A Cloud-friendly RFID Trajectory Clustering Algorithm in Uncertain Environments

Yanbo Wu, Hong Shen, and Quan Z. Sheng *Member, IEEE*

Abstract—In the emerging environment of the Internet of Things (IoT), through the connection of billions of radio frequency identification (RFID) tags and sensors to the Internet, applications will generate an unprecedented number of transactions and amount of data that require novel approaches in mining useful information from RFID trajectories. RFID data usually contain a considerable degree of uncertainty caused by various factors such as hardware flaws, transmission faults and environment instability. In this paper, we propose an efficient clustering algorithm that is much less sensitive to noise and outliers than the existing methods. To better facilitate the emerging cloud computing resources, our algorithm is designed cloud-friendly so that it can be easily adopted in a cloud environment. The scalability and efficiency of the proposed algorithm are demonstrated through an extensive set of experimental studies.

Index Terms—Internet of Things, radio frequency identification (RFID), uncertainty, clustering algorithm, cloud computing.

1 Introduction

Identification technology is the bridge to connect the physical world with the digital one. Accurate and efficient identification is very important to realize automatic traceability. Radio Frequency Identification (RFID) has the capability of automatically extracting information from microelectronic tags attached to objects using radio waves. The identification is wireless and does not require the line of light. RFID was first explored in 1940s [13]. In the past decade, research initiatives by academic organizations, industrial interests from companies and government initiatives have rapidly escalated new development and interests in RFID technology. Alongside, Moore's law has ensured that integrated circuits reduce in size, cost and power consumption. Consequently, RFID systems have become more efficient and affordable.

Networked RFID is one of the important technological advances that help make RFID-enabled traceability possible [29]. The basic idea behind this technique is to connect otherwise isolated RFID systems and other software. RFID tags only carry an unambiguous ID, meanwhile other data pertaining to the objects, including the past and current states are stored and accessed over the Internet. Networked RFID brings significant and promising benefits to traceability applications. For example, it makes it possible for applications to automatically analyze recorded RFID events to discover the current or past information of an object, without physical access to it. Many organizations are planning or already exploiting RFID to enable traceability. Wal-Mart, the world's largest public corporation by revenue, in 2005, mandated its top 100 suppliers to tag their pallets and cases using RFID [2]. The U.S. Department of Defense released a policy on the use of RFID to its external suppliers and for internal operations in July of 2005 [28].

However, due to the cost and power constraints of RFID tags, reliability concerns arise under certain circumstances. In particular, RFID tags might not be read at all, leading to the mistaken belief that the object is not present (noted as "missing reading"). While some errors are of continuous nature, for example, tags which are attached to metal surface might never be detected, most of the missing readings only occur temporarily. For example, collisions on the air interface or interference from other radio sources. In distributed traceability applications that use RFID to judge whether a number of tagged objects are presented, the temporary missing readings result in unreliable operations. One of the consequences is that objects' trajectories become incomplete. This not only affects the tracking and tracing of individual objects (*Individual Traceability*), but also causes inaccurate statistics (*Statistical Traceability*), which leads to biased business decisions. We illustrate these two problems in the following two everyday life services.

Individual Traceability. One of the key functionalities in postal services is to allow end customers to track and trace their products. Currently, this is enabled by using barcode labels, which is time-consuming and labor-intensive because barcode has to be read by an operator using a scanner. RFID has the ability to eliminate this operation. However, due to the possibility of missing readings, it may generate confusing information. For example, a package may seem to have "jumped" from one country to another without clearing customs. Improving the accuracy of RFID-based traceability can boost its adoption in this kind of applications and save human resources.

- *Yanbo Wu is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044, China. E-mail: ybwu@bjtu.edu.cn*
- *Hong Shen (corresponding author) is with the School of Information Science and Technology, Sun Yat-sen University, China, and School of Computer Science, University of Adelaide, Australia. E-mail: hong@cs.adelaide.edu.au*
- *Quan Z. Sheng is with School of Computer Science, University of Adelaide, Australia. E-mail: qsheng@cs.adelaide.edu.au*

Statistical Traceability. In supply chain management systems, distribution route planning is very important for the cost control. Historical shipping information can be used to adjust the routes. However, missing readings seriously affect its accuracy. For example, assuming the average missing reading rate is 5%, in a path where there are 10 nodes, 45% of trajectories generated by the RFID system are incomplete in the worst case (where different objects are missed at different nodes). Moreover, 2^{10} different routes may be generated in the worst case. Consequently, it becomes very difficult for trajectory-based mining and analysis.

One approach to tackle these problems would be to design more sophisticated tags that are not as vulnerable to shielding or interference. However, for RFID technology to be ubiquitous, they must be affordable and able to be used in hostile environments. Given these constraints, it is more feasible to deal with uncertainties of RFID data in the software layer [39]. There have been many data cleaning and probabilistic event extraction methods ([9], [8], [26], [11]) as middleware to improve the quality of RFID data. However, most of them do not take the RFID trajectory as the target, especially in a distributed application that may be deployed in a wide area.

In this paper, we propose a software approach that handles uncertainties in distributed RFID-enabled traceability applications. We formalize the problems as trajectory clustering and propose an efficient clustering algorithm with a novel similarity measurement model. In a nutshell, the major contributions of our work are as the following:

- We propose a novel similarity measure model which is designed for RFID trajectories. This model can deal with variants in both time and space dimensions.
- We propose an efficient clustering algorithm which is much less sensitive to noise and outliers than existing methods. We further parallelize the clustering algorithm using MapReduce paradigm to make better use of the cloud computing resources.
- We evaluate our approach with extensive experiments and the results demonstrate the efficiency and scalability of the proposed approach.

The rest of the paper is organized as follows. In Section 2, we give a brief review of the state-of-the-art research on uncertainties in RFID-based systems. In Section 3, we formalize the trajectory recovery problem. Then we introduce the similarity measurement models for RFID trajectories in Section 4. The clustering and merging algorithms are presented in Section 5. In this section, we also further present the parallelization of the clustering algorithm based on MapReduce. We report the experimental evaluation results in Section 6. Finally, we offer some concluding remarks in Section 7.

2 Related Work

In this section, we overview major techniques that are most closely related to our work discussed in this paper.

2.1 RFID Data Cleaning

To compensate the unreliability of RFID streams, most of the RFID stream cleaning middleware employ the "Smoothing Filter" technique. *SMURF* [9] is a first declarative, adaptive smoothing filter for RFID data cleaning. SMURF models the unreliability of RFID readings by viewing RFID streams as a statistical sample of tags in the physical world. SMURF continuously adapts the size of the smoothing window size to provide accurate RFID data. However, it does not propose the optimal smoothing filter for the readings of single tag and an aggregate signal. In [8], the authors propose an abstracted adaptive RFID framework called *MDI-SMURF* which cleans the RFID data while shields applications from the challenges that arise when interacting directly with sensor devices. However, this work does not consider the impact of energy consumption of sensor devices.

The work in [26] introduces a deferred approach for detecting and correcting RFID data anomalies. Unlike *SMURF* which cleans data according to its own rules, this work allows applications to define detection and correction rules and rewrites the queries based on these rules on the data. However, it requires more overhead for the cleaning task. In [21], a new adaptive data cleaning scheme called *WSTD* is proposed. WSTD compares two window subrange observations or estimated tag counts to detect when transitions occur within a window. It does not consider how to deal with duplicates and false positive readings.

In [41], the authors introduce a data cleaning approach that exploits basic characteristics of RF signals and maximum likelihood operations. This method enables reasoning about the position of RFID tags in the reader's range without measuring the signal strength of tag responses. However, the scheme considers simplified properties of RF signals, rather than general properties arising in applications. The work in [19] deals with another problem when data streams collected by multiple readers usually contain cross-reads[1], by introducing a method that estimates the density of each tag using a kernel-based function and keeps the event from the reader corresponding to the micro-cluster with the largest density. However, this scheme assumes a fixed window size, which is not applicable for most applications. In [12], a Bayesian inference-based framework for cleaning RFID raw data is proposed. The authors first study an n-state detection model and formally prove that 3-state model can maximize the system performance. Then they extend the model to support 2-dimensional RFID reader arrays, which is not suitable to support RFID readers in a 3-dimensional space.

There are also some generic data cleaning frameworks. The work [5] proposes a cleaning framework that takes an RFID data set and a collection of cleaning methods, with associated costs, and a cleaning plan that optimizes the overall cleaning costs by determining the conditions under which inexpensive methods are appropriate and necessary.

1. In a small environment, a tag may be read by more than one readers at the same time, as a result, its location is uncertain because of inconsistency derived from the readers.

Although the proposed scheme can achieve reduced cost, its accuracy is not discussed. The work [7] presents the Extensible Sensor Stream Processing (*ESP*) framework to build sensor data cleaning infrastructures. ESP is designed as a pipeline using declarative cleaning methods, which does not consider the energy consumption of sensor devices.

These cleaning methods for RFID streams mostly focus on improving the data quality in the statistical perspective, i.e., they are specifically designed for aggregated queries such as COUNT, AVG and SUM. Moreover, these methods are mainly concerned with the aggregated information of individual objects. As the result, they are not suitable for dealing with cross-site trajectory queries in distributed traceability applications.

