OLAP for moving object data

Oliver Baltzer

Flagstone RE, Suite 700 Cogswell Tower,
2000 Barrington Street,
Halifax, NS, B3J 3K1, Canada
E-mail: obaltzer@flagstonere.com

Frank Dehne

School of Computer Science,
Carleton University,
1125 Colonel By Drive,
Ottawa, ON, K1S 5B6, Canada
E-mail: dehne@scs.carleton.ca

Andrew Rau-Chaplin*

Dalhousie University,
Faculty of Computer Science,
1459 LeMarchant St.,
P.O. Box 15000, Halifax,
NS, B3H 4R2, Canada
E-mail: arc@cs.dal.ca
*Corresponding author

Abstract: In this paper, we present an OLAP framework for moving object data. We introduce a new operator GROUP_TRAJECTORIES for group-by operations on moving object data and present two implementation alternatives for computing groups of moving objects for group-by aggregation: *group by overlap* and *group by intersection*. We also present an interactive OLAP environment for resolution drill-down/roll-up on sets of trajectories and parameter browsing. We evaluate the performance of our GROUP_TRAJECTORIES operator by using generated as well as real life moving object datasets.

Keywords: online analytical processing; OLAP; group-by operator; moving object data; spatial OLAP.

Reference to this paper should be made as follows: Baltzer, O., Dehne, F. and Rau-Chaplin, A. (2013) 'OLAP for moving object data', *Int. J. Intelligent Information and Database Systems*, Vol. 7, No. 1, pp.79–112.

Biographical notes: Oliver Baltzer received his PhD from Dalhousie University in 2011, and in 2002, Dipl-Ing and MSc from HTW Berlin and Reading University, respectively. Since 2007, he has been the lead Architect for high-performance computing at Flagstone Reinsurance and continues to collaborate on various research projects. His research interests include high-performance computing, location intelligence and spatial OLAP, as well as risk analytics.

Frank Dehne received his MCS (Dipl. Inform.) from the RWTH Aachen University, Germany in 1983 and PhD (Dr. Rer. Nat.) from the University of Wurzburg, Germany in 1986. In 1986, he joined the School of Computer Science at Carleton University in Ottawa and in 2009 he was appointed Chancellor's Professor of Computer Science. His research interests include parallel computing, coarse grained parallel algorithms, parallel computational geometry, parallel data warehousing and OLAP, and parallel bioinformatics.

Andrew Rau-Chaplin received his MCS and PhD from Carleton University in Ottawa in 1990 and 1993, respectively. In 1994, he joined the faculty at the Technical University of Nova Scotia and in 1997 became a Professor in the Faculty of Computer Science at Dalhousie University. His research interests are in the application of high performance computing to data and computationally intensive domains including data warehousing, OLAP, spatial information systems, catastrophe modelling, and risk analytics.

1 Introduction

The analysis of moving object databases is a field of research that has received significant attention in recent years (Anwar Hossain and Bazlur Rashid, 2012; Vaisman, 2012; Sistla et al., 1997; Wolfson et al., 1998; Güting and Schneider, 2005; Benetis et al., 2006; Gidófalvi and Pedersen, 2006, 2009; Kuijpers and Vaisman, 2007; Leonardi et al., 2010; Gómez et al., 2008). Typical applications of this discipline are location-based services (Saltenis and Jensen, 2002; Yim et al., 2011), traffic control (Papadias et al., 2001), transport logistics (Ding and Guting, 2004), wild life tracking (Laube and Imfeld, 2002; Li et al., 2011), and epidemiology (Sinha and Mark, 2005). With the large adoption of global positioning systems (GPS), radio frequency identification (RFID) and mobile devices in everyday life, an increasing amount of moving object data is being collected by such applications. Moving object datasets are becoming so large that there is a growing need for the analysis of aggregated information about moving objects.

In traditional data warehouses, a key instrument for the analysis of aggregated information is *online analytical processing* (OLAP) (Chaudhuri and Dayal, 1997). OLAP enables the efficient analysis of multidimensional data by allowing the user to interactively explore the multidimensional space. The selection of views of the multidimensional space is realised by projecting multidimensional points (facts) into a lower-dimensional space defined by only a relevant subset of dimensions. This approach projects a number of points onto the same point in lower-dimensional space for which the measures of the projected points are being aggregated using a predefined aggregation function. Additionally, there may be hierarchies defined for each dimension. They provide a hierarchical grouping of dimension values into categories, resulting in a decrease of cardinality for the dimension at each hierarchy level. During a *roll-up* operation, the categorical grouping of dimension values at each level of the hierarchy is used to aggregate the measures of facts that are mapped into the same group, similar to the aggregation of measures of multidimensional facts when they are projected into lower-dimensional space. Other important operations provided by OLAP are *slice* and *dice*, which can select sub-spaces of the multidimensional space and are comparable to range queries.

To enable a similar interactive analysis of moving object data in an OLAP manner, mechanisms need to be provided that support basic OLAP operations such as conceptual aggregation of facts with respect to selected dimensions and at varying levels of granularity. When analysing moving object data, the objects' trajectories become the facts of the analysis. Consider for example a ride sharing (car pooling) type application where our fact table consists of records representing people and their daily drive from their homes to their offices. Assume that each person's daily drive is represented as a sequence of points representing locations along the drive in time order. We refer to such location sequences as *trajectories*. Our goal is to use a GROUP BY operator to create groups of persons that can share a vehicle. The problem here is that traditional OLAP would group and aggregate records with *equal* values in a given set of feature dimensions. For a set of trajectories, such as in the above ride sharing example, it is very unlikely that any two trajectories are exactly the same (i.e., equal). Using the traditional OLAP approach would result in a GROUP BY operator that does nothing; i.e., creates no groups whatsoever. For OLAP on moving object data, we need new types of GROUP BY operators. That is the main problem addressed in this paper. As can be seen from the above ride share example, GROUP BY operators need to be based on trajectory *similarity* rather than equality. Furthermore, when grouping moving object data, different types of grouping logic may be required depending on the application. Consider for example a fact table representing the movements of a set of people and an application in which we wish to determine the spread of a virus. We desire a GROUP BY operator which returns groups of people that have infected each other. For two persons A and B to infect each other, their paths do not even need to be similar (as in the ride share example). All that is required is that their paths overlap for a certain minimum period of time. This will give rise to a very different GROUP BY operator that will be discussed later in this paper.

When grouping trajectories, issues of resolution that do not occur in the context of traditional OLAP need to be considered. This is largely because the grouping criteria needs to capture the idea of similarity rather than exact match. A GROUP BY operator for trajectories needs to consider both *time resolution* and *space resolution* in determining groupings. For example, in the ride sharing application the space resolution parameter controls how similar trajectories need to be in order to be grouped, that is how far people are willing to walk to their ride share. The time resolution parameter would control the time people are willing to wait for their ride share and how precise the arrival time at their destination needs to be. Needless to say, selection of these parameters have a considerable impact on how the GROUP BY operation should be performed.

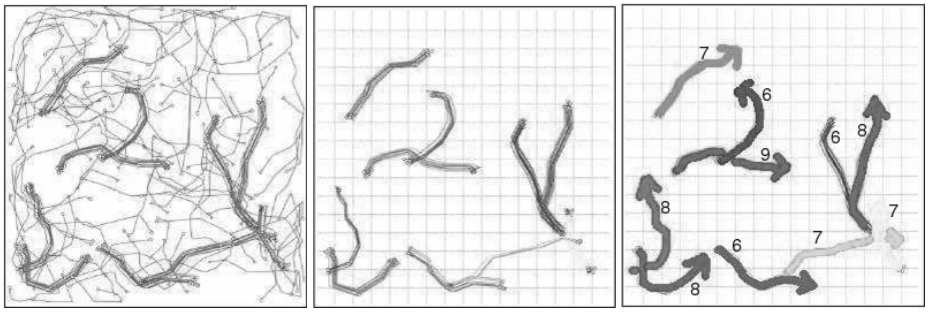
In the remainder of this section, we will first introduce a formal framework for OLAP on moving object data and then outline the main results of this paper, namely how to implement GROUP BY operations on trajectories. Consider a set of moving objects stored as a relational table *objects* where each record has an attribute *trajectory* = $[(x_1, y_1, t_1), (x_2, y_2, t_2),\ldots(x_m, y_m, t_m)]$ representing the movement of the respective object as a sequence of positions at times $t_1, t_2,\ldots t_m$. Our goal is to evaluate OLAP-typical GROUP BY queries with respect to the *trajectory* dimension. The problem is illustrated using the example shown in Figure 1. In Figure 1(a), we observe a number of individual objects that move on random paths plus ten groups of objects that move together on similar paths. Each group consists of more than five objects moving on similar paths which, taken together, appear to the human eye as 'bold' paths. Consider the following SQL query where trajectory is both, a *feature dimension* that is the subject of the GROUP

BY operator as well as a *measure dimension* that is subject to the aggregation function AGGREGATE:

```
SELECT AGGREGATE(trajectory) AS trajectory
    COUNT(trajectory) AS count
FROM objects
GROUP BY GROUP_TRAJECTORIES(trajectory, resolution)
HAVING COUNT(*) >= 5
```

In this example, the aim of the GROUP BY operation with respect to the *feature dimension* trajectory is to group similar trajectories and eliminate groups with less than a given minimum support (less than five similar trajectories). The resulting set of groups is shown in Figure 1(b). Once the groups of trajectories have been determined, aggregate trajectories summarising the trajectories in each group are reported. In this example, an aggregate trajectory is the average trajectory computed by calculating for each time t_i the average of the locations (x_i, y_i) of the trajectories contained in the corresponding group. The result is shown in Figure 1(c), where each qualifying group is represented by its aggregate trajectory and support (count).

Figure 1 Example of OLAP for moving object data, (a) Input data (b) Groups with support above the required minimum support (c) Aggregate results reported (aggregate trajectories and counts)



(a) (b) (c)

In this paper, we propose a new class of GROUP BY operators specifically targeted to OLAP analysis of trajectories and to answering aggregate queries with respect to the spatiotemporal movement of a set of objects. A preliminary description of our results was published in Baltzer et al. (2008) and more details can also be found in a PhD thesis (Baltzer, 2011). The main problem studied here is how to identify aggregation groups with respect to a feature dimension representing trajectories. As discussed earlier, it is very unlikely that any two trajectories are exactly the same. Hence, standard aggregation of records based on groups with equivalent trajectory values is not very useful in most cases. Instead, we propose to partition the given trajectories into groups of trajectories using a new GROUP BY operator, which we term GROUP_TRAJECTORIES. This operator returns a group identifier for each trajectory, and then OLAP can proceed with standard aggregation according to the group identifiers, instead of the trajectories themselves. The simplest case for forming these groups is to identify disjoint groups.

However, when grouping moving objects, identification of disjoint groups may not always be possible, or even desired given that group association may change over time. The goal of the new GROUP_TRAJECTORIES operator is to identify groups of objects that have sufficiently similar behaviour and to deal with the variances in moving object data which often renders traditional GROUP BY operators unsuitable.

Before we proceed with outlining our results, we briefly mention some related approaches for identifying groups of moving objects which we will discuss in detail in Section 2. Various solutions have been developed for specific domains but they do not easily adapt to an application in a general purpose OLAP framework. An exception is frequent pattern mining which has been widely adopted by the data warehousing and data mining community. Many frequent pattern mining approaches have shown good results when identifying patterns that are shared between individual objects. However, they often do not produce results suitable for the moving object data analysis studied in this paper. The most significant reason for this is the amount of redundant information generated by many of these algorithms. For example, even with only a small amount of noise present in the moving object dataset, it is documented, e.g., in Gidófalvi and Pedersen (2006), that a lot of frequent patterns are detected that do not provide any valuable information. Additionally, frequent pattern mining algorithms consider each pattern as an independent piece of information and do not take into account any relationships that may exist between the detected patterns. In many scenarios, interesting patterns may be related to each other by the trajectories they share. These relationships are of importance in applications such as disease tracking, as they may characterise groups of people who potentially have communicated a disease virus through transitive relationships.

