International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Performance Evaluation of Weighted Time-Parameterized Edit–Distance based Trajectory Clustering in RFID Environment

Sagar V. Mahajan[1], R. H. Borhade[2]

[1]Department of Information Technology, Smt. Kashibai Navale Collage of Engineering, Pune, India

[2]Professor, Department of Information Technology, Smt. Kashibai Navale Collage of Engg., Pune, India

Abstract: ***Technologies such as Radio-Frequency Identification (RFID) are used for automatic tracking and tracing in a wide range of applications. RFIDs are accompanied by noise. The Performance of RFIDs is dependent on cleaning, filtering and reducing errors. RFID tags data capture can be enhanced using trajectories for dataset generation and applying mining, clustering and, probabilistic techniques. Trajectory clustering is using RFID data management and mining has a wide range of applications in various areas, such as traffic monitoring, video surveillance, cattle tracking and supply chain management. Trajectory clustering is the method of classifying similar trajectories according to a similarity distance. Depending on the task, given a set of trajectories, one may want to find clusters of objects that followed the same path or detect groups that moved together for given period of time. Hierarchical RFID Trajectory Clustering uses similarity measure called; Time-parameterized Edit Distance (TED) is improvised weightage for the different parameters. TED considers the same weightage for all the parameters.Our proposed weighted time parameterized similarity measure is used to find the similarity between to trajectory path by taking into consideration time dimension values in the calculation. Also, this model can deal with variants in both time and space dimensions and the clustering algorithm are much less sensitive to noise and outliers than existing methods. In this paper we are presenting performance evaluation of our proposal.***

Keywords: Radio-Frequency Identification (RFID); Time Parameterized Edit Distance (TED); Weighted Time-parameterized Edit Distance (WTED)

1. Introduction

RFID (Radio Frequency Identification) technology promises revolutions in areas such as supply chain management and omnipresent computing enabled by pervasive, low-cost sensing and identification [1]. One of the primary factors limiting the widespread adoption of RFID technology is the unreliability of the data streams produced by RFID readers [2, 3]. The observed read rate (i.e., percentage of tags in a reader's vicinity that are actually reported) in real-world RFID deployments is often in the 60−70%

Range [3, 4]; in other words, over 30% of the tag readings are routinely dropped. Unfortunately, such error rates render raw RFID streams essentially useless for higher-level applications (such as accurate inventory tracking). Instead, RFID middleware systems are typically deployed between the readers and the application(s) to correct for dropped readings and provide "clean" RFID readings to application logic. The standard data-cleaning mechanism in most such systems is a temporal "smoothing filter": a sliding window over the reader's data stream that interpolates for lost readings from each tag within the time window [5, 6]. The goal, of course, is to decrease or eliminate dropped readings by giving each tag more opportunities to be read within the smoothing window. RFID signals are characterized by inaccuracy in RFID read events. Noise may affect Read events [7]. Accurate data capture in a stack reader of a production robot that must identify the topmost item in the stack [8]. Another example is asset tracking with reader equipped forklifts[9].Here, one need to determine where items are picked and dropped.

Due to an advancement of RFID technology, the tremendous amount of information has been triggered recently through Networked RFID is one of the crucial technological advances that help make RFID-enabled traceability possible. The increasing information requires the data abstraction techniques that can be simply achievable trough clustering. Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects. Clustering has been broadly applied in numerous applications such as market research, pattern recognition, data analysis, and image processing. Here, trajectory clustering is a novel and statistically well-founded method for clustering time series data from gene expression arrays, and it has applications in many areas such as traffic monitoring, video surveillance, cattle tracking and supply chain management. Trajectory clustering uses non-parametric statistics and is hence not receptive to the particular distributions underlying gene expression data. Each cluster is clearly defined regarding direction of change of expression for successive time points, i.e., its trajectory. Recent developments in satellites and tracking facilities have made it possible to collect a large amount of trajectory data and there is increasing interest to perform data analysis over these trajectory data. Thus, an efficient clustering algorithm for trajectories is essential for such data analysis tasks.

2. Related Work

To remunerate for the fundamental unreliability of RFID data streams, most RFID middleware systems employ a "smoothing filter", a sliding-window aggregate that interpolates for lost readings.

