contains all the possible connections between two sensors in the positioning data. This technique generates candidate patterns by expanding patterns formed in the most recent pass $(n-1)$ with patterns from the 2-pattern table.

The transitional technique is formalized in Algorithm 4. It differs from Algorithm 2 only in the first two lines, where base patterns are retrieved from the $(n-1)$-pattern table (line 1) and seed patterns from the 2-pattern table (line 2).

Algorithm 4 transitionalTechnique(n)

```
1: bases ← PT_{n-1}.pattern;
2: seeds ← PT_2.pattern;
3: candidates ← generateCandidates(bases, seeds);
4: for p ∈ candidates do
5:    s ← calculateSupport(p);
6:    if s ≥ S_min then
7:       PT_n.add(p, s);
8: return PT_n;
```

Table IV illustrates the generation of candidate 4-patterns by the transitional technique. Here, the 3-patterns and the 2-patterns are used to generate 4-patterns.

Table IV
GENERATING CANDIDATE 4-PATTERNS

3-patterns		2-patterns		Candidate 4-patterns
$\langle BS_1, BS_2, BS_3\rangle$	×	$\langle BS_1, BS_2\rangle$	⇒	$\langle BS_1, BS_2, BS_3, BS_5\rangle$
$\langle BS_2, BS_3, BS_5\rangle$		$\langle BS_2, BS_3\rangle$		$\langle BS_1, BS_2, BS_3, BS_6\rangle$
$\langle BS_2, BS_3, BS_6\rangle$		$\langle BS_3, BS_5\rangle$		$\langle BS_2, BS_3, BS_5, BS_7\rangle$
$\langle BS_5, BS_7, BS_8\rangle$		$\langle BS_3, BS_6\rangle$		
		$\langle BS_5, BS_7\rangle$		
		$\langle BS_7, BS_8\rangle$		

D. Aggregate Support

So far we have used Def. 2 to calculate a pattern p's support as the number of occurrences of p in the trajectory table $\mathcal{TT}$. This definition of support does not take into account the number of objects that contribute to the support of a pattern. Thus, it does not distinguish between the extreme cases where a support of s is due to a single object and where s is due to s objects.

In traditional sequential pattern mining [9], [10], the support of a sequence is calculated with regard to the number of objects supporting it. In our setting, this means that the support of a pattern is incremented only once per object. However, when a moving object 9 times out of 10 tends to move according to pattern p_1 instead of pattern p_2, the support of p_1 should be able to reflect this.

On the other hand, incrementing the support of a pattern every time it is encountered in the data without paying attention to the identity of the objects may also not always be ideal. For example, this can introduce sensitivity to outliers because a single outlier object whose trajectories frequently uphold a pattern can significantly affect, or distort, the final result.

To account for the above observations, we introduce the concept of *aggregate support*.

Definition 3: **Aggregate Support**
Given an aggregation weight $w \in [0, 1]$, the *aggregate support* of a pattern p is defined as:

$$support_A = support_D + (support_T - support_D) \times w$$

Here, $support_D$ represents the number of distinct objects with a trajectory that upholds a pattern; $support_T$ is the support according to Def. 2, the total number of times a pattern is upheld by trajectories in the trajectory table. Thus, $support_T \geq support_D$.

Parameter w controls the impact of a single object on the support of a pattern. When $w = 0$, $support_A$ mirrors the typical sequential pattern mining scenario where support is incremented once per object. When $w = 1$, $support_A$ is the same as Def. 2. Different values of w can disclose different information from the data, according to the needs of an application.

Different settings for w are useful in different scenarios. In a shop planning application, a location is good if many different potential customers pass by the location, while it is not good if only a few customers pass by many times. In this case, we set w to a low value. In a safety study, it is useful to know the use that some part of an indoor space is getting. It does not matter whether the space is being "re-used" by relatively few people or is being used by many different people. In this case, we set w to a large value.

IV. REGION-BASED PATTERN MINING

The techniques presented in Sec. III mine frequent indoor moving patterns based on positioning sensors. This fine granularity of individual sensor ranges may obscure interesting patterns that are clearer at a coarser spatial granularity. It can also be of interest to know how many objects moved from one large indoor region to another large region. For example, when managing a large airport, it is relevant to determine

how passengers move between different regions such as tax-free shopping zone, the airport security area, and the different boarding gates.