The solution proposed in this paper extends frequent pattern mining methods, and presents new algorithms that are more appropriate for identifying groups of moving objects. Our methods are specifically designed to capture relationships between movement patterns and thus supports identification of groups of objects that exhibit a complex behaviour, as in the example of a spreading virus. At the same time, our approach integrates well with existing OLAP models by using established data representations and query languages, as well as allowing the user to interactively browse the results at varying levels of resolution and aggregated information. The main challenge addressed in this paper is how to define and compute the GROUP_TRAJECTORIES operator in such a way that that it defines groups in a manner that supports a meaningful analysis of object movements via OLAP. We propose two versions of the GROUP_TRAJECTORIES operator to compute groups of trajectories that are appropriate for OLAP analysis in different circumstances and applications:

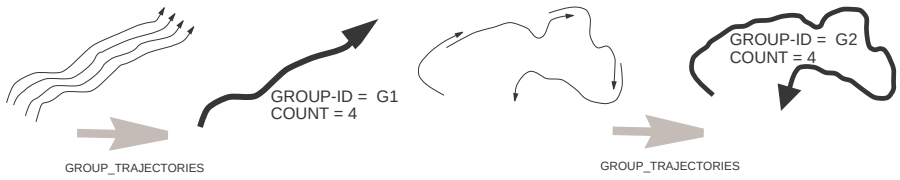
1 *group by overlap*

2 *group by intersection*.

Section 3 shows in detail how the two versions of the GROUP_TRAJECTORIES operator are defined and computed. The following outlines the intuition and motivation behind these operators.

The *group by intersection* method identifies subsets of trajectories that correspond to movements along a similar path. Figure 1 shows an example where movements that follow similar trajectories are aggregated. *Group by intersection* also identifies groups with parallel movements such as ‘marching band’ style parallel trajectories. A schematic illustration is shown in Figure 2(a).

Figure 2 Illustration of two different version of operator GROUP_TRAJECTORIES, (a) group by intersection (b) group by overlap



(a) (b)

The trajectories shown could, for example, represent a group of four people walking together, and the aggregate would be a simplified representation of that movement.

The *group by overlap* method aggregates subsets of trajectories that correspond to sequences of movements with sufficient overlap between subsequent trajectories. A schematic illustration is shown in Figure 2(b). The trajectories shown could, for example, represent movements of people who pass on a disease virus, and the aggregate would then represent the total movement of the virus.

Our *group by intersection* and *group by overlap* methods for the GROUP_TRAJECTORIES operator are presented in Section 3. Both methods depend on various parameters including spatial and time resolution. That allows for analysing trajectories at various levels of detail/resolution and provides another opportunity for an OLAP-style analysis by enabling drill-down and roll-up on the resolution dimension. In Section 4, we outline an interactive OLAP environment for the analysis of trajectories that allows resolution drill-down and roll-up as well as parameter browsing. An experimental evaluation of the proposed algorithms is presented in Section 5. The main goal of the experiments is to determine how well the two versions of the GROUP_TRAJECTORIES operator allow meaningful analysis of object movements via OLAP. We have used various generated and real-life moving object datasets and tested whether the GROUP_TRAJECTORIES operator is appropriate for the OLAP analysis of trajectories in the context of different application scenarios.

2 Related work

A number of data structures and access methods have been proposed to efficiently store, retrieve and update information about moving objects (Kollios et al., 1999; Ŝaltenis et al., 2000; Pfoser et al., 2000; Porkaew et al., 2001; Procopiuc et al., 2002; Saltenis and Jensen, 2002; Hadjieleftheriou et al., 2002; Agarwal et al., 2003; Tao et al., 2003; Chen and Meng, 2009; Zhang et al., 2010). An overview and classification of most of these data structures can be found in Mokbel et al. (2003). In many applications, however, the amount of collected data makes it infeasible to analyse the raw information of every individual object. Instead, data is being processed and analysed in an aggregated manner to extract trends, rules and typical behaviour, as well as exceptions from this typical behaviour. There are a multitude of criteria and information embedded in moving object data that can be analysed in an aggregated manner. However, much of the previous work focused on the aggregation of only numerical facts that are without correlation to the spatiotemporal properties of the object's trajectories. Typical choices for aggregate

dimensions are often time-only dimensions to provide aggregation 'by day' or 'by year', or space-only dimensions for aggregation based on topological relationships, such as 'by location square' or 'within 10 km of' (see for example, Marchand et al., 2004). A comprehensive survey of these methods can be found in López et al. (2005). A different kind of aggregated analysis of moving object trajectories that focuses on the combined spatiotemporal properties of the trajectories has its roots in the area of visual reasoning and artificial intelligence (Yip and Zhao, 1996). In these disciplines, it often is not sufficient to aggregate the numerical properties of trajectories with respect to auxiliary dimensions. Instead, the spatiotemporal properties of the trajectories are aggregated with respect to groups of trajectories that exhibit a similar behaviour.

Next we discuss the previously published work in more detail, using the following classification of techniques:

- clustering
- computational geometry
- edit distance and variants thereof
- frequent pattern or association rule mining.

Note that, no single currently know mining technique is superior to all others in all circumstances. Each of them has its strengths and weaknesses for different patterns of trajectories and applications. A framework that combines several techniques in order to cover a wider range of applications was recently presented in Li et al. (2010, 2011).

2.1 Clustering

An intuitive approach to identifying groups of moving objects is clustering. However, traditional static geometry-based clustering techniques such as *k*-means (Hartigan, 1975), or more recent variants (Har-Peled, 2004), are not always sufficient for moving objects, as objects may leave and enter clusters over time and thus impact the set of clusters that is formed. Li et al. (2004) addressed this problem by applying a micro-clustering strategy, which allows for efficient updating of clusters as time passes and the original set of clusters deteriorates. The underlying concept of this strategy is that objects move continuously and micro-clusters are formed among a small number of close objects with a similar short-term movement (their bounding box does not exceed a given threshold). As the motion within each micro-cluster is roughly the same, each micro-cluster can be considered as an individually moving unit, and updates over time only need to consider interactions among micro-clusters and locally within each cluster. This separation of global and local interactions leads to a significantly reduced amount of computation required to maintain a reasonable clustering over time when compared to recomputing clusters for the entire dataset at each time step. To report groups of similarly moving objects, however, the sequence of clustering that was found through this approach has to be post-processed, an issue not addressed in Li et al. (2004).

In Kriegel and Pfeifle (2005), Kriegel and Pfeifle revisited static clustering for moving object data and proposed an approach using what they call medoid clustering. This approach takes into account the uncertainty that is associated with the position of an object for a particular observation. By sampling object locations from a spatial density function that models the location uncertainty, it is possible to compute a set of static

clusterings. These static clusterings are then compared and ranked according to their distance to each other. The clustering with the smallest rank, i.e., the smallest average distance to other clusterings, is then considered the medoid clustering – the average and most stable clustering of the original dataset.

To address the shortcomings of traditional clustering methods such as *k*-means with respect to clustering of moving object trajectories, Nanni and Pedreschi (2006) proposed the use of density-based clustering (Ester et al., 1996). Using a naïve distance function for trajectories, they showed that density-based clustering is more suitable for clustering trajectories, as the produced clusters can have an arbitrary non-spherical shape and are more robust with respect to noise. However, Nanni and Pedreschi noted that the naïve distance function considers trajectories as atomic entities, which is not suitable in many applications. To address this issue, they proposed a technique called temporal focusing, which examines the neighbourhood of each trajectory to identify time intervals that potentially produce a more meaningful and interesting clustering. Their results show that temporal focusing greatly improves the clustering quality of the naïve distance function.

Clustering is a technique that would be intuitive to use for the identification of groups of moving objects. However, the spatiotemporal nature of trajectories makes it difficult to apply common clustering approaches. While the solutions discussed above are suitable for a number of special purpose applications, they do not fit into a general-purpose data warehousing and OLAP context. Our research focuses on methods that can be applied universally and build on top and integrate with existing database, data warehousing, and data mining technologies.

2.2 Computational geometry

One of the first approaches to use computational geometry to identify patterns in large moving object datasets was proposed by Laube et al. (2004) as an extension to an analysis approach called REMO (Laube and Imfeld, 2002). REMO is based on a coarse-grained analysis of motion parameters that have been stripped of their absolute positions in Euclidean space and instead use parameters such as orientation, speed, and acceleration. These parameters, once mapped to discrete values, and obtained at constant time intervals for each object can be represented as a two-dimensional bitmap. The two dimensions of the bitmap represent the set of objects and time respectively and the value for each element of the bitmap is a numerical value, representing a particular motion parameter. Interesting patterns in the dataset are then identified by determining continuous regions of identical value within the bitmap. The patterns that can be detected using this approach are *constance* (an object maintains constant motion parameter values over a consecutive set of time intervals), *concurrence* (a number of objects have the same motion parameters during the same time interval), and *trend-setter* (a *constance* pattern followed by a concurrence pattern). Laube et al. extended this method by including absolute position information for each recorded observation and employed computational geometry algorithms to identify movement patterns in addition to those above. The general approach is to compute the spatial region, e.g., a circular region, for which a set of constraints is satisfied. These constraints can be, for example, the size of the region or the number of observations for different objects in the region. The additional patterns Laube et al. proposed to detect using the computational geometry approach were *track* (a *constance* pattern additionally constrained by a maximum Euclidean distance between two consecutive observations of the same object), *flock* (a *concurrence*

pattern constrained by a maximum distance of the objects to each other), *leadership* (a *trend-setter* pattern constrained by a maximum distance between the objects when concurrent movement commences), *convergence* (a minimum number of objects pass through a region of fixed size independent of time), and *encounter* (a *convergence* pattern with the constraint that the objects are within the identified region at the same time). Gudmundsson et al. improved the complexity of the exact *encounter* algorithm from Laube et al. (2004) and provided for all grouping patterns discussed in Laube et al. (2004) approximation algorithms to estimate the size of a region, and the minimum number of objects for which interesting patterns can be detected in a given dataset (Gudmundsson et al., 2004). More recently, Benkert et al. (2006) provided computational geometry approximation algorithms with improved asymptotic bounds for the *flock* pattern, and Andersson et al. (2007) discussed an algorithm to detect *leadership* without an a priori knowledge of the time interval of interest.

The described computational geometry approaches to identifying groups of moving objects appear very powerful and able to identify well-defined patterns. However, these approaches are not designed to integrate with technologies such as data warehouses or OLAP, and rather represent stand-alone solutions without the support of established data processing frameworks.

2.3 Edit distance

The edit distance approach, and variations of it, has also been the subject of studies regarding similarity measures for trajectories that can be represented as sequences of motion parameters (e.g., location, orientation, speed, etc.). The most common method that is related to the edit distance and used to measure the similarity between two or more trajectories is finding the longest common subsequence (LCSS). LCSS has originally been proposed for finding similarity in time-series databases (Yazdani and Ozsoyoglu, 1996), which is a problem area closely related to moving object databases. Sclaroff et al. (2001) extended the LCSS approach to trajectories and proposed three new similarity functions:

1 trajectories are similar when they are close to each other and, within a given tolerance, represent the same path

2 trajectories are similar if, independent of their extent and location, their change in orientation and movement is similar

3 trajectories are similar if they follow a similar path, but are translated from one another.

By relaxing the requirement for an exact match and allowing limited deviation in space and time, Sclaroff et al. were able to provide better complexity bounds for their versions of LCSS compared to exact match approaches. For their third similarity function, Sclaroff et al. additionally proposed an approximation algorithm to estimate the amount of translation at which two trajectories match within a given error.