In [10], 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 and it does not propose the optimal smoothing filter for the readings of single tag and an aggregate signal. The inherent uncertainty in RFID signals requires an RFID middleware system to clean the input data after capturing. Typically these systems employ a low pass filter for reducing errors.

In [11], an approach is proposed for data cleaning that exploits primary characteristics of RF signals as well as maximum likelihood operations. With this filter, proximity detection of RFID tags is improved. This permits reasoning about the position of RFID tags in the reader's range without measuring the signal strength of tag responses. It is, therefore, applicable on top of standard reader interfaces. Also, it improves data cleaning wherever the tag to reader distance is relevant. For instance, this qualifies correct ordering of items that pass by a reader on a conveyor or enhances tracking scenarios with RFID-equipped forklifts. However, the scheme considers simplified properties of RF signals, rather than general properties arising in applications.

A major problem in recognizing events in streams of data is that the data can be imprecise (e.g. RFID data). However, some state-of-the-art event detection systems assume that the data is accurate. Noise in the data can be captured using methods such as hidden Markov models. Inference on these models creates streams of probabilistic events that cannot be directly queried by existing systems. To address this challenge Lahar, an event processing system for probabilistic event streams is proposed. By using the probabilistic nature of the data, Lahar yields a much higher recall and precision than deterministic techniques operating over only the most feasible tuples. By using a static analysis and novel algorithms, Lahar processes data orders of magnitude more efficiently than a naïve approach based on sampling. Unfortunately, it does not provide a comprehensive analysis on the performance of these algorithms [12].

Recent discoveries in RFID technology are facilitating large-scale, cost-effective deployments in retail, healthcare, pharmaceuticals and supply chain management. The advent of mobile or handheld readers adds meaningful new challenges to RFID stream processing due to the inherent reader mobility, increased noise, and incomplete data. In [13], they addressed the problem of translating noisy, incomplete raw streams from mobile RFID readers into clean, perfect event streams with location information. Specifically, it proposes a probabilistic model to capture the mobility of the reader, object dynamics, and noisy readings. This model can self-calibrate by automatically estimating key parameters from observed data. Based on this model, it employed a sampling-based technique called particle filtering to gather clean, precise information about object locations from raw streams from mobile RFID readers. Since inference based on standard particle filtering is neither scalable nor efficient in our settings, three enhancements such as particle factorization, spatial indexing, and belief compression were proposed for scalable inference over large numbers of objects and high volume streams. Uncertain data streams, where data is incomplete, imprecise, and even misleading, have been observed in many environments. Feeding such data streams to existing stream systems produces results of unknown quality, which is of paramount concern to monitoring applications.

In [14], the PODS system is aimed, that supports stream processing for random data naturally captured using continuous random variables. PODS employ a unique data model that is flexible and allows efficient computation. Built on this model, evaluation techniques are improved for complex relational operators, i.e., aggregates and joins, by exploring advanced statistical theory and approximation. However, the works on ranking in probabilistic databases give simplistic solutions to handling continuous distributions.

In [15], an approach is proposed for data cleaning that exploits basic characteristics of RF signals as well as maximum likelihood operations. With this filter, proximity detection of RFID tags is improved. This enables reasoning about the position of RFID tags in the reader's range without measuring the signal strength of tag responses. It is therefore applicable on top of standard reader interfaces. Also, it improves data cleaning wherever the tag to reader distance is relevant. For instance this enables correct ordering of items that pass by a reader on a conveyor or enhances tracking scenarios with RFID equipped forklifts. However, the scheme considers simplified properties of RF signals, rather than general properties arising in applications.

3. Design Details

This work aims to devise a Weighted Time-parameterized Edit Distance-based trajectory clustering in RFID environments. In the proposed method, this similarity function will be enhanced with weighted values. The Weighted Time-parameterized Edit Distance (WTED) will be then applied for finding the similarity between two trajectories in trajectory clustering. At first, every RFID points are projected to single cluster by putting all the trajectory points into single plane. In the second step, every data point points are merged to get the required number of trajectory clusters. During the merging operation, the RFID points are joined together into sub-clusters which represent the "branches" at that node. Here, merging operation is used for merging two sets of trajectories into one set using the proposed similarity function which find the suitable RFID points to join into a single cluster. This process is continued until we get the required number of clusters. The proposed trajectory clustering will be implemented using JAVA programming and the performance of the proposed clustering will be validated using clustering quality. The use case for the same is depicted in Fig 1.