Here, we define region-based pattern mining as an optional step that occurs after the mining process described in Sec. III. This optional step only accesses the frequent indoor movement pattern results, without any need for additional scanning of the object trajectory table.

We assume that an indoor space is divided into disjoint regions, each of which covers a set of positioning sensors. Accordingly, we define a mapping $S2R : S \rightarrow R$ that maps a sensor to its region. Given a sensor s_i, $S2R(s_i)$ thus returns the region that covers s_i. In addition, a region r_j can be regarded as the set of sensors $\{sensor_{i_1}, sensor_{i_2}, \ldots, sensor_{i_n}\}$ for which $S2R$ returns r_j.

We then define the region pattern notion as follows.

Definition 4: **Region Pattern**

A *region pattern* rp is a sequence of region identifiers, i.e., $\langle r_{i_1}, r_{i_2}, \ldots, r_{i_n} \rangle$, $n \geq 1$.

Using the mapping $S2R$, we are able to generate the frequent region patterns of length n from the set of all frequent indoor movement patterns of length n. Algorithm 5 describes the process. It takes PT_n as input and returns all frequent region patterns of length n in a table RPT_n with schema $(regionPattern, support)$. The for-loop (lines 1–7) iterates

Algorithm 5 computeRegionPatterns(PT_n)

Require: $n \geq 2$
1: **for** $(p, s) \in PT_n$ **do**
2: $rp \leftarrow$ mapToRegion(p);
3: **if** $rp.length = n$ **then**
4: **if** $RPT_n.contains(rp)$ **then**
5: $RPT_n.incrementSupport(rp, s)$;
6: **else**
7: $RPT_n.add(rp, s)$;
8: **return** RPT_n;

through all the frequent indoor movement n-patterns. Each such pattern p is converted to a region pattern rp using a mapping function (line 2). If rp is already in RPT_n, its support is incremented by p's support s (lines 4–5); otherwise, rp is inserted into RPT_n with s being its initial support (line 7). The algorithm can be used to produce region patterns of any length n.

The function mapToRegion(.) is described in Algorithm 6. It converts an indoor movement pattern p to a region pattern based on mapping $S2R$. Specifically, each sensor s in p is mapped to its region (line 3). If two adjacent sensors map to the same region, only one region identifier is kept in the region pattern (lines 4–6).

An example of region pattern computation is illustrated in Table V. It is based on the running example, where the base stations are grouped into three regions, $r_1 = \{BS_1, BS_2\}$, $r_2 = \{BS_3\}$, and $r_3 = \{BS_4, BS_5, BS_6, BS_7, BS_8\}$. The table shows the frequent region patterns that can be obtained from a 3-region-pattern table.

Algorithm 6 mapToRegion(p)

Require: $p \neq \perp$
1: $r_{prev} \leftarrow$ null; $rp \leftarrow \langle\rangle$;
2: **for** each element $s \in p$ **do**
3: $r \leftarrow S2R(s)$;
4: **if** $r \neq r_{prev}$ **then**
5: $rp.concat(r)$;
6: $r_{prev} \leftarrow r$;
7: **return** rp;

Table V
CALCULATING 3-REGION-PATTERNS SUPPORT

3-pattern	Support
$\langle BS_1, BS_2, BS_3 \rangle$	4
$\langle BS_2, BS_3, BS_5 \rangle$	2
$\langle BS_2, BS_3, BS_6 \rangle$	2
$\langle BS_5, BS_7, BS_8 \rangle$	2

⇒

region pattern	Support
$\langle r_1, r_2 \rangle$	4
$\langle r_1, r_2, r_3 \rangle$	4
$\langle r_3 \rangle$	2

In the example, the first 3-pattern is converted to a 2-region-pattern, the middle two 3-patterns are converted to one 3-region-pattern with support 4, i.e. the sum of the two corresponding supports, and the last 3-pattern is converted to a 1-region-pattern.

In our frequent pattern mining application, the regions to use depend on the specifics of the application, and the task of defining regions is to be accomplished by domain experts and should be done before starting the mining.