2.2 Probabilistic RFID Event Processing

In the uncertain environment, extracting and processing events from RFID data streams poses various challenges. Consequently, regular event processing techniques are no longer effective. In [11], a probabilistic approach called *PEEX* is presented for high-level event extraction from RFID data. PEEX translates event definitions into SQL queries and replies on confidence tables to determine the probability of ambiguous events. It uses partial events to handle data errors. PEEX just adopts the existing probabilistic data models and shows no improvement in effectiveness. In [27], an event processing system named *Lahar* for probabilistic event streams is proposed. Lahar exploits the probabilistic nature of the data to enable declarative queries over real time and archived streams of probabilistic events. Lahar proposes algorithms for each class of queries including regular queries, extended regular queries, safe queries and unsafe queries. Unfortunately, it does not provide a comprehensive analysis on the performance of these algorithms. *Cascadia* [38] is another RFID event processing system which can cope with ambiguous RFID data by transforming RFID readings into probabilistic events. However, the event detection performance of the scheme needs to be improved. In [23], the authors propose an approach to perform complex event processing directly over unreliable RFID event streams by incorporating cleaning requirements into complex event specifications. The approach, however, suffers a higher cost.

In [36] and [37], the authors introduce an approach for event materialization under uncertainty, which includes a model for representing materialized events using *Bayesian Network*, and the algorithms for specifying the probability space of an event history where *Monte Carlo* sampling algorithm is used to assess materialized event probabilities. This approach does not discuss other factors, such as the priority of events. In [31], the authors propose a probabilistic model to capture the mobility of RFID readers, object dynamics and noisy readings. This model can self-calibrate by automatically estimating key parameters from the observed data. Based on the model, the authors also propose a sampling-based technique to infer clean, precise information about object locations from raw streams from mobile RFID readers. However, it does not discuss query processing over inferred data for various monitoring applications.

These models, algorithms and frameworks are designed mainly for the extraction of high-level events from unreliable RFID data and derivation of their confidences. They focus on each individual RFID-tagged object and location, rather than the trajectories of their movements.

2.3 Uncertain RFID Stream Processing

RFID data generated by distributed-deployed readers are often treated as streams. As the result, generic uncertain data stream processing techniques can be applied. In fact, some of these techniques are actually motivated by the research on RFID data streams.

In [33], a system called *PODS* is proposed which supports stream processing for uncertain data captured using continuous random variables. It also includes a data model that is flexible and allows efficient computation. The authors develop evaluation techniques for complex relational operations based on the model. However, their work on ranking in probabilistic databases give simplistic solutions to handling continuous distributions. The authors further propose an evaluation framework [32] that includes a general data model, approximation metrics and approximate representations for uncertain data streams. With this framework, both deterministic and randomized data-stream algorithms are designed to answer complex conditioning queries with approximate distributions with bounded errors. However, this work mainly focuses on the discrete probabilistic attributes. In [30], the *CLARO* system is presented, which extends the work in [32] by considering query planning for complex queries given an accuracy requirement.

In [18], the similarity join processing problem on uncertain data streams is studied. The authors formalize this problem and propose effective pruning methods on both object and sample levels to filter false alarms. Only event queries including regular queries are handled. In [16], the authors model the noisy sensor data, including RFID data, as *Morkovian Streams* and develop the access methods on a Markovian stream event query processing system in *Lahar* [27]. This work ignores the correlations that can result in highly inaccurate results to the queries, such as aggregate queries, and what-if queries.

Pattern matching over uncertain event streams is another important topic. In [1], the authors propose a formal evaluation model that offers precise semantics for the new class of queries and a query evaluation framework permitting optimizations in a principled way. Though this work can obtain good performance through a variety of optimizations, it imposes a fixed evaluation order determined by the state transition diagram, and hence is hard to handle concurrent events. The work in [40] proposes a model to measure pattern frequentness based on the *possible world semantics*. Two mining algorithms are developed based on the model for mining frequent sequential patterns. This work focuses on approximate string matching over sequence data, which may not be applicable for subsequence matching in the general case due to the existence of essential differences between the semantics of subsequence and string matches. In [20], two solutions for dealing with out-of-order event streams are proposed, namely *aggressive* and *conservative* strategies. The aggressive strategy produces maximal output under the optimistic assumption that out-of-order event arrival is rare, while the conservative one is to deal with the common out-of-order data. In [35], the authors model the mining of frequent itemsets as a Poisson binomial distribution and develop an approximate algorithm which can efficiently and accurately discover frequent itemsets in a large uncertain database. The proposed approach is useful for a probabilistic database with Boolean attributes, but not applicable for probabilistic, quantitative data.

2.4 Trajectory Clustering

Trajectory clustering is an important topic in data management and mining, which has a wide range of applications in various areas, such as traffic monitoring, video surveillance, cattle tracking and supply chain management.

The authors of [3] propose a clustering algorithm based on probabilistic modeling of a set of trajectories as individual sequences of points generated from a finite mixture model consisting of regression model components. The proposed method is a model-based approach that is not scalable. Moreover, it implicitly assumes that the trajectories follow the basic models, which is not applicable for many real datasets. Unsupervised learning is carried out using maximum likelihood principles. In [14], a feature generation framework called *TraClass* for trajectory data is proposed. TraClass generates a hierarchy of features by partitioning trajectories and exploring two types of clustering: (1) region-based and (2) trajectory-based. The work in [15] presents a partition-and-group framework for clustering trajectories, which partitions a trajectory into a set of line segments and then groups similar line segments together into a cluster. A problem with this approach is that not all potential stops can be found during the clustering process.

The authors in [17] present an effective trajectory clustering framework in which a coarse-to-fine strategy is taken. A feature called *trajectory directional histogram* is proposed to describe the statistic directional distribution of a trajectory in the feature extraction stage. Both coarse clustering and fine clustering are based on a graph-theoretic clustering algorithm called *dominant-set clustering*. The resulting smoothed track can only be interpolated and sampled to a fixed size. In [25], the authors propose an on-line trajectory clustering algorithm. Trajectories are clustered on-line as the data are collected, and clusters are organized in a tree-like structure that, augmented with probability information, can be used to perform behavior analysis. The quality of the clusters built by the method decreases as the number of real-time requirements increases and the object resolution decreases.

The work [22] present an adaptation of a density-based clustering algorithm to trajectory data based on a simple notion of distance between trajectories. A new approach to the trajectory clustering problem, called *temporal focusing*, is sketched, to improve the quality of trajectory clustering by exploiting the intrinsic semantics of the temporal dimension. However, spatio-temporal co-presence does not explicitly include the interactions within individuals. Relations such as "leading", "following", or "setting a trend" cannot be investigated by clustering alone. In [34], the authors formalize non-metric similarity functions based on the *Longest Common Subsequence* (LCSS), which are very robust to noise and provide an intuitive notion of similarity between trajectories by giving more weight to the similar portions of the sequences. However, gaps between similar subsequences are not considered, which may lead to inaccuracy. The work [10] formalizes the concept of a convoy query using density-based notions. Three efficient algorithms are developed for convoy discovery that adopt the well-known filter-refinement framework. The proposed work focuses on raw trajectories, therefore missing the related semantic information contained in the background geographic and application databases.

3 Problem Definition

In this section, we first formally define basic concepts in typical RFID-enabled traceability systems and then specify the research problems targeted in this paper.

Definition 1 (*Traceability Network*). A traceability network is a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. $\mathbf{V}$ represents the set of nodes where RFID readers are deployed and $\mathbf{E}$ represents the set of *possible* connections between nodes. A node v_i is represented by its unique identifier, and a connection is represented by (v_s, v_e) where v_s and v_e are two nodes. ∎

It should be noted that a node refers to a location where more than one readers might be installed. Unlike other RFID systems where each reader is treated as a location, we aggregate the readers at the same location as one node. This is a reasonable abstraction in distributed RFID systems.

Definition 2 (*Trajectory*). A trajectory of a given RFID-tagged object o_i is a polyline in a three dimensional space $(\mathbf{V}, \mathbf{T}_s, \mathbf{T}_e)$, where $\mathbf{T}_s$ is the time space for arrival readings and $\mathbf{T}_e$ is the time space for leaving readings. A trajectory TR_i of o_i is represented as a sequence of points, accompanied by a unique ID of the object: $TR_i = \{o_i, \{(v_1, t_{s_1}, t_{e_1}), (v_2, t_{s_2}, t_{e_2}), \ldots, (v_n, t_{s_n}, t_{e_n})\}\}$. Its $\mathbf{V}$-axis values, ordered by $\mathbf{T}_s$ values, form a path P in $\mathbf{G}$. The set of all trajectories is denoted as $\mathbf{ST}$. ∎

The deployment of RFID readers may affect the timestamps of arrival/leaving readings:

- If readers are deployed at the entrance and exit of a node, t_s and t_e (captured by entrance reader and exit reader respectively) are different, i.e., $t_s < t_e$.
- If only one reader is deployed at a node and only one reading of each object is captured, $t_s = t_e$.
- If only one reader is deployed at each node, but the first and last readings of each object are captured, t_s and t_e (captured by the same reader) may be different, i.e., $t_s \leq t_e$.