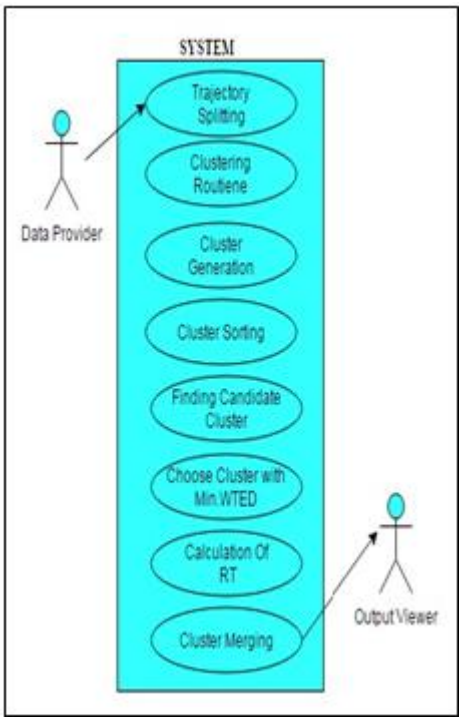

Figure 1: Use Case Of Proposed System

Our clustering algorithm is a hierarchical one. The basic idea is to first cluster the points projected on the (Ts, Te) 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.

In Algorithm 1, a cluster TCpi has two characteristics, namely the path p and the sub-cluster IDi. Path p highlights this cluster with the common moving path of the trajectories, which is the projection of the trajectories on the V dimension, while the sub-cluster ID highlights the difference of the trajectories along the same path p, by clustering them with the values of the depth elements in T s and Te 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. Then, for each set, the OPTICS clustering routine is performed on the time dimensions of the next stop. This process continues recursively until the length of the path reaches the MAX DEPTH constant. This algorithm actually generates a forest of clusters incrementally. 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 has to contain at least a minimum number (npts) 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 -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 (ts, te) as a point in a two dimensional plane. As a result, we denote the function as OPTICSt. 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.

3.1 Merging Clusters

Generally, when missing readings happen, the trajectories will have a missing node. For example, a cluster with path p1 = {v1, v2, v3, v4} will become p0 1 = {v1, v3, v4} when objects are not read at v2. Using the hierarchical clustering algorithm, this path will be treated as a different path than p1, instead of an outlier. However, noticing that p0 1 is a subpath of p1, 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, an efficient approach is developed. Given two clusters T C1 and T C2, 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, {v1, v2, v3, v4} and {v1, v3, v4} might be able to be merged, but {v1, v2, v3, v4} and {v1, v4, v3} might not. Obviously, the time complexity of this merge is O(1).

Also it is necessary to consider the time dimension in merging. In this work, the similarity function TED introduced is used to determine whether two clusters are similar enough to be merged. However, since clusters might contain many trajectories, calculating average similarity for trajectories in T C1 and T C2 will take O(|T C1| ∗ |T C2|) 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 (|T C1|+|T C2|).

RTS is the similarity of the representative trajectories of two clusters. First, generate the representative trajectory of a cluster. Given the fact that a cluster might contain trajectories of different lengths because of previous merging, the longest path is selected as the node dimension representation. In order to generate the time dimension representations, the average ts/te is calculated 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, simply ignore this trajectory when calculating the average time representations at that node. It is easy to prove that the calculation of RT is of complexity O(|T C|). As the result, calculating RTS will take O(|T C1| + |T C2|) time. With RTS, the cluster merging algorithm is proposed. Firstly, the clusters are sorted by the length of the paths, so that the merging is an one-path operation. Then starting from the cluster with the shortest path, we find all the candidate clusters and choose the one with the minimum T ED distance and the distance is lower than a given threshold to merge into. Note that the calculation of RT can be time consuming, so cache that the value for each cluster and re-calculate it 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(|TR|).

3.2 Nearest Neighbor Classification

For detected outliers, it is 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 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, it is suggested that when the merging process exists, the first approach is used. Otherwise, when the diversity is important, or the merging process does not exist, the second approach can be used. If the merging process does not exist, O(|TC|) time is required to calculate the representative trajectory.