Vlachos et al. (2002) built on top of this approach and provided a more extensive analysis. Shim and Chang (2003) extended the basic LCSS approach with a k-warping technique, allowing up to k replications of motion segments in the query trajectory in order to match similar trajectories. Further, Zeinalipour-Yazti et al. (2006) addressed the interesting problem of querying similar trajectories from a bulk of trajectory fragments

that are distributed across a network of nodes. Their approach performs localised top-*k* queries using LCSS to find the *k* most similar trajectory fragments at each node and then combine only fragments that are associated with the same trajectory across all nodes.

Edit distance techniques are powerful approaches for determining similarity between two sequences of discrete items. However, for large datasets they become computationally very expensive. In contrast, approaches applying sequential pattern mining, as described in the next subsection, are significantly more efficient. They are also better suited for data processing frameworks, such as data warehousing and OLAP as they are very similar to other pattern mining approaches already available in such systems.

2.4 Frequent pattern mining

Another, recently very common, approach to identifying groups of trajectories is pattern mining. Pattern mining is a method that has its origins in the mining of association rules from large sets of transactional data (Agrawal et al., 1993). It is often described in the context of mining frequent sets of items from shopping baskets to identify items or products that are frequently bought together (Han and Kamber, 2001). It has also been shown that the pattern mining approach is applicable to identifying patterns in sequence databases (Agrawal and Srikant, 1995). The analysis of sequential data is a requirement for a wide area of applications including, for example, the analysis of DNA or protein sequences, data streams in telecommunication, or tracking of diseases. A number of approaches that focus on the analysis of sequence data have been proposed and show an improved performance in identifying patterns when compared to transactional approaches. Most recent representatives of these approaches are PrefixSpan introduced by Pei et al. (2001, 2004) and CloSpan introduced by Yan et al. (2003), as well as an algorithm suitable for noisy data streams by Yang et al. (2002). A special case of identifying patterns in sequence data is the identification of patterns that occur periodically in constant time intervals. This special case of sequential pattern mining has been addressed by Han et al. (1999) and Ma and Hellerstein (2001).

One of the first approaches that utilises frequent pattern mining for the analysis of spatiotemporal databases was introduced by Tsoukatos and Gunopulos (2001). Their algorithm is based on the SPADE algorithm (Zaki, 2001), which was originally proposed for mining frequent subsequences from sequence databases. Both algorithms represent the search space for frequent subsequences as a lattice and then traverse the lattice to identify frequent subsequences that occur in the dataset. The two algorithms differ by the method that is used to search the lattice and by the results they produce. The SPADE algorithm performs a lattice-decomposition to obtain sublattices on which it can perform localised in-memory breadth-first search to find all frequent subsequences. Tsoukatos and Gunopulos's algorithm, on the other hand, uses a depth-first search on the entire search space lattice and finds only maximal frequent subsequences, i.e., frequent subsequences with the maximal number of items for which no supersequences exist that are also frequent. As outlined in the remainder of this section, many authors have since focused on developing improved algorithms for the mining of patterns from spatiotemporal databases and in particular moving object databases (Agrawal et al., 1993; Cao et al., 2005; Gidófalvi and Pedersen, 2006; Han et al., 2004; Hwang et al., 2005; Mamoulis et al., 2004; Peng and Chen, 2003; Wang et al., 2003).

Other than in classical subsequence mining, the data stored in moving object databases is inherently noisy and exact subsequences are rarely found. Wang et al. (2003) and Hwang et al. (2005) introduced algorithms specifically for the detection of grouping behaviour among moving objects. In both cases the algorithms are intended to find patterns which resemble the movement of multiple objects in unison along similar trajectories. In Wang et al. (2003), the trajectories are represented as sequences of observation points and Wang et al. proposed algorithms which identify similar trajectories by permitting a certain level of spatial and temporal mismatch when comparing observation points of two trajectories. The two algorithms Wang et al. proposed have been derived from the well-known frequent pattern mining algorithms Apriori (Agrawal et al., 1993) and FP-growth (Han et al., 2004) and extended to allow for the desired 'fuzziness' when comparing observation points. The Apriori algorithm exploits the property of frequent itemsets, that every subset of a frequent itemset is also a frequent itemset, to efficiently prune the search space for frequent itemsets. The FP-growth algorithm, on the other hand, first encodes the dataset in a data structure called FP-tree and then extracts frequent itemsets from this data structure directly.

Hwang et al. extended the algorithms proposed in Wang et al. (2003) by representing trajectories as sequences of line segments rather than observation points, such that for each point along a trajectory the distance to another trajectory can be determined. This approach improves the quality of results compared to those obtained in Wang et al. (2003) and is in particular superior when observations are sparsely distributed along trajectories. Similarly, Cao et al. described an approach allowing trajectory data to be noisy and imprecise by employing a line simplification method which locally removes segment points from a trajectory if they are within a certain distance to the resulting trajectory. Frequent patterns are then determined based on the simplified trajectories (Cao et al., 2005).

With a focus on mining periodic spatiotemporal patterns, Mamoulis et al. (2004) proposed two algorithms: STPMine1 and STPMine2. Both algorithms consider trajectories as sequences of locations that have been sampled in uniform time intervals. To determine frequent periodic patterns within these trajectories, both algorithms first employ a clustering of locations that have been sampled at time steps that are multiples of a fixed period length apart from each other. The clustering determines dense regions of locations, and regions with a number of locations below the minimum support are discarded. The remaining regions are then considered frequent one-itemsets from which the STPMine1 (Apriori-based) and STPMine2 (FP-growth-based) algorithms can generate all remaining frequent itemsets. While the generation of the remaining frequent itemsets is efficient, especially when using the STPMine2 algorithms, the generation of the initial frequent one-itemsets is costly for both algorithms as it requires the use of expensive clustering methods.

Two interesting application scenarios for the mining of patterns from moving object databases are described by Peng and Chen (2003) and Gidófalvi and Pedersen (2006). Peng and Chen addressed the problem of data allocation in mobile communication systems and showed how the identification of frequent patterns in the movement of the system's users can help to optimise the allocation of system resources and improve the system's performance (Peng and Chen, 2003).

In Gidófalvi and Pedersen (2006), Gidófalvi and Pedersen focused on the problem of identifying rideshare opportunities for commuters. Their dataset consisted of long-term route information for a number of cars that were equipped with GPS recording devices.

Gidófalvi and Pedersen's approach employs a frequent pattern mining algorithm that finds closed frequent itemsets and is based on a database projection method allowing the algorithm to be entirely implemented in SQL and executed on the database system storing the actual trajectory data.

In summary, frequent pattern mining approaches are most similar to the methods presented in this paper. Many frequent pattern mining approaches have shown good results when identifying patterns that are shared between individual objects. However, they often do not produce results suitable for the moving object data analysis studied in this paper. The most significant reason for this is the amount of redundant information generated by frequent pattern mining. For example, even with only a small amount of noise present in the moving object dataset, it is documented, e.g., in Gidófalvi and Pedersen (2006), that a lot of frequent patterns are detected that do not provide any valuable information. Additionally, frequent pattern mining algorithms consider each pattern as an independent piece of information and do not take into account any relationships that may exist between the detected patterns. In many scenarios, interesting patterns may be related to each other by the trajectories they share. These relationships are of importance in applications such as disease tracking since they may characterise groups of people who potentially have communicated a disease virus through transitive relationships. The methods presented in this paper aim at solving these shortcomings.

3 GROUP_TRAJECTORIES: a new operator for OLAP on moving object data

Consider N moving objects, each identified by a unique tag number i. Object movements are recorded through a set of readings $(i, t, (x, y))$ indicating that object i was located at position (x, y) at time t. The N moving objects are represented by a relational table *objects* with N records. Each record contains attributes such as *tag*, *name*, *size*, *colour*, etc. describing an object in a traditional relational manner that can be represented as a star schema (Chaudhuri and Dayal, 1997). Among these attributes is an attribute *trajectory* representing the movement of the respective object as a sequence $[(x_1, y_1, t_1), (x_2, y_2, t_2),\ldots,(x_m, y_m, t_m)]$ of positions at times $t_1, t_2,\ldots,t_m$. Our goal is to group objects with respect to attribute dimension *trajectory*. For this purpose, we define a new operator GROUP_TRAJECTORIES, which returns for each trajectory a group identifier, and then proceeds with standard OLAP aggregation according to the group identifiers instead of the trajectories themselves.

In this section, we present two different implementations of the operator GROUP_TRAJECTORIES which compute groups of trajectories that are appropriate for OLAP analysis in different application scenarios: *group by overlap* and *group by intersection*.

3.1 General framework and preprocessing

We begin with an outline of the high-level framework and preprocessing steps for the two implementations of the GROUP_TRAJECTORIES operator. The details of these implementations are then discussed in Sections 3.2 and 3.3.

The analysis of trajectories begins with a preprocessing step. Note that, the time complexity is quadratic in the number of trajectories. The goal of this preprocessing step

is to prepare the set of input trajectories T for the further processing by one of the implementations of our GROUP_TRAJECTORIES operator. The preprocessing stage is composed of three parts:

1 the mapping of trajectories to sequences of distinct items

2 the extraction of frequent patterns from the set of mapped trajectories

3 the reverse association of frequent itemsets with sets of trajectories that contain them.

As discussed earlier, the readings of real-world trajectories are often inherently noisy and even though two trajectories follow approximately the same path, the attribute values of their readings may differ significantly. This issue is addressed in the first part of the preprocessing stage by transforming the attribute values of each reading into corresponding discretised dimensions whose resolution is selected by the user. The motivation behind this transformation is two-fold:

1 noise and minor variances within trajectories are being compensated by mapping fine-grained coordinates to a coarser-grained grid

2 the user has control over the granularity of the mapping in an interactive manner.

This transformation of attribute values is also applied to the time dimensions, allowing trajectories to be analysed at different levels of time resolution. For example, the time granularity 'day' may be sufficient for a high-level analysis of GPS data for the movement of a fleet of ships. An analysis of the set of paths taken by a group of ships entering a port, on the other hand, may require a time granularity 'minute'.

Following the mapping of each trajectory's time/position pairs into a discrete space, individual readings from different trajectories may now be mapped onto the same discrete attribute values and consequently can be considered equivalent. Based on this property, we now employ frequent itemset mining to identify sets of distinct time/position pairs that are shared among a minimum number of trajectories. We refer to each of these sets as a frequent itemset and call the number of trajectories which share the frequent itemset the 'support' of the frequent itemset. The minimum support an itemset must have to be considered frequent is user-specified and allows the user to control the size of initial groups of trajectories that should be considered for further analysis.

Additionally, we apply a threshold on the size of each frequent itemset which is controlled by the user. It allows the user to limit the frequent itemsets to those which contain a minimum number of items. The motivation for this threshold is that frequent itemsets with a larger number of distinct items corresponds to longer shared paths, while frequent itemsets with fewer items correspond to shorter shared paths. The rationale for this is that we assume longer shared paths to be more interesting than shorter paths. However, this method can be analogously applied to prefer shorter paths.

The last step of the preprocessing is dedicated to the mapping of frequent itemsets back to the sets of trajectories that contain them. For each frequent itemset f, we determine the set C of trajectories such that each trajectory contains all of the time/position pairs in f. We let $\mathcal{C}$ be the set of all pairs (f, C), where f is a frequent itemset and C is the corresponding set of trajectories. After the completion of the preprocessing phase, the group merging phase which employs our new GROUP_TRAJECTORIES operator follows. The details of our two versions of the GROUP_TRAJECTORIES

operator, *group by overlap* and *group by intersection*, are discussed in the following subsections.