Figure 1 shows examples of RFID trajectories. When objects move together, e.g., o_2 and o_3 in Figure 1, the

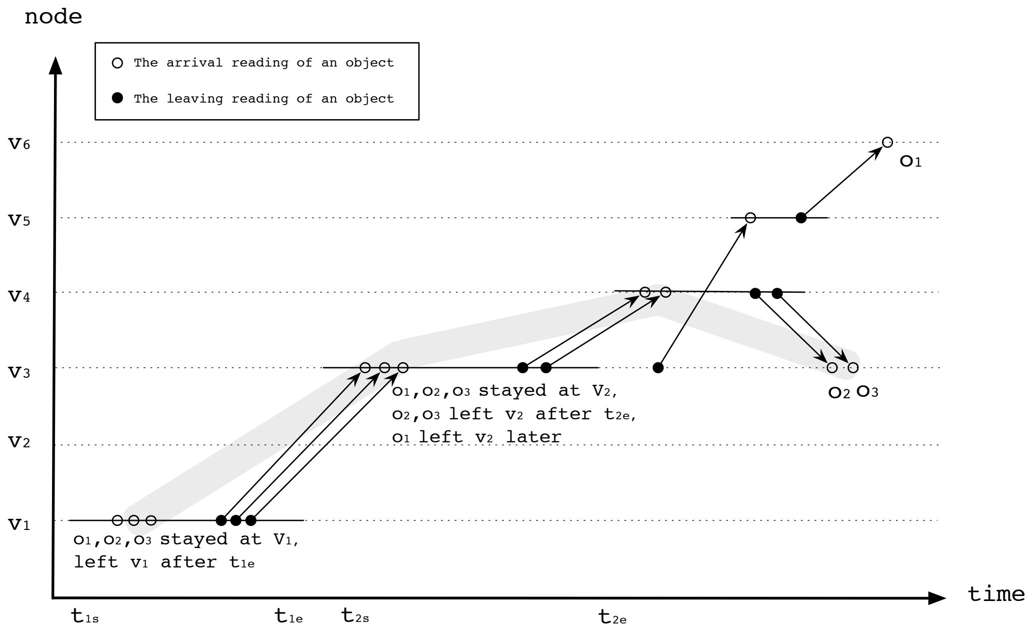


Fig. 1: Examples of RFID Trajectories

timestamps when they are captured at the same node fall into a certain range $[t_s, t_e]$. With this, we define the concept of *Trajectory Cluster* as the following:

Definition 3 (*Trajectory Cluster*). A trajectory cluster TC is a sequence of node-range pairs, which describes the common temporal-spatio relationship of a group of objects. Formally, $TC = \{(v_1, (t_{s_1}, t_{e_1})), (v_2, (t_{s_2}, t_{e_2})), \ldots, (v_n, (t_{s_n}, t_{e_n}))\}$. ▮

Now we define the research problems of managing uncertainties for traceability as follows.

- *Outlier Detection.* Suppose TR is the real trajectory of an object and TR' is the one captured by the readers, the first task is to determine whether $TR' = TR$, i.e., to determine whether there are missing readings in the process of capturing the object. Evidently, there is no deterministic way to do so in the software layer. We define the probability of $TR' \neq TR$ as $p_{outlier}(TR')$. The reason why we classify this problem as *outlier detection* is that when objects move together and are correctly captured, these objects should follow the same path P during the same time range. When TR' misses at least one segment of the trajectory, it can be treated as an outlier. TR' is an outlier if $p_{outlier}(TR') \geq \epsilon_{outlier}$, where $\epsilon_{outlier}$ is a predefined threshold.
- *Classification.* If TR' is detected as an outlier (missing readings exist), the next task is to recover the missing readings. Similar to the outlier detection, we can assume that most objects' trajectories are captured correctly. Suppose we have the correct and complete trajectory clusters $\mathbf{SC} = \{TC_1, TC_2, \ldots, TC_n\}$, the recovery task can be transformed to a classification problem.
- *Clustering.* In most cases, the set of trajectory clusters $\mathbf{SC} = \{TC_1, TC_2, \ldots, TC_n\}$ is not known beforehand. Moreover, it may change occasionally. As the result, in order for the classification to work, it is necessary to generate $\mathbf{SC}$ by clustering the existing trajectories.

If we can cluster the trajectories into different classes, it is possible to detect the outliers and recover the missing readings. Clearly, clustering is the key issue. A number of clustering algorithms have been reported in the literature. The representative algorithms include k-means, BIRCH, DBSCAN, OPTICS, and STING. The models and algorithms discussed in Section 2.4 are designed to cluster trajectory data. However, in RFID-enabled traceability networks, especially those *distribution* systems such as postal services and supply chains, clustering trajectories as a whole could not detect similar portions of the trajectories. We note that a trajectory may have a long and complicated path. Hence, even though some portions of the trajectory show a common behavior, the whole trajectory might not. For example, in a supply chain network, when the products are first shipped from the manufacturer's facility to warehouses and distribution centers, objects all move together. Later, when they are distributed to supermarkets and eventually end customers, their trajectories no longer share the common patterns. If we cluster the trajectories as a whole, it will be too sparse. As a result, the outlier detection will not be effective because an outlier will most likely to be treated as a different cluster. For example, in Figure 1, objects o_1, o_2, o_3 moved in groups from node v_1 to v_3. Their sub-trajectories from v_1 to v_3 can be clustered in the same group, though the full trajectories fall into different clusters.

Name	Definition	Limitations
Eu	$\sqrt{\sum_{i=1}^{n} dist(TR_1(i), TR_2(i))}$	Sensitive to noise. TR_1 and TR_2 must have the same length.
DTW	$\begin{cases} 0 & \text{if } \lvert TR_1\rvert = \lvert TR_2\rvert = 0 \\ \infty & \text{if } \lvert TR_1\rvert = 0, or \lvert TR_2\rvert = 0 \\ dist(TR_1(1), TR_2(1)) + min\{ & \text{otherwise} \\ DTW(Rest(TR_1), Rest(TR_2)), & \\ DTW(Rest(TR_1), TR_2), & \\ DTW(TR_1, Rest(TR_2))\} & \end{cases}$	Sensitive to noise. Cannot handle time range.
ERP	$\begin{cases} \sum_1^{\lvert TR_1\rvert} dist(TR_1(i), g) & \text{if } \lvert TR_2\rvert = 0 \\ \sum_1^{\lvert TR_2\rvert} dist(TR_2(i), g) & \text{if } \lvert TR_1\rvert = 0 \\ min\{ & \text{otherwise} \\ ERP(Rest(TR_1), Rest(TR_2)) + dist(TR_1(1), TR_2(1)) & \\ ERP(Rest(TR_1), TR_2) + dist(TR_1(1), g), & \\ ERP(TR_1, Rest(TR_2)) + dist(TR_2(1), g)\} & \end{cases}$	Sensitive to noise. Cannot handle time range.
LCSS	$\begin{cases} 0 & \text{if } \lvert TR_1\rvert = 0, or \lvert TR_2\rvert = 0 \\ LCSS(Rest(TR_1), Rest(TR_2)) + 1 & \text{if } TR_1(1).v = TR_2(1).\text{v} \\ max\{ & \text{otherwise} \\ LCSS(Rest(TR_1), TR_2)), & \\ LCSS(TR_1, Rest(TR_2))\} & \end{cases}$	Cannot handle time range.
EDR	$\begin{cases} \lvert TR_1\rvert & \text{if } \lvert TR_2\rvert = 0 \\ \lvert TR_2\rvert & \text{if } \lvert TR_1\rvert = 0 \\ min\{ & \text{otherwise} \\ EDR(Rest(TR_1), Rest(TR_2)) + subcost, & \\ EDR(Rest(TR_1), TR_2) + 1, \} & \\ EDR(TR_1, Rest(TR_2)) + 1\} & \end{cases}$	Cannot handle time range.

TABLE 1: Similarity Measurement Models

4 SIMILARITY OF TRAJECTORIES

The challenge of measuring the similarity of two RFID trajectories lies on the variation of both time and space dimensions. As a result, we cannot simply adopt the models such as Euclidean Distance (Eu), Dynamic Time Warping (DTW), Edit Distance with Real Penalty (ERP), Longest Common Subsequence (LCSS) and Edit Distance on Real Sequences (EDR). In this section, we will first briefly review these similarity measurement models and then propose our model.

All the models we mentioned assume that the space dimension is a two (or higher) dimensional vector such as $v = (x, y)$, thus Euclidean Distance for points are used to determine the distance between two locations, i.e., $dist(v_1, v_2) = \sqrt{(v_1.x - v_2.x)^2 + (v_1.y - v_2.y)^2}$. We cannot simply degrade this function to one dimension by eliminating its y axis, because the locations in our model are represented by IDs. In our model, Euclidean Distance of two locations will degrade to: $dist(v_1, v_2) = \begin{cases} 1 & \text{if } v_1 = v_2 \\ 0 & \text{if } v_1 \neq v_2 \end{cases}$. For the simplicity of discussion, we denote the i_{th} element of a trajectory TR by $TR(i)$ and the length of TR by $|TR|$. Given two trajectories TR_1 and TR_2, their distance is defined by each of these models as shown in Table 1.

DTW and ERP both suffer from the noise in the data. To illustrate this, suppose we have four trajectories, namely[2]:

- TR_1={(1, 1, 2),(2, 3, 4),(3, 5, 6),(4, 7, 8)},
- TR_2={(1, 1, 2),(3, 5, 6),(4, 7, 8)},
- TR_3={(1, 1, 2),(2, 3, 4),(4, 6, 8),(3, 9, 10)}, and
- TR_4={(1, 1, 2),(3, 3, 6),(4, 7, 8),(2, 9, 10)}.

2. To be able to adopt these models, we ignore the time dimension values and use the order as the time dimension.

If TR_1 is the query trajectory, and TR_2 is the same trajectory but with readings at node v_2 missing. DTW and ERP will produce the same order: TR_3, TR_2, TR_4, while the correct ranking should be: TR_2, TR_3, TR_4 (notice that the time ranges of the trajectories are different). LCSS and EDR do not consider the variant of time dimension either.