3.3 Parallelizing Clustering in the Cloud Using

The 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. MapReduce-based parallel trajectory clustering algorithm works as the following. In the Map function, the sets of trajectories are split into subsets according their routes. For each subset, OPTICSt algorithm is used to cluster it. The small clusters are then sent to workers for further processing. During the Reduce phase, the small clusters generated from the worker nodes are merged and classify the outliers to their nearest clusters. This Traceability Network: A traceability network is a directed graph G = (V, E). V represents the set of nodes where RFID readers are deployed and 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 reader 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.

Trajectory: A trajectory of a given RFID tagged object o_i is a polyline in a three dimensional space (V, T_s, T_e), where T_s is the time space for arrival readings and 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: $TR_i = \{o_i, \{(v_1, t_{s1}, t_{e1}), (v_2, t_{s2}, t_{e2}), \ldots, (v_n, t_{sn}, t_{en})\}\}$. Its V-axis values, ordered by T_s values, form a path P in G. The set of all trajectories is denoted as ST. The deployment of RFID readers may affect the timestamps of arrival/leaving readings: (i) 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., $t_s < t_e$. (ii) If only one reader is deployed at a node and only one reading of each object is captured, $t_s = t_e$. (iii) 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., $t_s \leq t_e$.

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, $TC = \{(v_1, (t_{s1},t_{e1})), (v_2, (t_{s2},t_{e2})),\ldots, (v_n, (t_{sn},t_{en}))\}$. Now we define the research problems of managing uncertainties for traceability as follows.

1) 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 reasons 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$p_{outlier}(TR') \geq$ $_{outlier}$, where $_{outlier}$ is a predefined threshold.

2) 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 $SC = \{TC_1,TC_2,\ldots,TC_n\}$, the recovery task can be transformed to a classification problem.

3) Clustering. In most cases, the set of trajectory clusters $SC = \{TC_1,TC_2,\ldots,TC_n\}$ is not known beforehand. Moreover, it may be change occasionally. As the result, in order for the classification to work, it is necessary to generate *SC* by clustering the existing trajectories. A similarity measurement model called Weighted Time-parameterized Edit Distance (WTED) is proposed where the time dimension values are also used in the calculation.

In WTED's we introduce a new function called Weighted Time-parameterized Distance (WTPD) for two elements e_1 and e_2 in two trajectories, namely

$$dist_t(e_1,e_2) = \begin{cases} \sqrt{\propto (e_1t_e - e_2t_e)^2 + \beta(e_1t_s - e_2t_s)^2} & ; \ e_1v = e_2v \\ \sqrt{\alpha(e_1t_e - e_2t_e)^2 + \beta(e_1t_s - e_2t_s)^2 + 1} & ; \ e_1v \neq e_2v \end{cases}$$

Where, $\alpha = \dfrac{\mathrm{var}(t_e)}{mean(t_e)}$ $\beta = \dfrac{\mathrm{var}(t_s)}{mean(t_s)}$

4. Experimental Datasets

There are no public datasets for RFID trajectories available. In these experiments, the CENTRE [15] trajectory generator is altered 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, the algorithm is modified to generate spatial value single-dimensionally, while generate observation time as two-dimensional points. The dataset sample and the DATA VIEW are as depicted in the fig 2 :