V. EXPERIMENTAL STUDY

This section reports on an experimental study conducted to gain insight into the properties of the proposed method and techniques. Experiments are performed on both synthetic and real-world data. All algorithms described in previous sections are implemented in Java 6. All the experiments that study runtime performance are executed on a 2.50 GHz Core i5 machine with 8 GB main memory, running Windows 7 operating system.

A. Results of Indoor Frequent Pattern Mining

1) Using Synthetic Data: We use a simulation application to generate synthetic indoor moving object data. We vary three parameters. The first is the number of moving objects, which is simply the number of different users moving in the indoor space during the simulation. The second is the number of simulation rounds, which influences the time that each moving object spends in the space before leaving. The third is the number of positions, which indicates how many positioning sensors are deployed in the indoor space.

First, we fix the number of rounds to 100, the number of positions to 19, and the minimum support to 100. We then vary the number of moving objects among 100, 250, 500, and 1,000. The results on mining efficiency are reported in Fig. 4.

When the number of objects is 100 or 250, the two candidate pattern generation techniques perform equally well. However, as the number of objects increases to 500 and 1000, the transitional technique performs much more efficiently. With

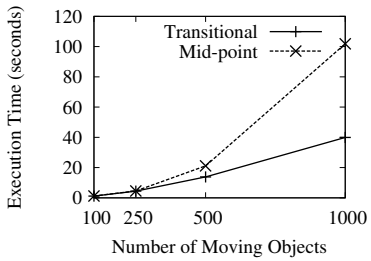


Figure 4. Varying the Number of Moving Objects (100–1,000)

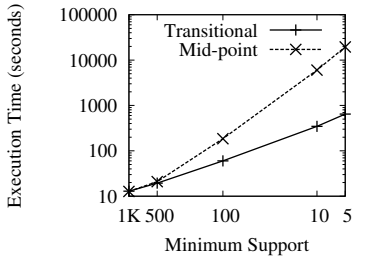


Figure 5. Varying Minimum Support (1,000–5)

1,000 objects, it needs 39.9s to finish, whereas the mid-point technique needs up to 101.8s. The transitional technique makes use of the 2-pattern transition table, which significantly reduces the number of candidate pattern combinations that must be checked during the mining process.

Second, we fix the number of objects to 500, the number of rounds to 250, and the number of positions to 19. We then vary the minimum support S_{min} among 5, 10, 100, 500, and 1,000. The results on mining efficiency are reported in Fig. 5.

The figure shows that the minimum support has high impact on the mining efficiency, as it directly affects the number of candidates that are pruned. A higher minimum support results in a larger number of candidates being pruned at the beginning of the mining, which reduces the subsequent computational loads. The figure also shows that the transitional technique outperforms the mid-point technique. A huge performance gap is observed when the minimum support is small (5 and 10). This again indicates that the transitional technique makes effective use of the indoor topology to constrain candidate pattern growth.

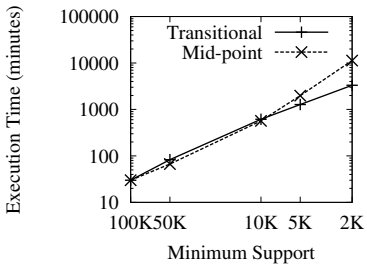
2) Using Real-World Data: We also use a tracking dataset from Copenhagen Airport to evaluate the indoor movement pattern mining techniques. The dataset is obtained from a Bluetooth-based positioning system that encompasses 25 Bluetooth base stations deployed primarily in the airport's Terminals 2 and 3; the user is located within a rough circle of about 30 meters [18]. When a device, typically a mobile phone, with an enabled Bluetooth interface is within range of such a base station, the positioning system continually records an identifier of the device. According to the supplier of the system, some 10–20% of all mobile phones have their Bluetooth interface enabled. The system supports a variety of services, including services that estimate the waiting time in the security area [19].

The total number of raw position records in the dataset slightly exceeds 7 million. This yields a total of 135,397 moving object trajectories when setting T_{split} = 45 minutes. We use Def. 2 to measure the frequency of patterns. The most frequent position in the dataset has a support of 1,064,350.