3.2 Group by overlap

The *group by overlap* method, shown in Algorithm 1, introduces a parameter called the *overlap ratio threshold* (ORT) that controls the strength of the grouping process. The interactive OLAP environment discussed in Section 4 allows for an interactive modification of this parameter. The method uses a relationship graph Γ whose vertices correspond to the trajectories. For each frequent itemset f and corresponding set C of trajectories, we consider all pairs of trajectories (t_i, t_j) with $t_i \in C$ and $t_j \in C$ and add an edge (t_i, t_j) to Γ if $\frac{2 \cdot |f|}{|t_i| + |t_j|} \geq ORT$. We call $\frac{2 \cdot |f|}{|t_i| + |t_j|}$ the overlap ratio of the trajectories. The intuition is to quantify the amount of overlap between two trajectories relative to their sizes. Figure 3(a) illustrates the relationship between trajectories and their overlap that is characterised by a frequent itemset. The resulting graph Γ then contains edges between those trajectories that have an overlap ratio of at least *ORT*. We then compute the connected components of graph Γ and report each connected component as a group of trajectories. The nature of the obtained groups of trajectories is characterised by two factors:

1 the *ORT* determines how much overlap two trajectories must have to be considered within the same group

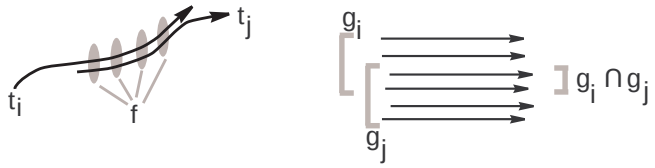
2 the construction of connected components captures transitivity among trajectories and thus cascaded 'relay'-type movements as illustrated in Figure 2(b).

Depending on the chosen *ORT*, objects will have to move in unison for more or less of their trajectories to form cascading trajectories.

Algorithm 1 Group by overlap

Input:	Set T of trajectories. Set $\mathcal{C}$ of mappings from frequent itemsets to sets of trajectories as determined in Section 3.1. Overlap ratio threshold *ORT*.	
Output:	Set of groups $\mathcal{G}$	
	Build relationship graph $\Gamma = (V_\Gamma, E_\Gamma)$	
	1	Initialize set of vertices $V_\Gamma \leftarrow T$
	2	Initialize set of labeled edges $E_\Gamma \leftarrow \emptyset$,
	3	**for all** $(f, C) \in \mathcal{C}$ **do**
	4	**for all** pairs (t_i, t_j) with $t_i \in C$ and $t_j \in C$ **do**
	5	Add an edge (t_i, t_j) to E_Γ if $\frac{2 \cdot \|f\|}{\|t_i\| + \|t_j\|} \geq ORT$
	6	**end for**
	7	**end for**
	Determine overlap groups in Γ	
	8	Compute connected components $\mathcal{G}$ of graph Γ
	9	Remove singletons from $\mathcal{G}$
	10	**return** $\mathcal{G}$

Figure 3 Illustration of (a) overlap ratio and (b) intersection ratio



(a) (b)

3.3 Group by intersection

The *group by intersection* method is shown in Algorithm 2. It introduces a parameter called the *intersection ratio threshold* (IRT), which is used to control how aggressively groups are formed. In an interactive OLAP environment, this parameter can be modified interactively as part of the explorative analysis process. The *group by intersection* algorithm first creates an initial set $\mathcal{G}$ of groups of trajectories, where each group C is the set of trajectories associated with a frequent itemset *f* as determined in the preprocessing step discussed in 3.1. Each group *C* is assigned a group strength *GS*(*C*), which is initially set to the size of the corresponding frequent itemset. Using the unweighted size of the frequent itemset allows us later to identify groups that are characterised by long frequent itemsets. The remainder of our method merges groups in $\mathcal{G}$ by iterating the following loop (Lines 6–20 in Algorithm 2): for each pair $g_i, g_j \in \mathcal{G}$, compute the *intersection ratio*

$$IR\left(g_i, g_j\right) = \min\left(\frac{\left|g_i \cap g_j\right|}{\left|g_i\right|}, \frac{\left|g_i \cap g_j\right|}{\left|g_j\right|}\right),$$

which represents the number of trajectories that occur in both g_i and g_j, relative to the sizes of g_i and g_j. The intuition behind this definition is that pairs of groups that share a large percentage of their trajectories are more strongly related to each other than pairs that do not share as many trajectories relative to their group sizes. Figure 3(b) illustrates a pair of trajectory groups and the trajectories they share. After computing the intersection ratio, we consider only those pairs (g_i, g_j) as candidates for merging whose intersection ratio is larger than the IRT. The parameter allows to control how strongly related two trajectory groups are required to be to qualify for merging. For each qualifying pair (g_i, g_j), we compute its *merge strength*, which is the average of their group strength values:

$$MS\left(g_i, g_j\right) = \frac{GS\left(g_i\right) + GS\left(g_j\right)}{2}.$$

The merge strength represents the average support in terms of shared locations the new group receives from each contributing group independent of the actual number of trajectories in the group. We chose not to weight the merge strength by the size of the groups that contribute to the merged group, as the goal of the merging is to identify groups that are supported by long frequent itemsets. The influence of the number of trajectories in each group is only intended to impact the *intersection ratio* and otherwise

have no influence on the grouping process. Hence, the number of trajectories in each group is not considered when computing the *merge strength*.

Algorithm 2 Group by intersection

Input: Set $\mathcal{C}$ determined in Section 3.1. Intersection Ratio Threshold *IRT*.

Output: Set of groups $\mathcal{G}$.

Create initial set of intersection groups

$\mathcal{G} \leftarrow \emptyset$,
for all $(f, C) \in \mathcal{C}$ **do**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{C\}$
  set initial Group Strength $GS(C) = |f|$
end for

Merge intersection groups

repeat
  for all $g_i, g_j \in \mathcal{G}, g_i \neq g_j$ **do**
    set Intersection Ratio

$$IR\left(g_i, g_j\right) = \min\left(\frac{|g_i| \cap |g_j|}{|g_1|}, \frac{|g_i| \cap |g_j|}{|g_2|}\right)$$

    if $IR(g_i, g_j) > IRT$ **then**
      set Merge Strength

$$MS\left(g_i, g_j\right) = \frac{GS\left(g_i\right) + GS\left(g_j\right)}{2}$$

    else
      set Merge Strength $MS(g_i, g_j) = 0$
    end if
  end for
  find $(g_{i'}, g_{j'})$ for which $MS(g_{i'}, g_{j'})$ is maximal
  if $MS(g_{i'}, g_{j'}) \neq 0$ **then**
    $\mathcal{G} \leftarrow (\mathcal{G} \setminus \{g_{i'}, g_{j'}\}) \cup \{g_{i'} \cup g_{j'}\}$
    set Group Strength
    $GS(g_{i'} \cup g_{j'}) = MS(g_{i'}, g_{j'})$
  end if
until $MS(g_{i'} \cup g_{j'}) = 0$
return $\mathcal{G}$

All candidate pairs are then ranked by their merge strength and a pair $(g_{i'}, g_{j'})$ with maximum merge strength is merged to produce a new group. The intuition for using a pair with maximum merge strength is that the average support in terms of shared locations in the resulting group is highest. This implies that trajectories within the group are on average supported by more shared locations than in other groups, and are thus characterised by longer frequent itemsets. The group strength $GS(g_{i'} \cup g_{j'})$ of the new

merged group is the merge strength $MS(g_{i'}, g_{j'})$ of the pair $(g_{i'}, g_{j'})$. This process is repeated until there are no more pairs of groups with intersection ratio larger than the IRT and non-zero merge strength.

The *group by intersection* method aggregates subsets of trajectories that constitute a movement of individual objects in parallel, similar to that of a marching band or a flock of birds. The nature of the obtained groups of trajectories is characterised by:

1 the IRT, which determines how many shared trajectories between two groups are 'sufficient' for them to be merged

2 the merge strength, which favours the merging of groups that are supported by long frequent itemsets.

Thus, groups that are merged do not only share trajectories, but share trajectories that are supported by long and interesting frequent itemsets. We first merge those candidate groups whose shared trajectory segments are longest and then proceed top-down, merging groups with shorter shared trajectory segments. Unlike the *group by overlap* method, which combines sequences of trajectories, the *group by intersection* method combines parallel trajectories.

4 Interactive OLAP for trajectories

The algorithms for the two different versions of the GROUP_TRAJECTORIES operator presented in Section 3 are guided by the following parameters:

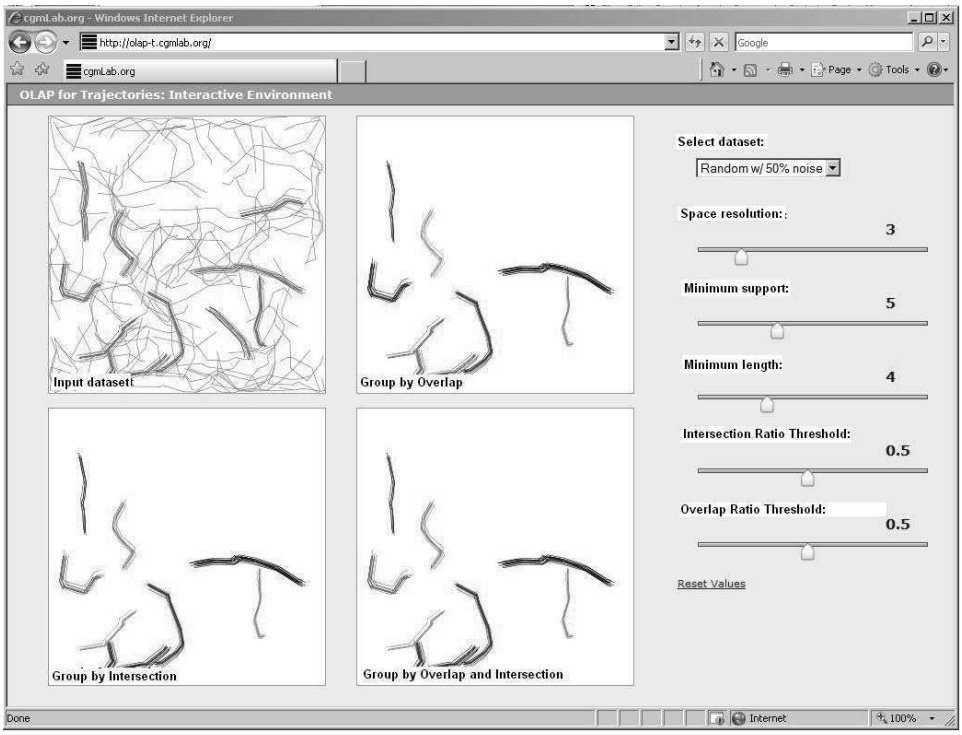
- space resolution
- minimum support of a frequent itemset
- minimum trajectory length
- IRT
- ORT.

This provides an opportunity for an interactive OLAP analysis of groups of trajectories at various levels of resolution or 'connectedness'. For example, consider the analysis of tracking data from GPS-enabled trucks. A high-level analysis of trajectories belonging to trucks that travel long distances may be sufficient at a time-dimension granularity of 'day'. However, an analysis of trajectories that belong to trucks within an urban area may require a time-dimension granularity of 'minute' to capture more detailed movement patterns.

We implemented a prototype real-time interactive environment to provide users with the ability to browse the parameter space and dynamically tune the parameters based on application specific knowledge. As shown in Figure 4, all parameters can be dynamically modified by moving sliders, with the clusters being displayed interactively and in real-time. Our code was written in C, using the Cairo library (http://www.cairographics.org/) for visualisation. Some utility code was written in Ruby. We used the AFOPT C++ library (Liu et al., 2003) for frequent pattern mining and the user interface was implemented in HTML and JavaScript. Our prototype ran on a dual-processor dual-core AMD Opteron 2.2 GHz-based machine. In order to achieve

real-time response, the relationship graph is constructed as a preprocessing step, where all edges are labelled with parameter ranges for which they are valid.