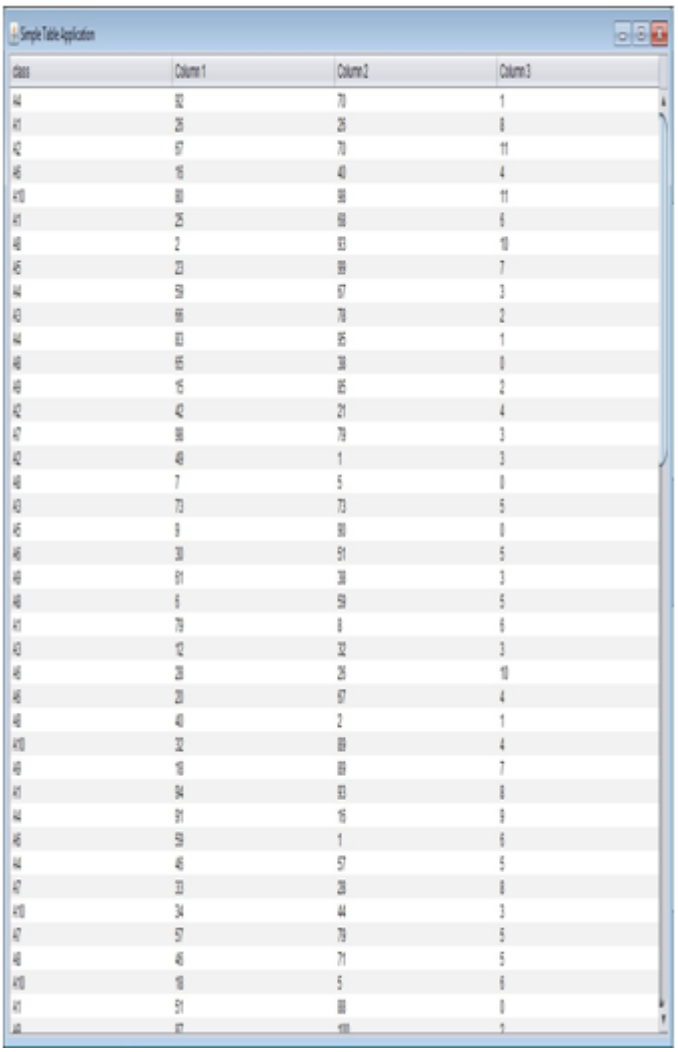

Figure 2: DATASET VIEW

5. Experimental Evaluation

Extensive experiments have been conducted to evaluate the performance of the proposed approach. This section focus on reporting four experimental results:
i) the quality of the trajectory clustering algorithm
ii) the performance of the clustering algorithm
iii) the effects of different distance functions
iv) the performance of the MapReduce-based parallelizing clustering.
All experiments were conducted on a Intel Pentium processor with 2GBytes of main memory, running on Windows 7. The proposed algorithms were implemented in Java netbeans 7.2.1

Software and Hardware Requirements

Software: Java Netbeans 7.2.1, Java version: JDK 7, OS: Windows 7, Hardware: Processor: Intel Pentium, RAM: 2GB, System type: 32-bit operating system.

6. Performance Evaluation

The performance of the RFID trajectory clustering algorithm is measured in terms of its accuracy. Table 1 shows the accuracy value of the existing and the proposed method. The accuracy of the proposed method and the existing method is high for cluster size of 90. For less number of cluster class, the accuracy is also minimum. But when compared to the existing method, the accuracy of the proposed method is high. The accuracy values for the various cluster sizes is shown in Table 1.

Table 1: Performance evaluation

Cluster size		90	80	70	60	50	40	30	20	10
Accuracy	Existing method	95.55	90.56	87.18	83.35	75.39	70.39	61.86	53.91	30.37
	Proposed method	95.55	91.45	88.42	84.70	76.29	72.36	63.08	58.14	34.42

The trajectory clustering algorithm is compared with two other algorithms in terms of quality. One is the Time-Focused Clustering (TFC) algorithm and the other is the Fuzzy C-Means (FCM). The variations of these two algorithms were implemented using the trajectory definition in this work.

Here the aim is to measure the clustering quality while varying the number of clusters in the datasets. Unfortunately, there is no well-defined measure for density-based clustering methods. So, a simple quality measure is defined for the analysis. In particular, the Sum of Squared Error (SSE) is exploited to represent the quality of the clustering.

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

In this experiment, we compared our trajectory clustering algorithm with TED BASED algorithm in terms of quality. Our aim is to measure the clustering quality while varying the number of clusters in the datasets. We used 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. The GUI for the same is reflected in Fig 3.