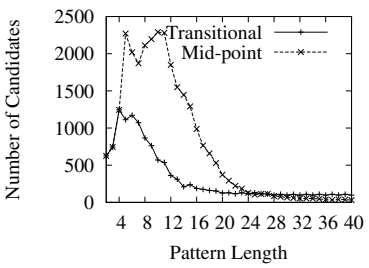
Experiments are conducted in order to discover the most frequent trajectories in the dataset. Five mining runs are performed with minimum support set to 100,000, 50,000, 10,000, 5,000, and 2,000. The results are summarized in Fig. 6(a). We use logarithmic scales on both axes in order to ease the readability of the results for the high minimum support values.

As with the synthetic data, the transitional technique outperforms the mid-point technique for small minimum support, even if the latter has a major advantage when minimum support is set to a high value. In fact, with minimum support 50,000, the mid-point technique finishes in 67 minutes, which is 16 minutes earlier than the transitional technique. The slight disadvantage of the latter technique seen here is attributed to the fact that the real dataset contains a large number of short frequent patterns—the use of the 2-pattern transition table is penalized by this. When the minimum support is set to 5,000, the transitional technique takes 1,279 minutes to finish, whereas the mid-point technique takes 1,977 minutes.

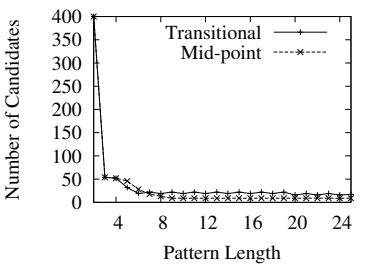
We also investigate the number of candidate patterns that are generated in each pass of the mining process. The results for minimum support 5,000 are shown in Fig. 6(b), and those



(a) Varying Minimum Support (100K–2K)



(b) Number of Candidates, $S_{min} = 5.000$



(c) Number of Candidates, $S_{min} = 50.000$

Figure 6. Number of Generated Candidates

for minimum support 50,000 are shown in Fig. 6(c).

From these two graphs, we find an explanation for the behaviors of the two techniques reported in Fig. 6(a). The mid-point technique performs better than the transitional technique when the minimum support is set to 50,000, but worse when it is set to 5,000. The reason is revealed by the number of candidates generated in each pass. The mid-point technique generates less candidates than the transitional technique when the minimum support is 50,000 as shown in Fig. 6(c). We also see that the number of candidates is extremely higher when minimum support is 5,000, as shown Fig. 6(b). Large numbers of candidates result in more room for the transitional technique to prune candidate patterns using the transition table.

3) Summary: The mid-point technique performs best when the size of the dataset is relatively small. It has a chance to outperform the transitional technique when the dataset is small or when most patterns are short. The transitional technique is significantly more scalable and less sensitive to variation in the minimum support. Overall, the transitional technique is more generally applicable, and the experimental study indicates that it performs better in almost all experimental settings.

B. Results on Aggregate Support

We use synthetic data to evaluate aggregate support (Def. 3) based frequent pattern mining. We mainly investigate the impact of the aggregation weight w. Recall that given a pattern p, w controls the influence of the distinctness of the objects that uphold p. We report the experimental results for the transitional technique. Use of the mid-point technique yields the same frequent patterns, but generally with somewhat lower performance.

We first fix the number of simulation rounds to 100 and vary the number of moving objects from 100 to 1,000. The results are shown in Fig. 7. For both object cardinalities, the

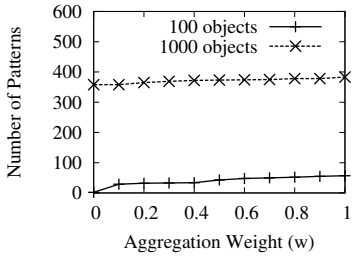


Figure 7. Varying Aggregation Weight and Number of Objects

number of patterns increases very slowly as w increases from 0 to 1. Therefore, the effect of w is insensitive to the number of objects.