In this paper, we propose a similarity measurement model called *Time-parameterized Edit Distance* (TED) (see Definition 4), where the time dimension values are also used in the calculation. Similar to the other models, we normalize the time dimension values. The normalization is done against the time range in consideration, i.e., if we want to cluster the trajectories from $\mathbf{T}_1 \to \mathbf{T}_2$, the t_s and t_e values will be normalized as $normalized(t_s) = \frac{t_s - \mathbf{T}_1}{\mathbf{T}_2 - \mathbf{T}_1}$ and $normalized(t_e) = \frac{t_e - \mathbf{T}_1}{\mathbf{T}_2 - \mathbf{T}_1}$. For the sake of simplicity, we will use t_s and t_e to represent their un-normalized counterparts in the remaining discussions.

In TED's definition, we introduce a new function called *Time-parameterized Distance* for two elements e_1 and e_2 in two trajectories, namely $dist_t(e_1, e_2)$, which is defined in Definition 5. TED is derived from EDR by replacing the $subcost$ with a time-parameterized cost. It will yield high precision for trajectory mining tasks, whereas its recall will not be as good as EDR because it imposes a heavier penalty. To cope with different requirements of different mining tasks, we also propose a recall-oriented similarity measurement model, called *Time-parameterized Longest Common Subsequences* ($TLCSS$), which is defined in Definition 6.

Similarity Function	TR_2	TR_3	TR_4
TED	1	1.28	3
$TLCSS$	3	2.9	2.8

TABLE 2: Examples of TED and TLCSS

Definition 4 (*Time-parameterized Edit Distance*).

$$TED(TR_1, TR_2) = \begin{cases} |TR_1| & \text{if } |TR_2| = 0 \\ |TR_2| & \text{if } |TR_1| = 0 \\ min\{dist_t(TR_1(1), TR_2(1)) + TED(Rest(TR_1), Rest(TR_2)), & \text{otherwise} \\ TED(Rest(TR_1), TR_2) + 1, TED(TR_1, Rest(TR_2)) + 1\} & \end{cases}$$

Definition 5 (*Time-parameterized Distance*).

$$dist_t(e_1, e_2) = \begin{cases} \sqrt{(e_1.t_e - e_2.t_e)^2 + (e_1.t_s - e_2.t_s)^2} & \text{if } e_1.v = e_2.v \\ \sqrt{(e_1.t_e - e_2.t_e)^2 + (e_1.t_s - e_2.t_s)^2} + 1 & \text{if } e_1.v \neq e_2.v \end{cases}$$

Definition 6 (*Time-parameterized Longest Common Subsequence*).

$$TLCSS(TR_1, TR_2) = \begin{cases} 0 & \text{if } |TR_1| = 0, or |TR_2| = 0 \\ LCSS(Rest(TR_1), Rest(TR_2)) + 1 - dist_t(TR_1(1), TR_2(1)) & \text{if } TR_1(1).v = TR_2(1).\text{v} \\ max\{LCSS(Rest(TR_1), TR_2)), LCSS(TR_1, Rest(TR_2))\} & \text{otherwise} \end{cases}$$

Table 2 shows the similarity of TR_1 against aforementioned TR_2, TR_3 and TR_4, which is calculated using the two methods. Both methods yield the correct ranking. It should be noted that with $TLCSS$, the larger the value is, the more similar the trajectories are, which is the opposite of edit-distance-based models.

5 Trajectory Clustering

In this section, we present our algorithms in trajectory clustering. We also further discuss an algorithm that enables parallel trajectory clustering based on MapReduce.

5.1 Hierarchical RFID Trajectory Clustering

Our clustering algorithm is a hierarchical one. The basic idea is to first cluster the points projected on the ($\mathbf{T}_s$, $\mathbf{T}_e$) plane. Then for each cluster in this plane, expand it to the third dimension - the location. During the expansion, the clusters are divided into sub-clusters which represent the "branches" at that node. The algorithm is formally defined in Algorithm 1 (see Figure 2).

```
Algorithm 1: Hierarchical RFID Trajectory Clustering
Input: The set of trajectories: TR = $\{TR_1, TR_2, \dots, TR_n\}$.
Output: The set of trajectory clusters: TC.
        The set of outliers: OL.
 1: $\mathbf{TC} = \Phi$, $\mathbf{OL} = \Phi$
 2: $hrtc(\mathbf{TC}, \mathbf{OL}, 1)$
 3:
 4: function $hrtc(clusters, outliers, depth)$
 5:     for each cluster $TC_{p_i}$ in $clusters$ with $|p_i| = depth$
 6:         for each trajectory $TR_i$ belongs to $TC_{p_i}$
 7:             if $|TR_i| > depth$
 8:                 $p'_i = p_i + TR_i(depth)$
 9:                 if $TC_{p'_i}$ exists, assign $TR_i$ to $TC_{p'_i}$
10:                 otherwise add $TC_{p'_i}$ to $clusters$
11:             end if
12:         end for
13:         for each newly added $TC_{p'_i}$
14:             replace $TC_{p'_i}$ with $OPTICS_t(TC_{p'_i})$
15:             $outliers += outlier(OPTICS_t(TC_{p'_i}))$
16:         end for
17:         remove $TC_p$ from $clusters$
18:     end for
19:     if $depth \leq MAX\_DEPTH$
20:         $hrtc(clusters, outliers, depth + 1)$
21: end function
```

Fig. 2: Hierarchical RFID Trajectory Clustering

In Algorithm 1, a cluster TC_{p_i} has two characteristics, namely the path p and the sub-cluster ID i. Path p highlights this cluster with the common moving path of the trajectories, which is the projection of the trajectories on the $\mathbf{V}$ dimension, while the sub-cluster ID i highlights the difference of the trajectories along the same path p, by clustering them with the values of the $depth_{th}$ elements in $\mathbf{T}_s$ and $\mathbf{T}_e$ dimensions.

The algorithm works as the following. Firstly, for all existing clusters, we split the trajectories belonging to each of them to different sets, according to their next stops (line 6 - 13). Then, for each set, the $OPTICS_t$ clustering routine is performed on the time dimensions of the next stop (line 13 - 16). This process continues recursively until the length of the path reaches the MAX_DEPTH constant (line 19). This algorithm actually generates a forest of clusters incrementally as illustrated in Figure 3. A demonstration of the process is shown in Figure 4.

OPTICS is a typical density-based clustering algorithm. The basic idea of OPTICS is that for each object in a cluster, the neighborhood of a given radius ϵ has to contain at least a minimum number (n_{pts}) of objects, i.e., the cardinality of the neighborhood has to exceed a given threshold. Therefore, in the case that noise emerging to an object results in a missing reading (outlier), it is guaranteed that the missing reading can be recovered through merging the readings of some ϵ-neighbors of the object. For this reason, OPTICS is robust to noise and outliers. In our work, we use it as the clustering algorithm to group the time dimension values for trajectory elements. In order for this to work, we treat (t_s, t_e) as a point in a two dimensional plane. As a result, we denote the function as $OPTICS_t$. MAX_DEPTH is an application-specific constant. In a quasi-static traceable network, it is easy to determine the value of the constant to gain the best clustering result.

5.2 Merging Clusters

The algorithm in Section 5.1 handles the outliers in time dimensions, but does not handle the missing readings. Generally, when missing readings happen, the trajectories

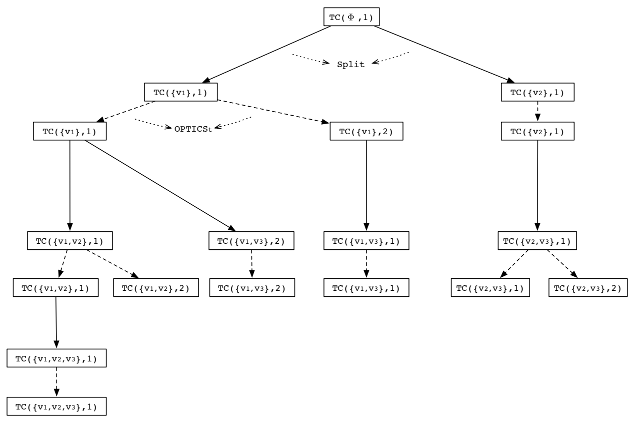


Fig. 3: Examples of Hierarchical RFID Trajectory Clustering

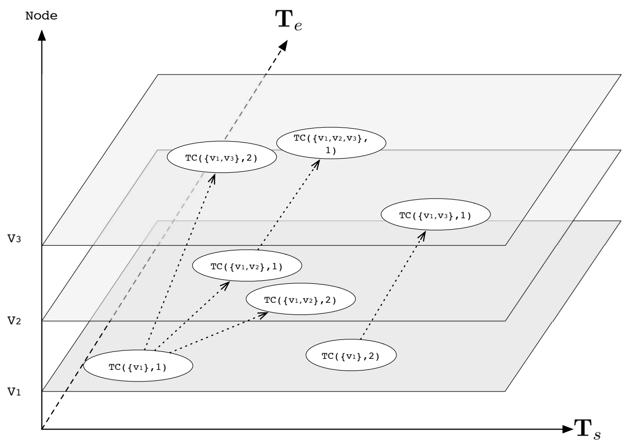


Fig. 4: Demonstration of Hierarchical RFID Trajectory Clustering

will have a missing node. For example, a cluster with path $p_1 = \{v_1, v_2, v_3, v_4\}$ will become $p'_1 = \{v_1, v_3, v_4\}$ when objects are not read at v_2. Using the hierarchical clustering algorithm, this path will be treated as a different path than p_1, instead of an outlier. However, noticing that p'_1 is a sub-path of p_1, we can merge these two clusters to recover the missing readings.