Figure 4 Screenshot of the interactive environment for *OLAP for trajectories*



As an example for browsing a parameter like the ORT, consider a set of trajectories representing movements of people who pass on a disease virus. The grouping determined by the *group by overlap* method could be used to analyse the total movement of the virus. In this example, the ORT would represent the amount of interaction between individuals required to pass on the virus. Changing the threshold value allows one to evaluate how far the virus will spread based on different assumptions about its transmission. A prototype of an interactive environment for the OLAP analysis of trajectories is shown in Figure 4. The interactive environment allows for resolution drill-down and roll-up as well as parameter browsing. The system visualises an implementation of the two GROUP_TRAJECTORIES operators presented in this paper. It does not yet include other OLAP functionality, such as spatiotemporal aggregation or aggregation with respect to other feature dimensions. The system allows the user to explore the results of the GROUP_TRAJECTORIES operators depending on different resolution and threshold values for several synthetic and real-life datasets. The user can load a dataset and then move the sliders representing the different parameters to observe the results for different parameter settings. For example in a ride share application, the space resolution would represent the distance people are willing to walk to their ride share. When moving the space resolution slider, our system interactively displays the different sets of sharable rides for different walk distances.

5 Experimental evaluation

The experimental evaluation of the GROUP_TRAJECTORIES operators presented in this paper is divided into four parts. Sub-Section 5.1 provides a detailed analysis of each algorithm using appropriate examples. In sub-Section 5.2, the robustness of the GROUP_TRAJECTORIES operators against background noise is evaluated. Sub-Section 5.3 then investigates the influence of the algorithms' parameters on the results produced. Finally, sub-Section 5.4 concludes the experimental evaluation with results obtained using real-world data and compares them to results obtained using frequent pattern mining only.

5.1 Detailed analysis

In this section, we evaluate our algorithms with respect to a variety of synthetic scenarios that are carefully designed to demonstrate the strengths and weaknesses of the grouping algorithms. Each scenario addresses a specific property of the dataset and evaluates the influence of changing that property on the results obtained by both of our algorithms.

5.1.1 Overlap

The overlap property represents the number of shared frequent items (grid cells) between at least two trajectories compared to their total lengths. This property is used by the *group by overlap* algorithm to identify trajectories that should be grouped together.

Figure 5 shows two datasets, each containing three frequent itemsets with a minimum support of 2. The frequent itemsets are *closed*, that is, there are no frequent itemsets with the same or greater support that are supersets of the frequent itemsets shown. Both datasets are identical except for the change in the length of the overlapping trajectories that appear to connect the 'left' and the 'right' side of each dataset. Note, the number of shared locations in frequent itemset $\mathcal{F}_3$ changes from $|\mathcal{F}_3| = 1$ to $|\mathcal{F}_3| = 2$.

Figure 5 Two datasets each containing three closed frequent itemsets but with different associated overlap (see online version for colours)

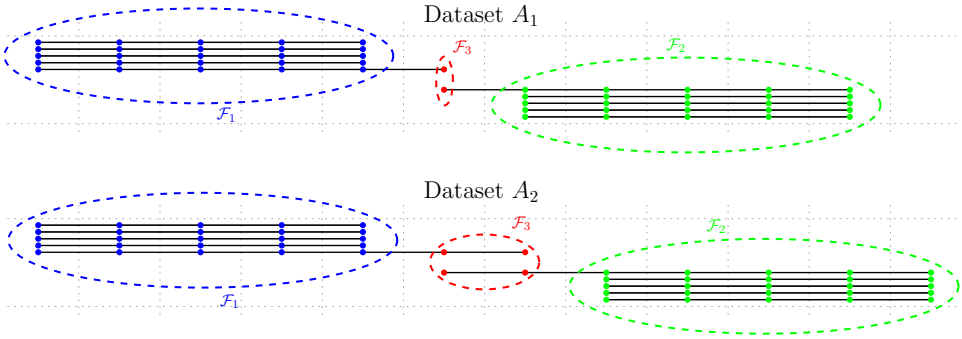
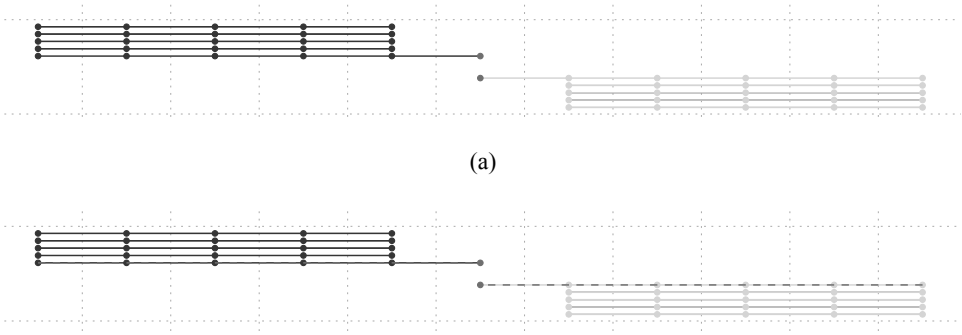


Figure 6 shows the resulting groups for each of our algorithms when applied to dataset A_1. For the *group by overlap* algorithm we chose an ORT of $ORT = 0.25$ and for the *group by intersection* algorithm we chose an IRT of $IRT = 0.25$. None of our algorithms combines the groups on either side of the overlap as the overlap ratio $OR = 1 / 6$ does not

satisfy the ORT of $ORT = 0.25$. Similarly, the intersection ratio of $IR = 1 / 5$ for the association of the overlapping trajectories with the shorter trajectories does not satisfy the threshold $IRT = 0.25$ required for grouping by intersection. Hence, for the *group by intersection* algorithm each of the overlapping trajectories is present in two groups (represented as dashed lines in Figure 6), one group for each closed frequent itemset it shares.

Figure 6 Grouping results for dataset A_1 with *ORT* and *IRT* set to 0.25 respectively, (a) group by overlap (b) group by intersection



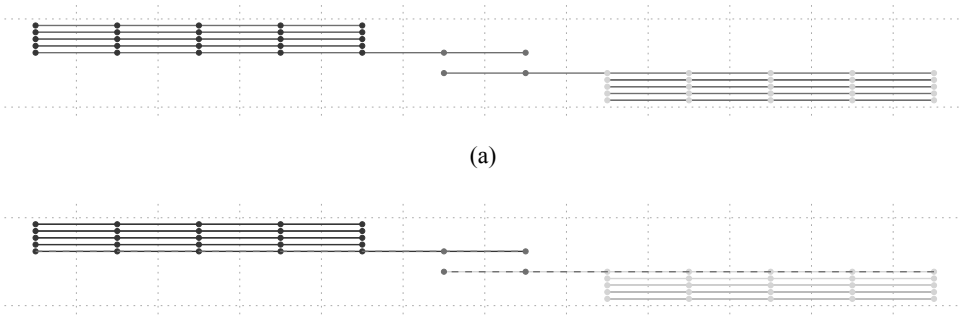
(a)

(b)

When increasing the length of the overlap between the 'left' and the 'right' side from $| \mathcal{F}_3 | = 1$ to $| \mathcal{F}_3 | = 2$ shared frequent items in dataset A_2, the overlap ratio for the overlapping segment changes from 1 / 6 to 2 / 7 and consequently satisfies the ORT of $ORT = 0.25$.

This results in the merging of the 'left' and 'right' side into a single group by the *group by overlap* algorithm, as shown in Figure 7. The result of the *group by intersection* algorithm does not change, as the length of the overlap does not impact the intersection ratio.

Figure 7 Grouping results for dataset A_2 with *ORT* and *IRT* set to 0.25 respectively, (a) group by overlap (b) group by intersection



(a)

(b)

Note: The change in overlap results in a different grouping for *group by overlap* when compared to dataset A_1.

5.1.2 Intersection

The *intersection ratio* between two groups is the number of trajectories shared between the groups relative to the size of the largest of the two groups. This property is used by the *group by intersection* algorithm to recursively identify groups that can be merged.

Figure 8 illustrates two datasets, each containing three closed frequent itemsets with a minimum support of 4. Both datasets are identical except for two additional trajectories added to dataset B_2. The added trajectories have an influence on the intersection ratio between the initial groups that are formed from the three frequent itemsets.

Figure 8 Changing the *intersection ratio* by changing the number of parallel trajectories (see online version for colours)

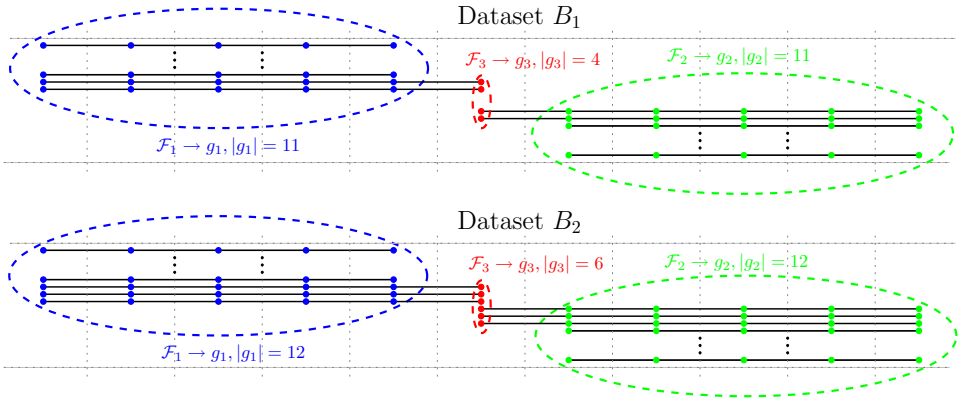
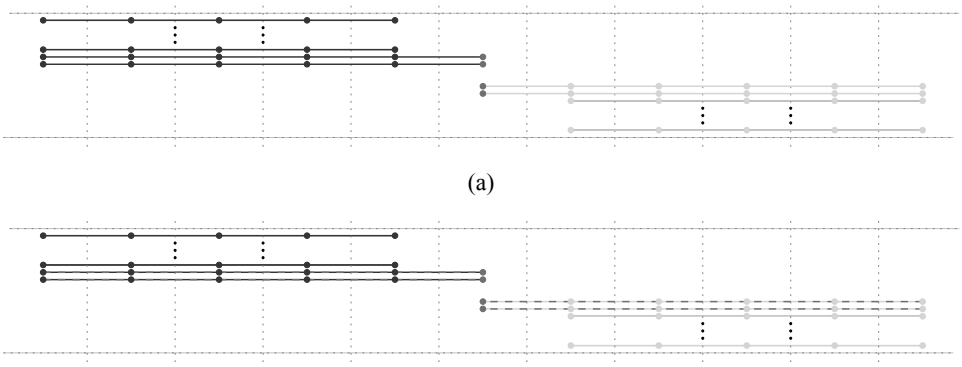


Figure 9 Grouping results for dataset B_1 with *ORT* and *IRT* set to 0.20 respectively, (a) group by overlap (b) group by intersection



(a)

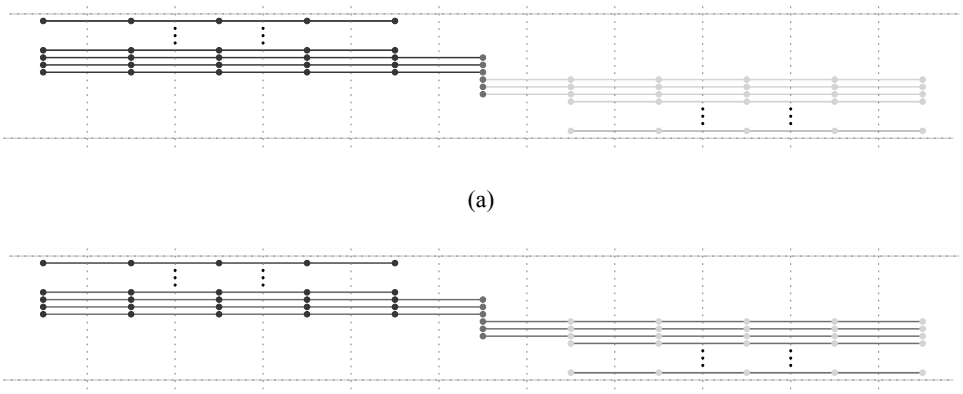
(b)

Figure 9 shows the results for each of our grouping algorithms when there are only two trajectories on every side overlapping with the trajectories of the other side (dataset B1). The *ORT* and *IRT* were both chosen to be 0.2. For the *group by intersection* algorithm, the initial groups g_1, g_2 and g_3 are created from the frequent itemsets $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ respectively. Since, the size of the intersection between groups g_1 and g_3, or g_2 and g_3 is only of size 2, and g_1 and g_2 each have 11 trajectories, the intersection ratio of $IR = 2 / 11$

does not satisfy the requirement of *IRT* = 0.2. Hence, the *group by intersection* algorithm does not combine the groups into a single group. Similarly, the overlap of $|\mathcal{F}_3| = 1$ is not sufficient to match the required ORT of *ORT* = 0.2 for the *group by overlap* algorithm.