Next, we fix the number of objects to 500 and vary the number of rounds from 100 to 1,000. The results are shown in Fig. 8. When the number of rounds is set to 100, w actually has no effect. When the number of rounds is increased to 1,000, w influences the number of patterns, which increases linearly as w increases. More simulation rounds make objects spend more time and generate more data in the simulation, which increases

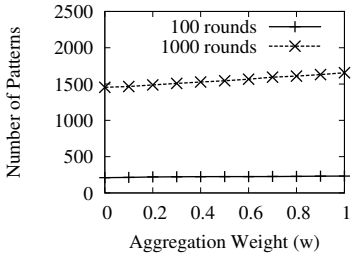


Figure 8. Varying Aggregation Weight and Number of Rounds

the probability that the same object supports the same pattern multiple times. Hence, the difference between $support_T$ and $support_D$ is higher for a higher number of rounds, and the effect of w is thus more significant. The effect of w on the

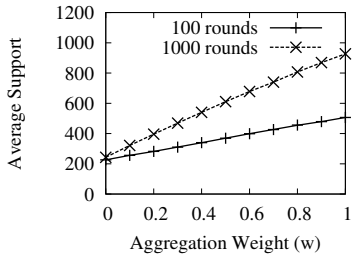


Figure 9. Varying Aggregation Weight and Number of Rounds

average support of frequent patterns is shown in Fig. 9. As expected, also the average support is increasing with w; and also in this case, the rate of increase is higher for a higher number of rounds.

We report on the effect of w on the mining efficiency in Fig. 10. For a small number of rounds, the execution time is constant. However, when the number of rounds is high, the execution time is clearly affected by the value of w. As we

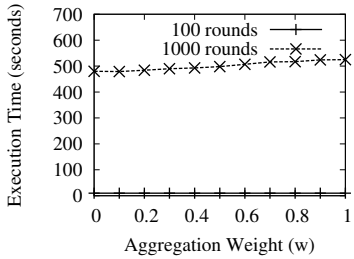


Figure 10. Varying Aggregation Weight and Number of Rounds

have already noticed, the number of patterns increases with w for high numbers of rounds, and a higher number of patterns implies a higher number of candidates to check, which results in a higher execution time.

C. Results on Region Pattern Mining

In order to gain insight into the workings of region-based mining, we perform experiments on the Copenhagen Airport

dataset. We aim to determine whether the use of different regions (coarse grained versus fine grained) yields different insights into the data.

To obtain meaningful region patterns, we clean the data by removing noise. The total number of positioning readings are reduced from 7 million to approximately 4 million. We use two different ways of grouping locations into regions.

First, we identify regions in the airport with different functions. Fig. 11 displays the 9 resulting regions. We then assign the locations to the function regions that they are located in. This grouping can be used in order to analyze the frequent paths that people follow inside the airport before getting to a gate. Specifically, grouping all gates into the same region is helpful in identifying these kinds of patterns.

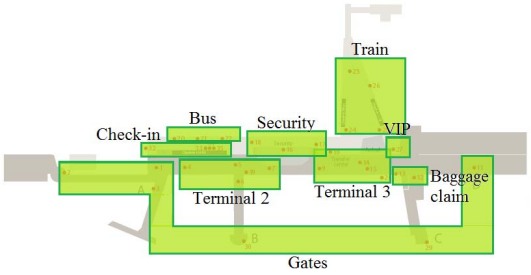


Figure 11. Regions in Function-based Grouping

The top-5 frequent 2-region patterns are listed in Table VI. Clearly, the first two patterns indicate that the Scandinavian Airlines (SAS) lounge is heavily used in Copenhagen Airport. This is reasonable because Copenhagen Airport is the airline's main hub. The frequent patterns involving the bus station conform to the fact that buses are convenient and extremely often used means of transportation for passengers from and to Copenhagen Airport and between the domestic terminal (Terminal 1) and the international terminals (Terminals 2, 3).

Table VI
REGION MINING RESULTS, FUNCTION-BASED GROUPING

Frequent 2-region-patterns	Support
$\langle Terminal2, VIP \rangle$	448545
$\langle VIP, Terminal2 \rangle$	234655
$\langle Terminal2, Bus \rangle$	210243
$\langle Security, Bus \rangle$	128445
$\langle Bus, Security \rangle$	127341

Second, we focus on the boarding gates, which we group into four disjoint regions. We divide all other areas into three regions: terminal, security, and outside. The regions are shown in Fig. 12. Performing region pattern mining using this region setting can discover patterns between different gates.