Merging operation is simple: just merging two sets of trajectories into one set. However, the key question is *when* to do the merging. To determine this, we develop an efficient approach. Given two clusters TC_1 and TC_2, the first condition is that the difference of their node sets must be 1, i.e., there is only one different node in their paths. It is worth noting that the orders of nodes in the paths must be the same. For example, $\{v_1, v_2, v_3, v_4\}$ and $\{v_1, v_3, v4\}$ might be able to be merged, but $\{v_1, v_2, v_3, v_4\}$ and $\{v_1, v_4, v_3\}$ might not. Obviously, the time complexity of this merge is $O(1)$.

We also need to consider the time dimension in merging. In this paper, we use the similarity function *TED* introduced in Section 4 to determine whether two clusters are similar enough to be merged. However, since clusters might contain many trajectories, calculating average similarity for trajectories in TC_1 and TC_2 will take $O(|TC_1| * |TC_2|)$ in calculation time. This is inefficient in large-scale applications. As a result, we define *Representative Trajectory Similarity (RTS)* of two clusters which reduces the running time complexity of merging detection to $O(|TC_1|+|TC_2|)$. RTS is the similarity of the representative trajectories of two clusters. First, we need to generate the representative trajectory of a cluster. Given the fact that a cluster might contain trajectories of different lengths because of previous merging, we choose the longest path as the node dimension representation. In order to generate the time dimension representations, we calculate the average t_s/t_e for all trajectories at each node. If a trajectory does not contain the values for a node, the average value of all the other trajectories will be used. That is, we simply ignore this trajectory when calculating the average time representations at that node. This method is formally defined in Definition 7. It is easy to prove that the calculation of RT is of complexity $O(|TC|)$. As the result, calculating RTS will take $O(|TC_1| + |TC_2|)$ time.

With RTS, we propose the cluster merging algorithm in Algotithm 2 (see Figure 5). Firstly, the clusters are sorted by the length of the paths (line 1), so that the merging is an one-path operation. Then starting from the cluster with the shortest path, we find all the candidate clusters (line 2-3) and choose the one with the minimum TED distance and the distance is lower than a given threshold ϵ to merge into (line 4 - 6). Note that the calculation of RT can be time consuming, we cache that the value for each cluster and re-calculate it (line 6) only when the merging happens. The recalculation does not need to scan all the trajectories. As the result, we only need one pass of all trajectories to the merging, so the merging cost is $O(|\mathbf{TR}|)$.

```
Algorithm 2: Trajectory Cluster Merging
Input: The set of trajectory clusters: $\mathbf{TC} = \{TC_1, TC_2, \ldots, TC_n\}$.
Output: The set of merged trajectory clusters (in place merging): $\mathbf{TC}$.
1: Sort $\mathbf{TC}$ by the lengths of the paths
2: for each $TC$ in sorted $\mathbf{TC}$
3:     find the set of $candidates$ to merge to: $\{TC_1, TC_2, \ldots, TC_m\}$
       where $|TC_i| - |TC| = 1$ and the difference of nodes is 1
4:     for each $TC'$ in $candidates$
5:         if $TED(RT(TC), RT(TC'))$ is the minimum and $TED(RT(TC), RT(TC')) < \epsilon$
6:             merge $TC$ into $TC'$, re-calculate the RT for $TC'$
7:             break
8:     end for
9: end for
```

Fig. 5: Trajectory Cluster Merging

5.3 Nearest Neighbor Classification

For outliers detected in Section 5.1, we are able to classify them to one of the clusters. There are two approaches to do so. On the one hand, we can calculate the similarity between the representative trajectory of each cluster and the outlier to find the nearest cluster. On the other hand, we can calculate the average of the similarity between the trajectory in a cluster and the outlier as the similarity between the cluster and the outlier. Both approaches have

Definition 7 (*Representative Trajectory (RT)*).

$RT(TC) = \{(\frac{\sum_{i=1}^{|TC|} e_1.t_{si}}{|TC|}, \frac{\sum_{i=1}^{|TC|} e_1.t_{ei}}{|TC|}), (\frac{\sum_{i=1}^{|TC|} e_2.t_{si}}{|TC|}, \frac{\sum_{i=1}^{|TC|} e_2.t_{ei}}{|TC|}), \ldots, (\frac{\sum_{i=1}^{|TC|} e_m.t_{si}}{|TC|}, \frac{\sum_{i=1}^{|TC|} e_m.t_{ei}}{|TC|})\}$, where m is the length of the longest trajectory in TC. ∎

```
Algorithm 3: Map-Reduce Based Trajectory Clustering
1: function Map(Trajectory Set $\mathbf{TS}$)
2:     split $\mathbf{TS}$ to a set of sub trajectory sets: $\mathbf{TS}_1$, $\mathbf{TS}_2$, ..., $\mathbf{TS}_n$
3:     for each $\mathbf{TS}_i$
4:         $\mathbf{TC}_i = OPTICS_t(\mathbf{TS}_i)$
5:         emit($\mathbf{TS}_i, \mathbf{TC}_i$)
6:     end for

7: function Reduce(Trajectory Cluster Sets $\mathbf{TC}$: $TC_1$, $TC_2$, ..., $TC_m$)
8:     merge($\mathbf{TC}$) using Algorithm 2 in Figure 5
9:     do classification for the outliers
```

Fig. 6: MapReduce-based Trajectory Clustering

their advantages and disadvantages. The first approach is fast and simple, because the representative trajectory is known after the cluster merging. However, it loses the diversity of the trajectories in the cluster. The second approach is better with the diversity, however, it takes much longer to calculate because the complexity is $O(|TC|)$.

Since the merging process is not mandatory, we suggest that when the merging process exists, we use the first approach. Otherwise, when the diversity is important, or the merging process does not exist, we use the second approach. If the merging process does not exist, we still need $O(|TC|)$ time to calculate the representative trajectory.

5.4 Parallelizing Clustering in the Cloud Using MapReduce

Our proposed Hierarchical RFID Trajectory Clustering algorithm can be naturally parallelized because it adopts the divide-and-conquer paradigm to reduce cost for dealing with large set of trajectory data. The sub-sets of trajectories do not overlap with each other, because they do not follow the same routes. The merging and classification steps are essentially gathering results from the subsets and providing a result in aggregation. In cloud computing environment, MapReduce is the best tool to parallelize such kind of algorithms.

We further propose a MapReduce-based parallel trajectory clustering algorithm, Algorithm 3 as shown in Figure 6. The algorithm works as the following. In the Map function, the sets of trajectories are split into subsets according their routes (line 2). For each subset, we use $OPTICS_t$ algorithm to cluster it (line 3). The small clusters are then sent to workers for further processing (line 4). During the Reduce phase, we merge the small clusters generated from the worker nodes (line 8) and classify the outliers to their nearest clusters (line 9).

6 Experimental Evaluation

We have conducted extensive experiments to evaluate the performance of the proposed approach. In this section, we particularly focus on reporting four experimental results: i) the quality of our trajectory clustering algorithm; ii) the performance of the clustering algorithm; iii) the effects of different distance functions, and iv) the performance of the MapReduce-based parallelizing clustering. All experiments were conducted on a Quad Core 2.7GHz PC with 2GBytes of main memory, running on Windows 7. Our proposed algorithms were implemented in Java 6.0.

6.1 Experimental Datasets

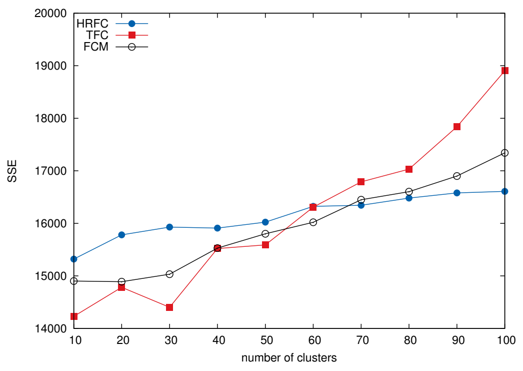
To the best of our knowledge, at the time of this writing, there are no public datasets for RFID trajectories available. In our experiments, we altered the CENTRE [4] trajectory generator to generate several sets of data. CENTRE is a generator of spatio-temporal objects that evolve in space and time producing a sequence of samples (i.e., spatial locations and their corresponding observation times) called trajectories. The main idea is to generate trajectories that follow some pre-defined clusters in order to stress and probe the limits of a clustering algorithm. In order to adapt with RFID trajectories, we modified the algorithm to generate spatial value single-dimensionally, while generate observation time as two-dimensional points.

Dataset	Density	Noise
1	Low	Low
2	High	Low
3	Low	High
4	Medium	Medium

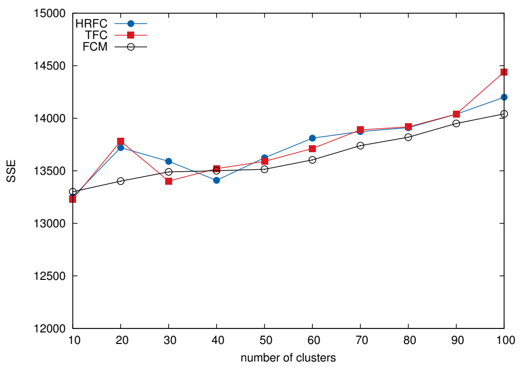
TABLE 3: Datasets Generation

The alerted generation algorithm works as the following. Firstly, we define the centroid trajectories which serve as the base of the generation. Then, for each node at the trajectory, we generate a randomized point in the time plane which is within a given threshold away from the centroid point. We name the threshold as *Shift Threshold*. To simulate the missing readings, we randomly choose some nodes at which no observations are made. The trajectories are along routes in an RFID network which is randomly generated. This network has 100 nodes and each node has random outbound degree between 1 and 10. The modified trajectory generator can produce data varying in several aspects.