However, the *group by intersection* algorithm is able to combine groups if the size of the intersection is increased such that the requirement of *IRT* = 0.2 is satisfied for candidate groups. For the first merge iteration, the example contains two candidate groups $g_1 \cup g_3$ and $g_2 \cup g_3$, both with an intersection ratio of 3 / 12 = 0.25. Since both candidate groups have the same merge strength, either group can be chosen to be created first. Assume group $g_1 \cup g_3$ is chosen to be created first. The next merge iteration then contains a single candidate group $(g_1 \cup g_3) \cup g_2$ with an intersection ratio of 3 / 15 = 0.2. The intersection ratio 0.2 satisfies the requirement *IRT* = 0.2, and the group is created, resulting in a single group containing all trajectories in dataset B_2 as shown in Figure 10. In this scenario the *group by overlap* algorithm is not able to combine the groups, as the size of the intersection does not have an influence on the overlap ratio between two trajectories.

Figure 10 Grouping results for dataset B_2 with *ORT* and *IRT* set to 0.2 respectively, (a) group by overlap (b) group by intersection



(a)

(b)

Note: The increase of intersection size results in a different grouping for *group by intersection* when compared to dataset B_1.

The experiments discussed in this section demonstrate the detailed behaviour of our algorithms for carefully designed synthetic datasets and explore various boundary cases of our algorithms. The experiments show the different properties of the input dataset for which either our *group by overlap* algorithm or our *group by intersection* algorithm is better suited to identify groups. For the *group by overlap* algorithm, the experiments show that the algorithm favours the combination of groups whose trajectories have a sufficient amount of locations in common. It can therefore be used to identify groups of trajectories that are formed by sequentially connected trajectories where each 'connection' must be of a certain length in comparison to the length of each trajectory in the group. The experiments for the *group by intersection* algorithm, on the other hand, show that the algorithm is better suited for identifying groups that are composed of

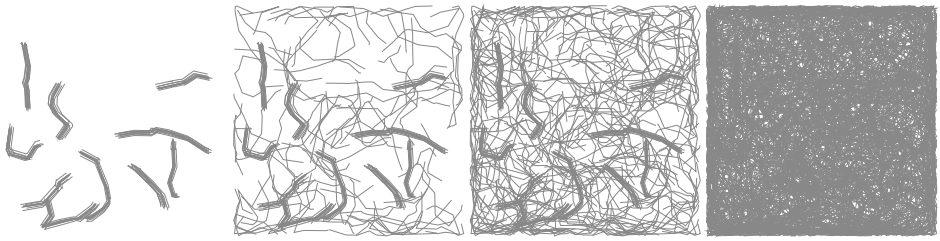
trajectories that run in parallel. It favours the merging of groups that have entire trajectories in common, i.e., trajectories that run in parallel, and the number of trajectories in common is at least a given fraction of the size of each group. Thus, if the goal is to identify groups that are formed by the partial overlap of trajectories, the *group by overlap* algorithm should be applied. However, if groups should be identified that have many parallel trajectories, then the *group by intersection* algorithm is a more appropriate choice.

5.2 Robustness against noise

We tested the robustness of the GROUP_TRAJECTORIES implementations against background noise. For that, we created a synthetic dataset of ten groups with ten trajectories each and then added random trajectories as background noise. Figure 11(a) shows four input datasets, all containing the same groups but each with a different amount of background noise. For this example we have chosen datasets with 0% (no noise), 50%, 75% and 95% noise. The percentage of noise is specified with respect to the total number of trajectories in the dataset. The subject of the evaluation is: What level of noise can be present in the input data while maintaining a correct result? Here, correctness means that GROUP_TRAJECTORIES reports the original groups and discards the randomly added trajectories. Figure 11(b) shows the initial groups obtained by the frequent itemset mining algorithm before applying the *group by overlap* or *group by intersection* algorithm. The frequent itemset mining algorithm identifies 51 groups for noise levels of 0%, 50% and 75% and 52 groups for a noise level of 95%. While it eliminates most of the noise from the input data, it does not identify a suitable grouping of the remaining trajectories as it does not capture the relationships between trajectories characterised by different frequent itemsets. The *group by overlap* algorithm on the other hand identifies for each dataset the correct set of distinct groups [see Figure 11(c)]. It improves on the grouping obtained from frequent itemset mining and the groups it identifies sufficiently represent the original groups. Only 16% of trajectories that belong to groups in the input dataset have not been captured by the algorithm (false negatives). However, this can be attributed to the quality of the frequent itemset mining algorithm, as those trajectories are not characterised by any frequent itemset and thus could not have been included in the final grouping. For this example, at 95% noise, the frequent itemset mining also classifies only a single trajectory (1.18%) as a false positive, which subsequently is reported as part of a final result group determined by the *group by overlap* algorithm.

Figure 11(d) shows the final groups identified by the *group by intersection* algorithm. Even though it improves the grouping determined by frequent itemset mining, it does not produce the same quality of groups as the *group by overlap* algorithm. For the 0% and 50% noise levels, it produces 12 final groups; and for 75% and 95% noise levels, it produces 13 final groups, resulting in some trajectories not being grouped together in the same group, though still being reported as a separate group. Depending on the application scenario, this deficiency is acceptable, as the overall result is still superior when compared to results obtained by frequent itemset mining only. Note, at a noise level of 95% it becomes difficult for the human eye to visually detect the original groups; however, both GROUP_TRAJECTORIES methods still report very good results.

Figure 11 Groups identified by each of our algorithms from a dataset with varying levels of noise, (a) Input data with 0%, 50%, 75% and 95% noise respectively (b) Groups identified by frequent itemset mining (c) Groups identified by *group by overlap* with *ORT* = 0.5 (d) Groups identified by *group by intersection* with *IRT* = 0.5



(a)

(b)

(c)

(d)

Note: The input parameters were set to a space resolution of 5, a minimum support of 4 and a minimum frequent itemset length of 4.

5.3 Input parameters

In this section, we examine the influence of the algorithms' input parameters on the results produced. As input data we use a synthetic dataset that consists of a mix of groups of trajectories. Some groups are of the type that is best for *group by overlap*, and some

groups are of the type that is best for *group by intersection*. The dataset is shown in Figure 12. It consists of three spirals with parallel paths in each spiral. While a spiral-like movement is not a pattern commonly occurring in real world data, it represents a challenging pattern for our methods. Note the subdivision of the spiral into several colour-coded segments. Each such segment represents a group of trajectories that are moving in unison. To achieve a more realistic movement, a small random variance is added to the movement of each trajectory. Furthermore, each segment overlaps with the previous and the following segment.

Figure 12 Synthetic dataset with 24 groups each consisting of ten trajectories and a partial overlap of approximately 25%

In the following, we examine the key parameters that influence the results obtained by our algorithms, namely:

1 the resolution at which the frequent pattern mining is performed

2 the *ORT* parameter of the *group by overlap* algorithm

3 the *IRT* parameter of the *group by intersection* algorithm.

5.3.1 Resolution

The resolution parameter determines the discretisation of the space that is performed before the mining of the frequent itemsets from the trajectories. It is used to generalise the trajectories' locations such that locations that are mapped to the same grid cell are regarded as identical locations. This approach reduces noise and local variances in trajectories and allows for more meaningful frequent itemsets to be found. It has a strong influence on the quality of the final groups found by our algorithms. Figure 13 shows the results of both of our algorithms for the dataset described above, with a set of fixed parameters *min_support* = 4, *min_length* = 4, *ORT* = 0.2, *IRT* = 0.2, and a variable resolution between 2 and 8 bits across the extent of each dimension. We observe that the *group by overlap* algorithm indeed favours the overlapping spiral segments and eventually identifies one group for each spiral at resolution levels of 5 and 6 bits. Conversely, the *group by intersection* algorithm does not make use of the overlap between the segments and cannot clearly identify individual spirals. However, even at

higher resolutions when the generalisation of the trajectories becomes less effective, our Group by Intersection algorithm can still be used to identify individual segments of the spirals which constitute parallel movements. Note, that at lower and higher resolutions the sizes of the detected groups significantly decrease compared to medium resolutions. This can be attributed to the fact that at lower resolutions most frequent itemsets do not satisfy the minimum length of at least four items and at higher resolutions frequent itemsets do not satisfy the minimum support of four trajectories due to the variance in the trajectories.

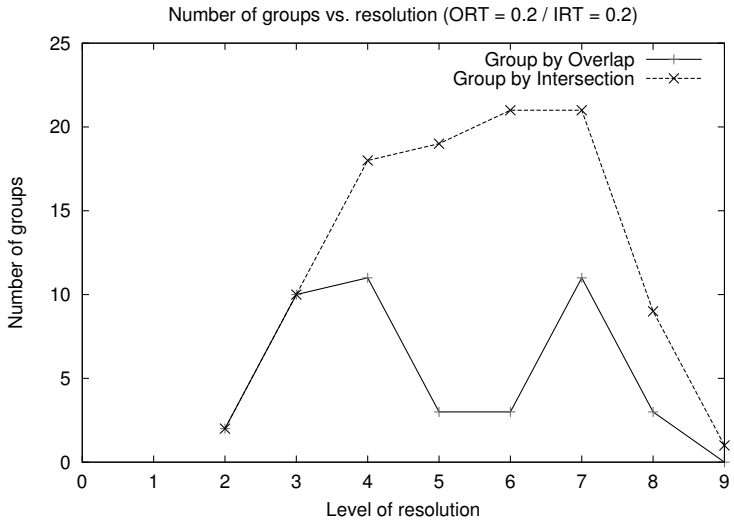
Figure 13 Groups identified by each of our algorithms at levels of resolution between 2 and 8 (left to right), (a) group by overlap (b) group by intersection

(a)

(b)

Note: Fixed parameters are *min_support* = 4, *min_length* = 4, *ORT* = 0.2, *IRT* = 0.2.

Figure 14 Number of groups each algorithm identifies for the given input dataset depending on the resolution



Note: *Group by overlap* tends to identify fewer but larger groups, *group by intersection*, on the other hand, identifies more but smaller groups.

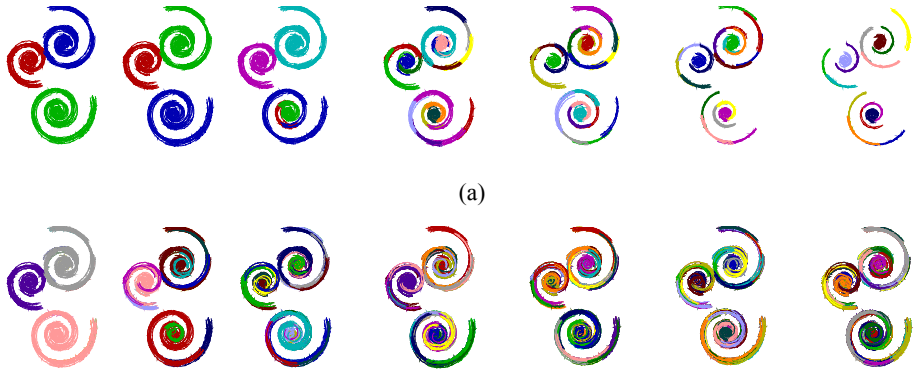
Figure 14 illustrates the relationship between the level of resolution and the number of groups identified by each algorithm. It can be observed that the *group by intersection* algorithm tends to identify many but smaller groups while *group by overlap* favours fewer but larger groups.