Table VII reports the top-5 frequent 2-region patterns that involve gates. We observe frequent movements from Terminal to Gate D, which has boarding gates for international flights. We also discover frequent movements to and from Gate A. This is because most domestic and many Schengen flights are accessed through gates there. Also, pattern

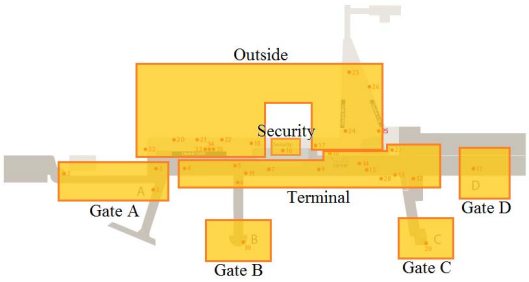


Figure 12. Regions in Gates-Based Grouping

Table VII
REGION MINING RESULTS, GATES-BASED GROUPING

Frequent 2-region-patterns	Support
$\langle Terminal, GateD \rangle$	153154
$\langle GateA, Outside \rangle$	126339
$\langle Outside, GateA \rangle$	97481
$\langle GateA, Terminal \rangle$	92795
$\langle Outside, GateD \rangle$	81177

$\langle GateA, Terminal \rangle$ suggests that many passengers transfer from domestic and Schengen flights to longer distance international flights that are accessed via the Terminal region.

D. Summary

Experiments on runtime performance show that the method used with the transitional technique is capable of dealing with a large dataset. Experiments on aggregate support show that it depends on the time objects spend in the space and not on the number of objects. Results of region-based pattern mining applied on a real-world dataset provide useful information for the analysis of movements of objects in the indoor space.

VI. RELATED WORK

We are aware of only a few works that consider the identification of typical object movements in indoor settings. However, there are a number of works that target other problems that are relevant to our work. We proceed to consider related work on sequential pattern mining, indoor tracking, and indoor movement mining.

Agrawal and Srikant [9] introduce the problem of sequential pattern mining: given a database of customer transactions (i.e., ⟨customer ID, time, set of items purchased⟩), sequential pattern mining solves the problem of discovering sequences of items that are most frequently purchased by customers. We use the structure of sequential pattern mining algorithms [9], [10] for our methods. However, indoor movement patterns differ from purchase patterns in three important respects: (i) an element of the former has a duration, while the latter is instantaneous; (ii) an element of a movement pattern is a single sensor, while in a purchase pattern, it can be a set of items; and (iii) while the sequence of elements is strict in movement patterns, in purchase patterns, if two elements occur one right after the other in one sequence and are interleaved with a third element in another sequence, the two sequences are still considered

as upholding the same pattern. Taking into account these differences and the specifics of indoor movements, we propose two new candidate generation techniques that are suitable for the indoor movement setting.

Next, the pattern mining method presented in this paper leverages a graph model based indoor tracking approach that utilizes the topology of indoor space [17]. In particular, the paper exploits a technique that transforms an initial dataset into a more compact representation.

Since we model the indoor space as a graph, it is natural to think of using graph mining methods [20]. However, it is not straightforward to represent our data as a graph. Doing so appears to call for a graph model that is not explicitly considered in standard graph mining. This led us to not use this approach, although it may be an interesting future research direction.

Finally, we are aware of two studies that address the problem of indoor movement mining [15], [16]. The first study [15] targets activity monitoring where it is important to identify frequent movements. An RFID infrastructure is proposed that can detect simultaneous movements of a limited number of people. However, this infrastructure can only be deployed in a small field. In contrast, we consider a setting where data is collected by existing positioning systems that cover large indoor spaces with very large populations of moving objects.

The second study [16] considers frequent pattern mining and prediction. The proposed mining method constrains a pattern on the time dimension; hence, if two users use the same path at different times of the day, these are considered as different patterns. This limitation is not reasonable in our context; therefore, we cannot directly compare to this method.

VII. Conclusions and Future Work

The paper considers the realistic setting where the movements of people in indoor spaces are captured using presence sensors. Using the resulting data, the paper tackles the problem of identifying typical movements of objects. Two data mining approaches are presented that take the logic of sequential pattern mining as their outset and utilize the topology of indoor space to improve performance by pruning irrelevant candidate pattern segments. The resulting approaches also support the concepts of aggregate and region-wise pattern mining.