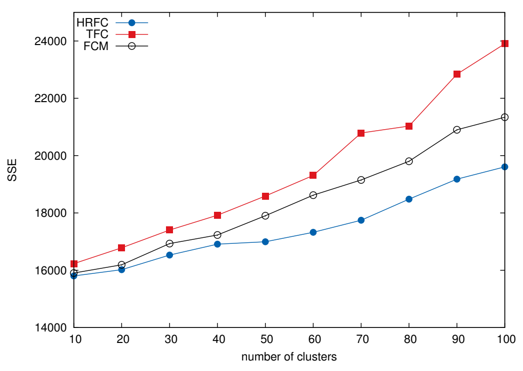
- *Number of clusters:* We can control the number of clusters contained in the generated trajectories by specifying different number of centroid trajectories.
- *Density of clusters:* We can control the density of clusters by constraining the shift threshold. The smaller the shift threshold is, the denser the cluster is. The effect of this attribute of clusters is that when the density of the clusters is high, they have less chance to be mixed



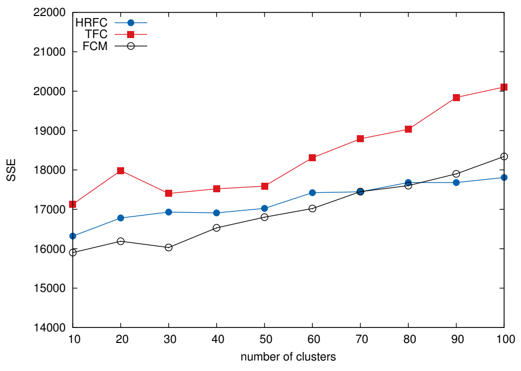
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Fig. 7: Clustering Quality Comparison

with each other. In contrast, when the density of the clusters is low, they have more chance to get mixed. We define three discrete values for densities, namely *High*, *Medium* and *Low*, as the following: for a node at a centroid trajectory, suppose the average distance at time dimensions is d, then we have high density when the shift threshold is $0.3 * d$, medium density when the shift threshold is $0.6 * d$ and low density when the shift threshold is $0.8 * d$.

- *Noise level:* We control the level of noises by specifying different probabilities of missing reading happening (noted as p_{mr}) during generation. Similar to the generation of different densities of clusters, we define three discrete values of noise levels, namely *High*, *Medium* and *Low*, as the following: we have high level of noising when the value of p_{mr} is 0.5, medium level of noising when p_{mr} is 0.3, and low level of noising when p_{mr} is 0.1.

In our experiments, we generated four static sets of data with different configurations, as described in Figure 3. Specifically, Dataset 1 is the "mixed scenario" where the density of the clusters are low, meaning they will likely overlap with each other. Dataset 2 is the "good scenario" where the objects move together and they seldom overlap with each other in the time dimension. In addition, Dataset 2 contains less missing readings. Dataset 3 is the "worst scenario", because the density is low while it contains more missing readings. Finally, Dataset 4 is the "medium scenario".

6.2 Clustering Quality

In this experiment, we compared our trajectory clustering algorithm with two other algorithms in terms of quality. One is the Time-Focused Clustering (TFC) algorithm [22] and the other is the Fuzzy C-Means (FCM) [24]. We implemented the variations of these two algorithms using the trajectory definition in our work.

Our aim is to measure the clustering quality while varying the number of clusters in the datasets. Unfortunately, there is no well-defined measures for density-based clustering methods. We therefore defined a simple quality measure for our analysis. In particular, we exploited the Sum of Squared Error (SSE) [6] to represent the quality of the clustering.

Definition 8 (*Qualify Measure of Clustering Algorithms*).

$$Q = \sum_{i=1}^{n_{clusters}} \left(\frac{1}{2|C_i|} \sum_{x \in C_i} \sum_{y \in C_i} dist_t(x, y)\right)$$

Figure 7 shows the SSE for the three clustering algorithms (HRTC, TFC and FCM) with the four datasets defined in Table 3. From the results, we can conclude that the number of clusters affects the performance of all three algorithms. The reason behind this is that with the increase of the number of clusters, trajectories are more likely to overlap in time dimension. As a result, the clustering errors will increase.

For Dataset 1 (Figure 7a), we can see that our proposed HRTC outperforms the other two algorithms after the number of clusters becomes considerably large (70 in our

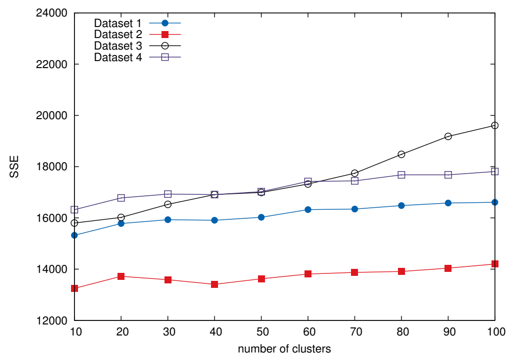


Fig. 8: Clustering Quality vs. Uncertainty

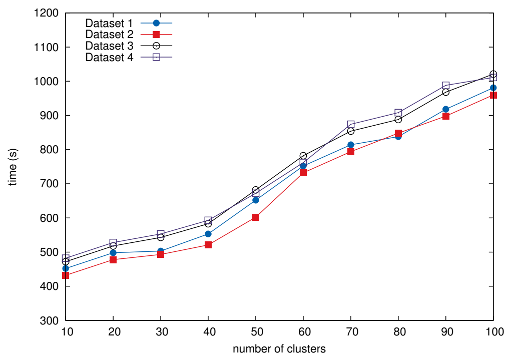


Fig. 10: Clustering Efficiency vs. Uncertainty

experiment). This is because HRTC is best among the three in describing the co-location characteristics of RFID objects that dominate the quality of clustering (SSE) when the number of clusters is large (above 70 in our experiment), FCM suits for the RFID data that follows a linear regression model when the data appears truly random, and TFC is best when the number of clusters is relatively small. However, TFC quickly becomes the worst among the three with the increase of the clusters because it does not cope well with uncertainties.

For Dataset 2 (Figure 7b), we can see the SSE for three algorithms are much similar with Dataset 1. This is because Dataset 2 is the "good scenario" where almost no uncertainty is introduced to the clustering data. Though HRTC and FCM both take uncertainty into account, they only outperform TFC noticeably when the number of clusters is significantly high (100 in our experiment), where the trajectories are more likely to overlap.

For Dataset 3 (Figure 7c), it is clear that in extreme conditions where trajectories overlap a lot and the level of noise is high, HRTC outperforms the other algorithms. The reason is twofold. Firstly, HRTC introduces the classification process which puts the identified outliers into the best possible clusters, which minimizes the errors. The other two algorithms both perform the hard partition where outliers are not identified but forced to be accepted by one of the clusters. Secondly, HRTC treats locations in the RFID trajectories as non-directional points and models the time dimension as a range, while the other two algorithms have to take the median of range in order to work.

For Dataset 4 (Figure 7d), the result exhibits a similar pattern as for Dataset 1, with the exception that TFC is always outperformed by the other two algorithms. We can conclude that for an RFID trajectory which has its unique characteristics, a model that fits better can bring much benefit for the clustering.

Because in most application scenarios, the number of clusters produced is usually large due to the high level of uncertainties in the datasets and weak correlation among events, our HRTC is clearly superior to FCM and TFC in clustering quality measured by SSE. This outperformance becomes increasingly significant as the number of clusters increases because HRTC can cope with uncertainties much better than the other two. This observation can be easily verified from Fig. 7a, Fig. 7c and Fig. 7d, with the exception of Fig. 7b in which no uncertainty is taken into consideration.

We are also interested in how HRFC performs on different datasets. Figure 8 shows the result of this experiment. It is easy to see that the performance of HRFC is affected by the level of uncertainty in the datasets. For Dataset 2, it performs the best, while for Dataset 3, the SSE increases noticeably faster than the other scenarios. However, the overall performance of HRFC is quite steady.

6.3 Clustering Efficiency

In this experiment, we compared our trajectory clustering algorithm with two other algorithms (TFC and FCM) in terms of efficiency. We recorded the time that was used to finish clustering for all three algorithms, and the results are shown in Figure 9.

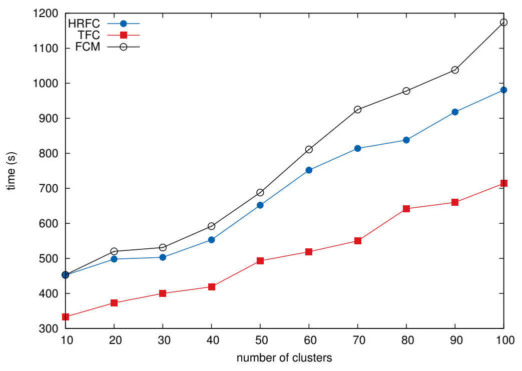
In general, we can see that TFC outperforms the other two algorithms. This is because it does not care about uncertainties in the data so it does not spend extra time on it. Between the two algorithms that take uncertainty into account, HRFC outperforms FCM because FCM needs to run several rounds of the clustering algorithm on all trajectories in the dataset as a whole, while HRFC splits the dataset first and run clustering algorithm on each partition.