5.3.2 ORT and IRT

The *ORT* and *IRT* input parameters for our GROUP_TRAJECTORIES implementations influence their sensitivity towards identifying groups.

Using the dataset in Figure 12, we tested our GROUP_TRAJECTORIES implementations for various values of ORT and IRT. The results are shown in Figure 15.

Figure 15 Groups (identified by colour) computed by both of our methods for *ORT* = *IRT* = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 (*min_support* = 4, *min_length* = 4), (a) group by overlap (b) group by intersection



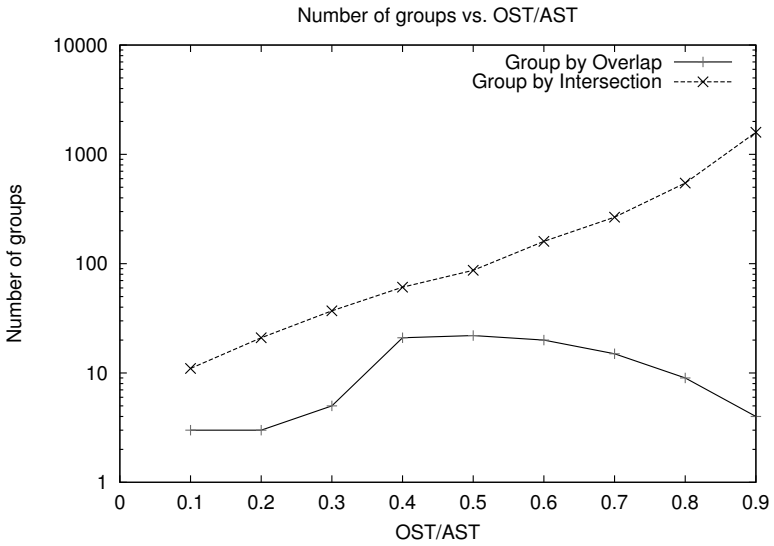
(a)

(b)

We observe that for lower values of *ORT* and *IRT*, the identified groups become larger in size, but fewer groups are identified. This behaviour is expected, as a lower threshold provides less constraints on the formation of groups. For larger values for *ORT* and *IRT* on the other hand, the algorithms become more restrictive in terms of identifying and merging groups, and other more subtle patterns are detected.

In the example shown in Figure 15, the *group by overlap* method identifies larger groups for low values of *ORT*, and reports the entire spirals. The identified groups become closer to the initial set of groups of trajectories in the spirals as the value for *ORT* increases. For large values of *ORT*, there is insufficient overlap between parallel paths and the reported groups become smaller to a point where individual trajectories are not grouped any more and are being discarded as singletons. For the *group by intersection* method, we observe that the number of groups reported increases with increasing values for *IRT* as the algorithm becomes more and more discriminating between the initial groups formed by frequent itemset mining. A summary of the number of groups reported as a function of *ORT* = *IRT* is given in Figure 16.

Figure 16 Relationship between the number of identified groups and values for *ORT* and *IRT*

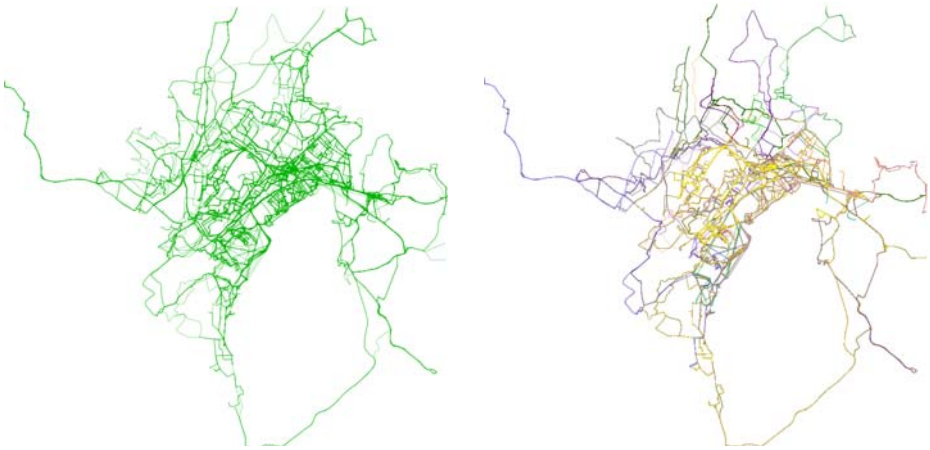


5.4 Real world data

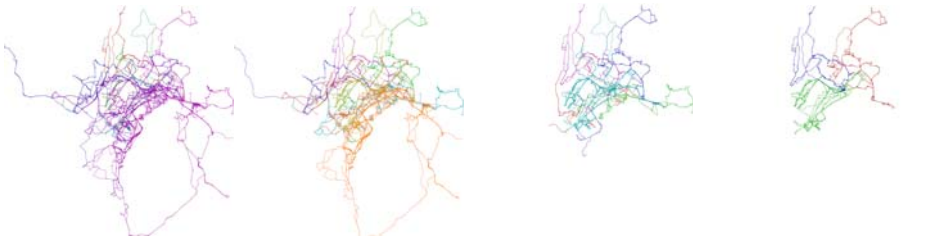
For the evaluation of our methods on real-world data, we have chosen the school buses dataset that can be freely obtained from Theodoridis (2003). The dataset contains 145 trajectories of buses that are moving in and around an urban area.

Figure 17(a) shows the majority of the dataset around the urban center (a few routes going to far out places were removed to better display the data). Figure 17(b) represents the 76 groups that are identified by applying only frequent pattern mining (as, e.g., in Gidófalvi and Pedersen, 2006; Mamoulis et al., 2004; Cao et al., 2005) plus a minimum length cutoff as used in our methods. The large number of groups reported by frequent-pattern-mining-based methods is often a disadvantage because it may lead to very little aggregation in an OLAP setting. Figure 17(c) shows the groups reported by the *group by overlap* method for *ORT* values 0.4, 0.5, 0.6, and 0.7. We observe that the parameter *ORT* in our *group by overlap* method allows for a much finer control over the grouping of trajectories reported and that this method reports a considerably smaller number of groups when compared to the original number of frequent patterns. Additionally, the groups found by the *group by overlap* algorithm are significantly more distinct than the original frequent itemsets and one can distinguish sets of trajectories that appear to be grouped due to their spatial proximity and the locations they share. For example, the orange group in Figure 17(c) for *ORT* = 0.5 primarily includes trajectories in the southern region of the sample dataset. For the larger *ORT* values of 0.6 and 0.7, however, this group is no longer identified as the overlap among trajectories in this group as it does not satisfy the ORT constraint anymore. The groups that remain for *ORT* values of 0.6 and 0.7 are spatially smaller and denser groups where many trajectories share locations within a smaller spatial region. This selectivity of groups depending on the values of *ORT* makes the *group by overlap* algorithm a good choice for the analysis of the given dataset. It furthermore enables an interactive exploration of moving object data and helps discovering hidden relationships among the moving objects.

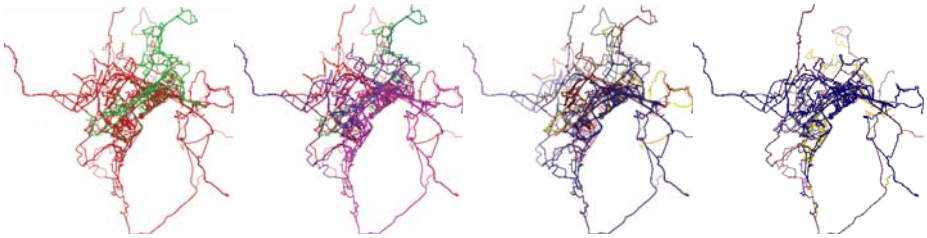
Figure 17 Results obtained for the school buses dataset, (a) Entire dataset (b) Groups reported (identified by colour) using frequent itemset mining only (76 groups) (c) Groups reported (identified by colour) using *group by overlap* and *ORT* = 0.4, 0.5, 0.6, 0.7 (*min_support* = 5, *min_length* = 30) (d) Groups reported (identified by colour) using *group by intersection* and IRT = 0.05, 0.35, 0.64, 0.95 (*min_support* = 9, *min_length* = 18)



(a) (b)



(c)



(d)

Figure 17(d) shows groups within the real-world dataset that have been identified using our *group by intersection* algorithm. At a first glance the groups that are identified by the *group by intersection* algorithm are not as distinct as those identified by the *group by overlap* algorithm. However, the *group by intersection* algorithm does exhibit a very subtle selection of groups with properties that satisfy the *IRT*. Assume the example dataset represents the trajectories of roaming individuals and the subject of the analysis is the spreading of an infectious disease, where a transfer of the disease is only likely when a critical mass of individuals in the same location is met. Given an *IRT* value of 0.05, i.e.,

a critical mass of 5% of the population in a particular location, we can see in the leftmost plot in Figure 17(d), that there are two distinct groups (red and green) which are able to carry the disease. If the critical mass increases to 65%, i.e., $IRT = 0.65$ (third plot from the left), we observe that the *group by intersection* algorithm identifies more distinct groups that can carry the disease, however, the population of each group is much smaller. Assuming the origin of a disease infection is known, the use of this algorithm can help to quickly identify populations that may be at risk.

6 Summary

In this paper, we have introduced a novel approach for the evaluation of *group-by* queries over trajectories to facilitate an OLAP-like analysis of moving object data. We introduced the concept of the GROUP_TRAJECTORIES *group-by* operator and provided two implementations of this operator. Our approach builds on top of the established frequent pattern mining method, which is readily available in many data warehousing systems, while improving its results by making them more suitable for interactive analysis. Each of our two algorithms is designed to group trajectories that exhibit a particular type of movement and we support this claim with a detailed experimental evaluation using synthetic and real-world datasets.

References

Agarwal, P.K., Arge, L. and Erickson, J. (2003) 'Indexing moving points', *Journal of Computer and System Sciences*, Vol. 66, No. 1, pp.207–243.

Agrawal, R. and Srikant, R. (1995) 'Mining sequential patterns', *Proceedings of the 11th International Conference on Data Engineering*, IEEE Computer Society, pp.3–14.

Agrawal, R., Imieliński, T. and Swami, A. (1993) 'Mining association rules between sets of items in large databases', *ACM SIGMOD Record*, Vol. 22, No. 2, pp.207–216.

Andersson, M., Gudmundsson, J., Laube, P. and Wolle, T. (2007) 'Reporting leadership patterns among trajectories', *Proceedings of the 2007 ACM Symposium on Applied Computing*, ACM, pp.3–7.

Anwar Hossain, Md. and Bazlur Rashid, A.N.M. (2012) 'Challenging issues of spatio-temporal data mining', *Computer Engineering and Intelligent Systems*, Vol. 3, No. 4, pp.55–63.

Baltzer, O. (2011) 'Computational methods for spatial OLAP', PhD thesis, Dalhousie University.

Baltzer, O., Dehne, F., Hambrusch, S. and Rau-Chaplin, A. (2008) 'OLAP for trajectories', *Proc. Database and Expert Systems Applications (DEXA)*, Springer, pp.340–347.

Benetis, R., Jensen, C.S., Karĉiauskas, G. and Ŝaltenis, S. (2006) 'Nearest and reverse nearest neighbor queries for moving objects', *The VLDB Journal The International Journal on Very Large Data Bases*, Vol. 15, No. 3, pp.229–249.

Benkert, M., Gudmundsson, J., Huebner, F. and Wolle, T. (2006) 'Reporting flock patterns', *Lecture Notes in Computer Science*, Vol. 4168, pp.660–671, Springer.