The approaches are subjected to empirical studies with both synthetic and real-world data in order to gain insight into their efficiency and other properties. For example, the studies show that one of the approaches has good scalability features and is able to efficiently contend with datasets comprising several millions of records.

Three research directions are of particular interest. The first relates to the invention of new ways of generating potentially frequent candidate patterns that minimize the number of candidates while assuring that every possible pattern is covered. As indicated by the paper's studies, different candidate generation approaches might perform differently in different phases of the algorithm; therefore, it is of interest to investigate the potentials of hybrid candidate generation models. Second, it would be interesting to extend the representation of patterns to include temporal information (e.g., duration of the stay in a location) and adapt the mining process to deal with this extra information. Third, as graph mining methods show good performance, their use for indoor tracking data deserves exploration.

Acknowledgments

This research was supported in part by the Geocrowd Initial Training Network, funded by the European Commission as an FP7 Peoples Marie Curie Action under grant agreement number 264994, and by the Indoor Spatial Awareness project, funded by the Korean Land Spatialization Group and BK21 program.

References

[1] Y. Su, S. Chelluboina, M. Hahsler, and M. Dunham, "A new data mining model for hurricane intensity prediction," in *Proc. ICDM Workshops*, 2010, pp. 98–105.

[2] T. Hauser and W. Scherer, "Data mining tools for real-time traffic signal decision support & maintenance," in *Proc. Systems, Man, and Cybernetics*, vol. 3, 2001, pp. 1471–1477.

[3] Z. Li, B. Ding, J. Han, and R. Kays, "Swarm: mining relaxed temporal moving object clusters," *PVLDB*, vol. 3, no. 1–2, 2010, pp. 723–734.

[4] K. A. Taipale, "Data mining and domestic security: connecting the dots to make sense of data," *The Columbia Science and Technology Law Review*, vol. 5, no. 2, 2003.

[5] A. Baniukevic, D. Sabonis, C. Jensen, and H. Lu, "Improving Wi-Fi based indoor positioning using Bluetooth add-ons," in *Proc. MDM*, 2011, pp. 246–255.

[6] E. Metola, S. Aparicio, P. Tarrio, and J. R. Casar, "Comparison of localization methods using calibrated and simulated fingerprints for indoor systems based on Bluetooth and WLAN technologies," in *Proc. MADRINET*, 2009, 10 pages.

[7] C. di Flora and M. Hermersdorf, "A practical implementation of indoor location-based services using simple WiFi positioning," *J. Location Based Services*, vol. 2, no. 2, 2008, pp. 87–111.

[8] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 37, part C, no. 7, 2007, pp. 1067–1080.

[9] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. ICDE*, 1995, pp. 3–14.

[10] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. EDBT*, 1996, pp. 3–17.

[11] H. Cao, N. Mamoulis, and D. W. Cheung, "Mining frequent spatio-temporal sequential patterns," in *Proc. ICDM*, 2005, pp. 82–89.

[12] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proc. KDD*, 2007, pp. 330–339.

[13] S. Khetarpaul, R. Chauhan, S. K. Gupta, L. V. Subramaniam, and U. B. Nambiar, "Mining GPS data to determine interesting locations," in *Proc. IIWeb*, 2011, article 8, 6 pages.

[14] N. S. Savage, S. Nishimura, N. E. Chavez, and X. Yan, "Frequent trajectory mining on GPS data," in *Proc. LocWeb*, 2010, article 3, 4 pages.

[15] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao, "Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays," in *Proc. PERCOM*, 2007, pp. 37–46.

[16] K. F. Hu, L. Zhao, Y. C. Xu, and L. Chen, "Research on mining frequent path and prediction algorithms of object movement patterns in RFID database," *Applied Mechanics and Materials*, vol. 109, 2011, pp. 715–719.

[17] C. S. Jensen, H. Lu, and B. Yang, "Graph model based indoor tracking," in *Proc. MDM*, 2009, pp. 122-131.

[18] J. P. Hansen, A. Alapetite, H. B. Andersen, L. Malmborg, and J. Thommesen, "Location-Based Services and Privacy in Airports," in *Proc. INTERACT*, 2009, pp. 168–181.

[19] BlipTrack website, http://www.bliptrack.com/, 2012.

[20] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," in *Proc. ICDM*, 2002, pp. 721–724.