From Dataset 3, we can see that both HRFC and FCM suffer from the uncertainties in terms of running time. For HRFC, it is because there are more outliers so more classification steps ($number_{outliers} * number_{clusters}$) were involved. For FCM, because it has to run more rounds of clustering to reach the stable condition, a significant amount of time is required. From Dataset 2, we can see that when the level of uncertainty is not high, the running time required by HRFC is close to TFC.

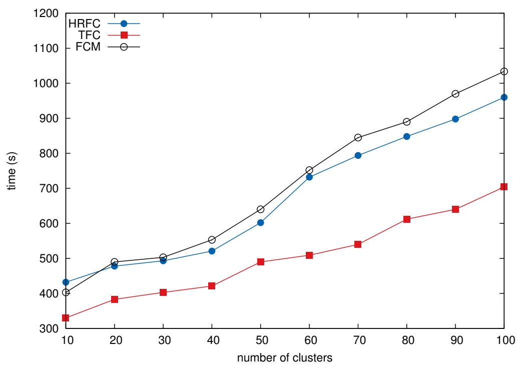
We are also interested in how efficient HRFC is for the four different datasets. Figure 10 shows the result of this experiment. From the figure, it is easy to see that the performance of HRFC is affected by the level of uncertainty in the datasets.

6.4 Similarity Functions

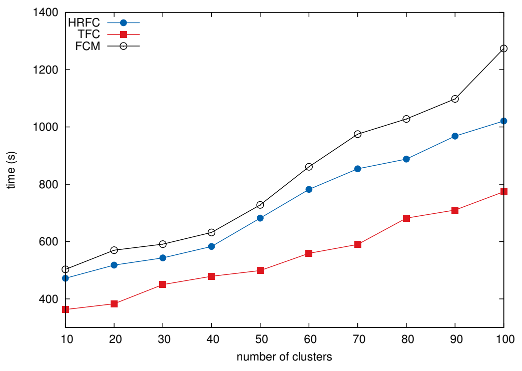
In this experiment, we compared the quality of our proposed algorithm with different similarity functions, including LCSS, EDR and the two functions that are introduced in Section 4, namely TED and TCLSS. The result is shown in Figure 11.



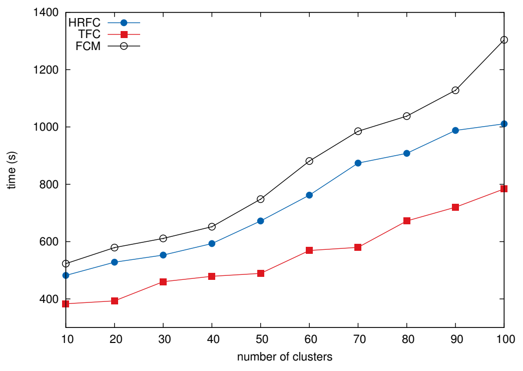
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Fig. 9: Clustering Efficiency Comparison

From the figure, it is clear that the proposed TED and TCLSS produce better clustering quality than LCSS and EDR. This is mainly due to the reason that they take the time dimension into account for measuring the similarity of two trajectories. TED and TCLSS almost yield the same clustering quality in all four datasets.

6.5 Parallelization Performance

In this experiment, we compared the performance of HRFC algorithm under centralized and parallelized settings. We particularly implemented the MapReduce-based clustering algorithm proposed in Section 5.4. The parallelized algorithm was implemented in Java using Hadoop framework. The experiment was conducted using ten Amazon EC2 micro instances. The result is illustrated in Figure 12.

From the figure, we can see that the MapReduce-based algorithm outperforms its centralized counterpart when the number of clusters reaches 55. When the number of clusters is less than 55, the communication cost used to transfer the data to worker nodes is larger than the time saved by parallelization.

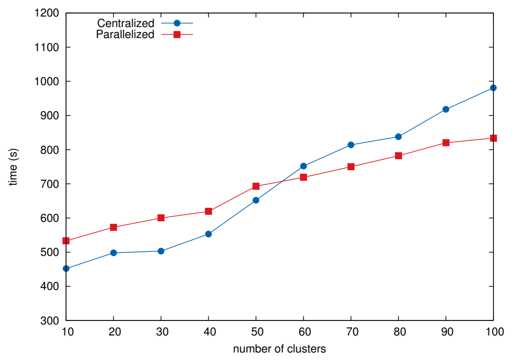
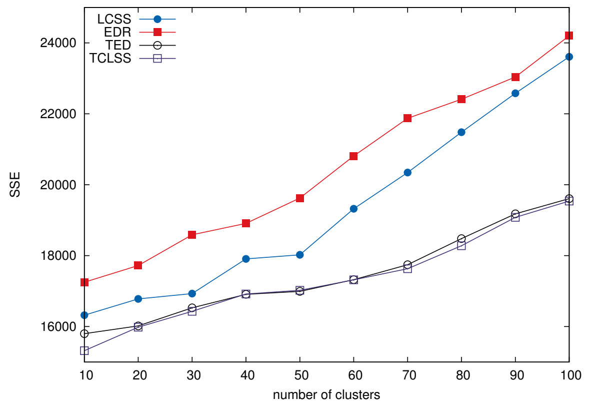


Fig. 12: Parallelization Performance

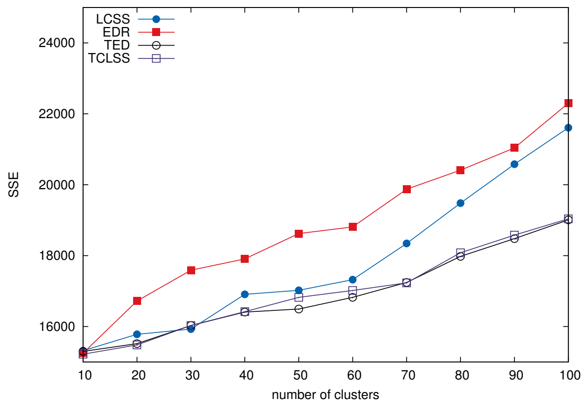
7 Conclusion

Recent advances in technologies such as radio-frequency identification (RFID) have made automatic tracking and tracing possible in a wide range of applications. However, there are still numerous technical difficulties in realizing traceability applications in large-scale, uncertain environments such as the emerging Internet of Things (IoT). In this paper, we have introduced an efficient trajectory model and developed a novel clustering algorithm to cluster RFID trajectories with the capability to recover missing readings. Our algorithm is scalable and efficient, outperforming existing methods such as Time-Focused Clustering (TFC) algorithm and Fuzzy C-Means, as demonstrated by the results from extensive experimental studies.

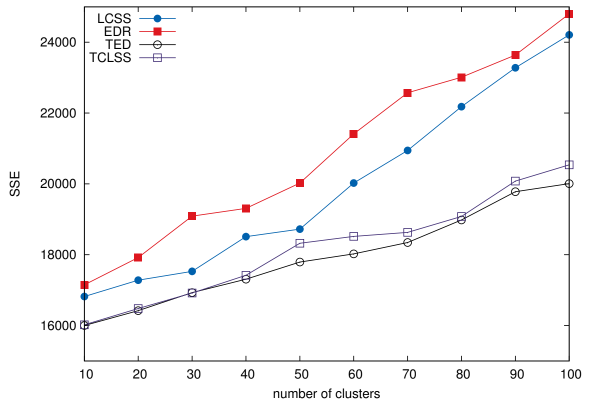
The experimental studies of our clustering algorithm have been conducted using synthetic and offline data. Our future work includes further performance evaluation with real data from a large-scale supply chain management system, and online clustering and recovering of RFID trajectories.



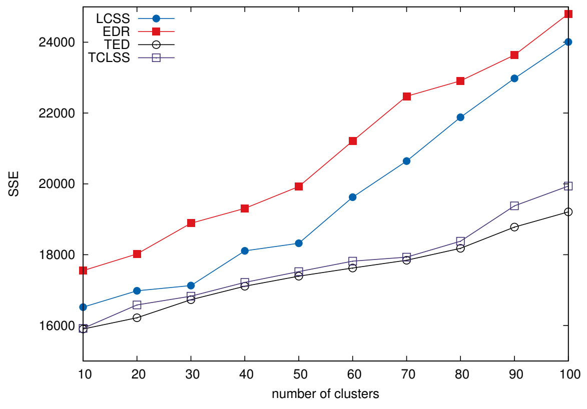
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Fig. 11: Similarity Functions Comparison

8 Acknowledgments

The work of Hong Shen was supported by the 985 Project funding of Sun Yat-sen University, National Science Foundation of China under its General Projects funding #61170232, State Key Laboratory of Rail Traffic Control and Safety Research Grant RS2012K011. Quan Z. Sheng's work was partially supported by Australian Research Council (ARC) Linkage Project LP100200114 and Discovery Project DP130104614. The authors would like to thank the anonymous reviewers for their valuable feedback on this work.

References

[1] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. Efficient Pattern Matching over Event Streams. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of data (SIGMOD'08)*, Vancouver, Canada, 2008.

[2] Rebecca Angeles. RFID Technologies: Supply-Chain Applications and Implementation Issues. *Information Systems Management*, 22(1):51–65, December 2005.

[3] Scott Gaffney and Padhraic Smyth. Trajectory Clustering with Mixtures of Regression Models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.