Cao, H., Mamoulis, N. and Cheung, D.W. (2005) 'Mining frequent spatio-temporal sequential patterns', *Proceedings of the 5th International Conference on Data Mining*, IEEE Computer Society, pp.82–89.

Chaudhuri, S. and Dayal, U. (1997) 'An overview of data warehousing and OLAP technology', *ACM SIGMOD Record*, Vol. 26, No. 1, pp.65–74.

Chen, J. and Meng, X. (2009) 'Update-efficient indexing of moving objects in road networks', *GeoInformatica*, Vol. 13, No. 4, pp.397–424.

Ding, Z. and Guting, R.H. (2004) 'Managing moving objects on dynamic transportation networks', *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, IEEE Computer Society, pp.287–296.

Ester, M., Kriegel, H-P., Sander, J. and Xu, X. (1996) 'A density-based algorithm for discovering clusters in large spatial databases with noise', *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pp.226–231.

Gidófalvi, G. and Pedersen, T.B. (2006) 'Mining long, sharable patterns in trajectories of moving objects', *STDBM '06: Proceedings of the 3rd Workshop on Spatio-Temporal Database Management*.

Gidófalvi, G. and Pedersen, T.B. (2009) 'Mining long, sharable patterns in trajectories of moving objects', *GeoInformatica*, Vol. 13, No. 1, pp.27–55.

Gómez, L.I., Kuijpers, B. and Vaisman, A.A. (2008) 'Aggregation languages for moving object and places of interest data', *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC08)*, ACM, pp.857–862.

Gudmundsson, J., van Kreveld, M. and Speckmann, B. (2004) 'Efficient detection of motion patterns in spatio-temporal data sets', *Proceedings of the 12th ACM International Workshop on Geographic Information Systems*, ACM, pp.250–257.

Güting, R.H. and Schneider, M. (2005) *Moving Objects Databases*, Morgan Kaufmann, San Francisco, CA, USA.

Hadjieleftheriou, M., Kollios, G., Tsotras, V.J. and Gunopulos, D. (2002) 'Efficient indexing of spatiotemporal objects', *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, pp.251–268.

Han, J. and Kamber, M. (2001) *Data Mining: Concepts and Techniques*, Morgan Kaufmann, Walttham, MA, USA.

Han, J., Dong, G. and Yin, Y. (1999) 'Efficient mining of partial periodic patterns in time series database', *Proceedings of the 15th International Conference on Data Engineering*, IEEE Computer Society, Vol. 99, pp.106–115.

Han, J., Pei, J., Yin, Y. and Mao, R. (2004) 'Mining frequent patterns without candidate generation: a frequent-pattern tree approach', *Data Mining and Knowledge Discovery*, Vol. 8, No. 1, pp.53–87.

Har-Peled, S. (2004) 'Clustering motion', *Discrete and Computational Geometry*, Vol. 31, No. 4, pp.545–565.

Hartigan, J.A. (1975) *Clustering Algorithms*, John Wiley & Sons, New York, NY, USA.

Hwang, S.Y., Liu, Y.H., Chiu, J.K. and Lim, E.P. (2005) 'Mining mobile group patterns: a trajectory-based approach', *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.713–718.

Kollios, G., Gunopulos, D. and Tsotras, V.J. (1999) 'On indexing mobile objects', *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp.261–272.

Kriegel, H-P. and Pfeifle, M. (2005) 'Clustering moving objects via medoid clusterings', *Proceedings of the 17th International Conference on Scientific and Statistical Database Management*, Lawrence Berkeley Laboratory, pp.153–162.

Kuijpers, B. and Vaisman, A.A. (2007) 'A data model for moving objects supporting aggregation', *Proceedings of the 23rd International Conference on Data Engineering*, IEEE Computer Society, pp.546–554.

Laube, P. and Imfeld, S. (2002) 'Analyzing relative motion within groups of trackable moving point objects', *Proceedings of the Second International Conference on Geographic Information Science*, Springer, pp.132–144.

Laube, P., van Kreveld, M. and Imfeld, S. (2004) 'Finding REMO – detecting relative motion patterns in geospatial lifelines', *Proceedings of the 11th International Symposium on Spatial Data Handling*, Springer, pp.201–214.

Leonardi, L., Marketos, G., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Raffaetà, A., Roncato, A., Silvestri, C. and Theodoridis, Y. (2010) 'T-warehouse: visual OLAP analysis on trajectory data', *2010 IEEE 26th International Conference on Data Engineering (ICDE)*, IEEE, pp.1141–1144.

Li, Y., Han, J. and Yang, J. (2004) 'Clustering moving objects', *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp.617–622.

Li, Z., Han, J., Ji, M., Tang, L-A., Yu, Y., Ding, B., Lee, J-G. and Kays, R. (2011) 'Movemine: mining moving object data for discovery of animal movement patterns', *ACM Trans. Intell. Syst. Technol.*, Vol. 2, No. 4, pp.37.1–37.32.

Li, Z., Ji, M., Lee, J-G., Tang, L-A., Yu, Y., Han, J. and Kays, R. (2010) 'Movemine: mining moving object databases', *SIGMOD*.

Liu, G., Lu, H., Yu, J.X. and Wei, W. (2003) 'AFOPT: an efficient implementation of pattern growth approach', *Proc. ICDM*.

López, I.F.V., Snodgrass, R.T. and Moon, B. (2005) 'Spatiotemporal aggregate computation: a survey', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 2, pp.271–286.

Ma, S. and Hellerstein, J.L. (2001) 'Mining partially periodic event patterns with unknown periods', *Proceedings of the 17th International Conference on Data Engineering*, IEEE Computer Society, pp.205–214.

Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y. and Cheung, D.W. (2004) 'Mining, indexing, and querying historical spatiotemporal data', *Proceedings of the 2004 International Conference on Knowledge Discovery and Data Mining*, ACM, pp.236–245.

Marchand, P., Brisebois, A., Bédard, Y. and Edwards, G. (2004) 'Implementation and evaluation of a hypercube-based method for spatiotemporal exploration and analysis', *ISPRS Journal of Photogrammetry & Remote Sensing*, Vol. 59, No. 1, p.620.

Mokbel, M., Ghanem, T. and Aref, W. (2003) 'Spatio temporal access methods', *IEEE Data Engineering Bulletin*, Vol. 26, pp.40–49.

Nanni, M. and Pedreschi, D. (2006) 'Time-focused clustering of trajectories of moving objects', *J. Intell. Inf. Syst.*, Vol. 27, No. 3, pp.267–289.

Papadias, D., Kalnis, P., Zhang, J. and Tao, Y. (2001) 'Efficient OLAP operations in spatial data warehouses', *SSTD'01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, Springer-Verlag, London, UK, pp.443–459.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.C. (2001) 'PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern', *IEEE Int. Conference on Data Engineering*.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M-C. (2004) 'Mining sequential patterns by pattern-growth: the PrefixSpan approach', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 11, pp.1424–1440.

Peng, W.C. and Chen, M.S. (2003) 'Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 1, pp.70–85.

Pfoser, D., Jensen, C.S. and Theodoridis, Y. (2000) 'Novel approaches in query processing for moving object trajectories', *Proceedings of the 26th International Conference on Very Large Data Bases*, pp.395–406.

Porkaew, K., Lazaridis, I. and Mehrotra, S. (2001) 'Querying mobile objects in spatio-temporal databases', *Proc. of 7th SSTD*, p.307.

Procopiuc, C.M., Agarwal, P.K. and Har-Peled, S. (2002) 'STAR-tree: an efficient self-adjusting index for moving objects', *Algorithm Engineering and Experiments: 4th International Workshop, ALENEX 2002*, 4–5 January 2002, revised papers, San Francisco, CA, USA.

Saltenis, S. and Jensen, C.S. (2002) 'Indexing of moving objects for location-based services', *Proceedings of the 18th International Conference on Data Engineering*, IEEE Computer Society, pp.463–472.

Ŝaltenis, S., Jensen, C.S., Leutenegger, S.T. and Lopez, M.A. (2000) 'Indexing the positions of continuously moving objects', *ACM SIGMOD Record*, Vol. 29, No. 2, pp.331–342.

Sclaroff, S., Kollios, G. and Betke, M. (2001) 'Motion mining: discovering spatio-temporal patterns in databases of human motion', *Proceedings of the 2001 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ACM.

Shim, C.B. and Chang, J.W. (2003) 'A new similar trajectory retrieval scheme using k-warping distance algorithm for moving objects', *Proceedings of the 4th International Conference on Advances in Web-Age Information Management*, Springer, pp.433–444.

Sinha, G. and Mark, D.M. (2005) 'Measuring similarity between geospatial lifelines in studies of environmental health', *Journal of Geographical Systems*, Vol. 7, No. 1, pp.115–136.

Sistla, A.P., Wolfson, O., Chamberlain, S. and Dao, S. (1997) 'Modeling and querying moving objects', *Proceedings of the 13th International Conference on Data Engineering*, IEEE Computer Society, pp.422–432.

Tao, Y., Papadias, D. and Sun, J. (2003) 'The TPR*-tree: an optimized spatio-temporal access method for predictive queries', *VLDB*, pp.790–801.

Theodoridis, Y. (2003) 'The R-tree-portal', available at http://www.rtreeportal.org/ (accessed on 30 July 2012).

Tsoukatos, I. and Gunopulos, D. (2001) 'Efficient mining of spatiotemporal patterns', *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, Springer, pp.425–442.

Vaisman, A. (2012) 'Business intelligence', in Aufaure, Marie-Aude and Zimányi, Esteban (Eds.): *Tutorial Lectures Series: Lecture Notes in Business Information Processing*, Chapter 'Data warehouses: next challenges', Vol. 96, pp.1–26, Springer Verlag, First European Summer School, eBISS 2011, Paris, France, 3–8 July 2011.

Vlachos, M., Kollios, G. and Gunopulos, D. (2002) 'Discovering similar multidimensional trajectories', *Proceedings of the 18th International Conference on Data Engineering*, IEEE Computer Society, pp.673–684.

Wang, Y., Lim, E-P. and Hwang, S-Y. (2003) 'On mining group patterns of mobile users', in Marík, V., Retschitzegger, W. and Stepánková, O. (Eds.): *DEXA*, Vol. 2736 of Lecture Notes in Computer Science, pp.287–296, Springer.

Wolfson, O., Xu, B., Chamberlain, S. and Jiang, L. (1998) 'Moving objects databases: issues and solutions', *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, IEEE Computer Society, pp.111–122.

Yan, X., Han, J. and Afshar, R. (2003) 'CloSpan: mining closed sequential patterns in large datasets', *Proceedings of the 3rd SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, pp.166–177.

Yang, J., Wang, W., Yu, P.S. and Han, J. (2002) 'Mining long sequential patterns in a noisy environment', *Proceedings of the 2002 International Conference on Management of Data*, ACM, pp.406–417.

Yazdani, N. and Ozsoyoglu, Z.M. (1996) 'Sequence matching of images', *Proceedings of the Eighth International Conference on Scientific and Statistical Database Management*, pp.53–62.

Yim, J., Joo, J. and Park, C. (2011) 'A Kalman filter updating method for the indoor moving object database', *Expert Systems with Applications*, Vol. 38, No. 12, pp.15075–15083.

Yip, K. and Zhao, F. (1996) 'Spatial aggregation: theory and applications', *Journal of Artificial Intelligence Research*, Vol. 5, pp.1–26.

Zaki, M.J. (2001) 'SPADE: an efficient algorithm for mining frequent sequences', *Machine Learning*, Vol. 42, No. 1, pp.31–60.

Zeinalipour-Yazti, D., Lin, S. and Gunopulos, D. (2006) 'Distributed spatio-temporal similarity search', *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, ACM, pp.14–23.

Zhang, R., Jagadish, H.V., Dai, B.T. and Ramamohanarao, K. (2010) 'Optimized algorithms for predictive range and KNN queries on moving objects', *Information Systems*, Vol. 35, No. 8, pp.911–932.