[4] F. Giannotti, A. Mazzoni, S. Puntoni, and C. Renso. Synthetic Generation of Cellular Network Positioning Data. In *Proc. of the 13th Annual ACM Intl. Workshop on Geographic Information Systems (GIS'05)*, Bremen, Germany, 2005.

[5] Hector Gonzalez, Jiawei Han, and Xuehua Shen. Cost-Conscious Cleaning of Massive RFID Data Sets. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE'07)*, Istanbul, Turkey, April 2007.

[6] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[7] Shawn Jeffery, Gustavo Alonso, Michael Franklin, Wei Hong, and Jennifer Widom. Declarative Support for Sensor Data Cleaning. *Pervasive Computing*, 3968:83–100, 2006.

[8] Shawn Jeffery, Michael Franklin, and Minos Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. *The VLDB Journal*, 17:265–289, 2008.

[9] Shawn Jeffery, Minos Garofalakis, and Michael Franklin. Adaptive Cleaning for RFID Data Streams. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*, Seoul, Korea, September 2006.

[10] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of Convoys in Trajectory Databases. *Proceedings of VLDB Endowment*, 1(1):1068–1080, August 2008.

[11] Nodira Khoussainova, Magdalena Balazinska, and Dan Suciu. Probabilistic Event Extraction from RFID Data. Cancun, Mexico, April 2008.

[12] Wei-Shinn Ku, Haiquan Chen, Haixun Wang, and Min-Te Sun. A Bayesian Inference-Based Framework for RFID Data Cleansing. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2012.

[13] Jeremy Landt. The History of RFID. *IEEE Potentials*, 24(4):8–11, October/November 2005.

[14] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. TraClass: Trajectory Classification Using Hierarchical Region-based and Trajectory-based Clustering. *Proceedings of VLDB Endowment*, 1(1):1081–1094, August 2008.

[15] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory Clustering: a Partition-and-Group Framework. In *Proceedings of the 2007*

ACM SIGMOD International Conference on Management of Data (SIGMOD'07), Beijing, China, 2007.

[16] Julie Letchner, Christopher Re, Magdalena Balazinska, and Matthai Philipose. Access Methods for Markovian Streams. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE'09)*, Shanghai, China, 2009.

[17] Xi Li, Weiming Hu, and Wei Hu. A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, Hong Kong, 2006.

[18] Xiang Lian and Lei Chen. Similarity join processing on uncertain data streams. *IEEE Transactions on Knowledge and Data Engineering*, 23:1718–1734, 2011.

[19] Guoqiong Liao, Jing Li, Lei Chen, and Changxuan Wan. KLEAP: an Efficient Cleaning Method to Remove Cross-reads in RFID Streams. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*, Glasgow, Scotland, UK, 2011.

[20] Mo Liu, Ming Li, Denis Golovnya, Elke A. Rundensteiner, and Kajal Claypool. Sequence Pattern Query Processing over Out-of-Order Event Streams. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE'09)*, Shanghai, China, 2009.

[21] L.V. Massawe, H. Vermaak, and J.D.M. Kinyua. An Adaptive Data Cleaning Scheme for Reducing False Negative Reads in RFID Data Streams. In *Proceedings of the 2012 IEEE International Conference on RFID (RFID'12)*, Orlando, USA, April 2012.

[22] Mirco Nanni and Dino Pedreschi. Time-focused Clustering of Trajectories of Moving Objects. *Journal Intelligent Information Systems*, 27(3):267–289, November 2006.

[23] Yanming Nie, Zhanhuai Li, and Qun Chen. Complex Event Processing over Unreliable RFID Data Streams. In *Proceedings of the 13th Asia-Pacific Web Conference on Web Technologies and Applications (APWeb'11)*, Beijing, China, 2011.

[24] Nikos Pelekis, Ioannis Kopanakis, Evangelos Kotsifakos, Elias Frentzos, and Yannis Theodoridis. Clustering Trajectories of Moving Objects in an Uncertain World. In *Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM'09)*, pages 417–427. IEEE, 2009.

[25] C. Piciarelli and G. L. Foresti. On-line Trajectory Clustering for Anomalous Events Detection. *Pattern Recognition Letters*, 27(15):1835–1842, November 2006.

[26] Jun Rao, Sangeeta Doraiswamy, Hetal Thakkar, and Latha S. Colby. A Deferred Cleansing Method for RFID Data Analytics. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*, Seoul, Korea, September 2006.

[27] Christopher Ré, Julie Letchner, Magdalena Balazinksa, and Dan Suciu. Event Queries on Correlated Probabilistic Streams. In *Proceedings of the 2008 ACM International Conference on Management of Data (SIGMOD'08)*, Vancouver, Canada, 2008.

[28] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. The Evolution of RFID Security. *IEEE Pervasive Computing*, 5(1):62–69, March 2006.

[29] Quan Z. Sheng, Xue Li, and Sherali Zeadally. Enabling Next-Generation RFID Applications: Solutions and Challenges. *IEEE Computer*, 41(9):21–28, September 2008.

[30] Thanh Tran, Liping Peng, Yanlei Diao, Andrew McGregor, and Anna Liu. CLARO: Modeling and Processing Uncertain Data Streams. *The VLDB Journal*, PP(99):1–26, 2011.

[31] Thanh Tran, C. Sutton, R. Cocci, Nie Yanming, Diao Yanlei, and P. Shenoy. Probabilistic Inference over RFID Streams in Mobile Environments. In *Proceedings of the 25th International Conference on Data Engineering (ICDE'09)*, Shanghai, China, April 2009.

[32] Thanh T. L. Tran, Andrew McGregor, Yanlei Diao, Liping Peng, and Anna Liu. Conditioning and Aggregating Uncertain Data Streams: Going Beyond Expectations. *Proceedings of VLDB Endowment*, 3(1-2), September 2010.

[33] Thanh T.L. Tran, Liping Peng, Boduo Li, Yanlei Diao, and Anna Liu. PODS: a New Model and Processing Algorithms for Uncertain Data Streams. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD'10)*, Indianapolis, Indiana, USA, 2010.

[34] Michail Vlachos, Dimitrios Gunopoulos, and George Kollios. Discovering Similar Multidimensional Trajectories. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, San Jose, United States, 2002.

[35] L. Wang, D. Cheung, R. Cheng, S. Lee, and X. Yang. Efficient Mining of Frequent Itemsets on Large Uncertain Databases. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1, 2011.

[36] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Complex Event Processing over Uncertain Data. In *Proceedings of the second International Conference on Distributed Event-based Systems (DEBS'08)*, Rome, Italy, 2008.

[37] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Efficient Processing of Uncertain Events in Rule-Based Systems. *IEEE Transaction on Knowledge and Data Engineering*, 24(1):45–58, January 2012.

[38] Evan Welbourne, Nodira Khoussainova, Julie Letchner, Yang Li, Magdalena Balazinska, Gaetano Borriello, and Dan Suciu. Cascadia: A System for Specifying, Detecting, and Managing RFID Events. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys'08)*, Breckenridge, USA, 2008.

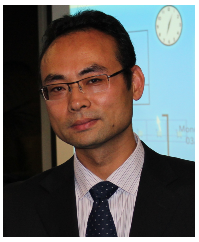
[39] Yanbo Wu, Damith Ranasinghe, Quan Z. Sheng, Sherali Zeadally, and Jian Yu. RFID Enabled Traceability Networks: A Survey. *Distributed and Parallel Databases*, 29(5-6):397–443, 2011.

[40] Zhou Zhao, Da Yan, and Wilfred Ng. Mining Probabilistically Frequent Sequential Patterns in Uncertain Databases. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT'12)*, Berlin, Germany, 2012.

[41] H. Ziekow and L. Ivantysynova. A Probabilistic Approach for Cleaning RFID Data. In *Proceedings of the 24th International Conference on Data Engineering Workshop (ICDEW'07)*, Istanbul, Turkey, April 2008.

Yanbo Wu received the PhD degree in computer science from the University of Adelaide, Australia. He is a lecturer in the School of Computer Science and Information Technology at Beijing Jiaotong University, China. His research interests include Internet of Things, distributed database and mobile computing. He has published papers in various peer reviewed journals, including IEEE Transactions on Parallel and Distributed Systems, Distributed and Parallel Databases and The Computer Journal (Oxford). He is the recipient of Google PhD Top-Up Grant in 2011.

Hong Shen is currently a specially-appointed professor at Sun Yat-sen University, China, and a tenured Professor (Chair) of Computer Science at University of Adelaide, Australia. He received the B.Eng. degree from Beijing University of Science and Technology, M.Eng. degree from University of Science and Technology of China, Ph.Lic. and Ph.D. degrees from Abo Akademi University, Finland, all in Computer Science. He was Professor and Chair of the Computer Networks Laboratory in Japan Advanced Institute of Science and Technology (JAIST) during 2001-2006, and Professor (Chair) of Compute Science at Griffith University, Australia, where he taught 9 years since 1992. With main research interests in parallel and distributed computing, algorithms, data mining, privacy preserving computing, high performance networks and multimedia systems, he has published more than 300 papers including over 100 papers in international journals such as a variety of IEEE and ACM transactions. Prof. Shen received many honours/awards including China National Endowed Expert of “1000 People Plan” and Chinese Academy of Sciences “Hundred Talents”.



Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Sydney, Australia. He is an Associate Professor in the School of Computer Science at the University of Adelaide. His research interests include service-oriented architectures, distributed computing, and Web of Things. He is the recipient of Chris Wallace Award in 2012 and Microsoft Research Fellowship in 2003. He is the author of more than 160 publications. He is a member of the IEEE and the ACM.