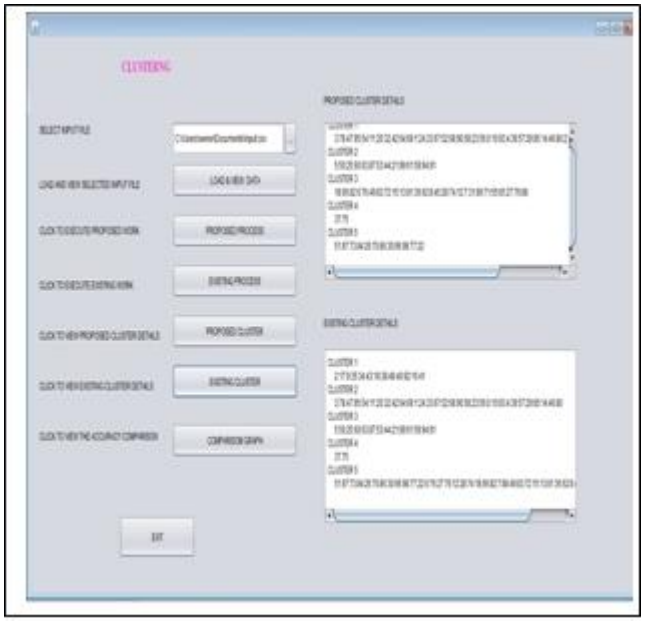

Figure 3: GUI

7. Results

The Experimental Studies of our clustering algorithm has 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. The accuracy graph for clustering quality is depicted in Fig 4 which demonstrates our technique superiority.

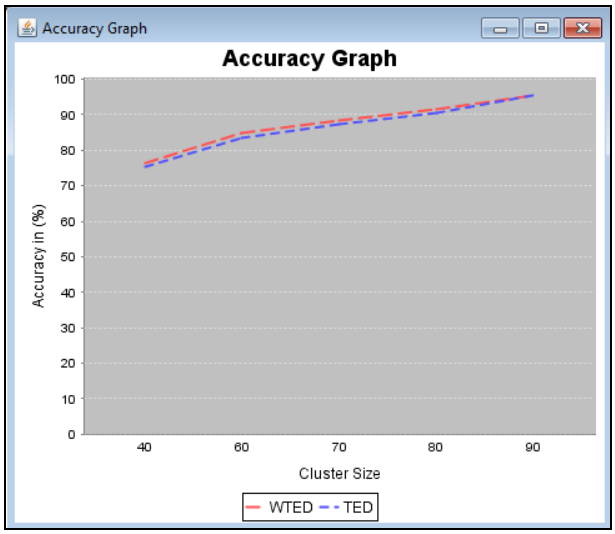

Figure 4: Accuracy Graph

8. 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 work, an efficient trajectory model is introduced and developed a novel clustering algorithm to cluster RFID trajectories with the capability to recover missing readings. The proposed algorithm is scalable and efficient, outperforming the existing method as demonstrated by the results from extensive experimental studies.

References

[1] EPCGlobal, Inc. http://www.epcglobalinc.org/
[2] 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.
[3] Laurie Sullivan. RFID Implementation Challenges Persist, All This Time Later. Information Week, Oct 2012.
[4] New-S. R. Jeffery, et al.. A Pipelined Framework for Online Cleaning of Sensor Data Streams. In ICDE. 2006
[5] New-A. Gupta et al.. Developing auto-id solutions using sun java system rfid software. Oct 2013.
[6] Yanbo Wu, Hong Shen, and Quan Z. Sheng, "A Cloud-friendly RFID Trajectory Clustering Algorithm in Uncertain Environments", IEEE transactions on parallel and distributed systems, Vol. 26, No. 8, pp. 2075-2088, 2014
[7] New- K. Finkenzeller, RFID Handbook: Fundamentals and Applications.
[8] KongfaHu, XiangqianXue, JianfeiGe, Yan Sun, Ling Chen "Dividing And Clustering Algorithms Of Moving Objects In Rfid Tracing System" Journal of Theoretical and Applied Information Technology E-ISSN: 1817-3195 2012. Vol. 45 No.1.
[9] Thanh T.L. Tran, LipingPeng, Boduo Li, YanleiDiao, 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
[10] New-Manage Data Successfully with RFID Anywhere Edge Processing http://www.ianywhere.com/developer/ , 2010
[11] New- J. Brusey, M. Harrison, C. Floerkemeier, and M. Fletcher, "Reasoning about uncertainty in location identification with rfid," 2003.
[12] Kongfa Hu and Long Li "Mining aNew Movement Pattern in RFID Database on Internet of Things"International Journal of Database Theory and Application Vol.7, No.2(2014), pp.37-44.
[13] SusantaSatpathy, Lokesh Sharma, Ajaya K. Akasapu, Netreshwari Sharma" Towards Mining Approaches for Trajectory Data"International Journal of Advances in Science and Technology Vol. 2, No.3, 2011.
[14] Akasapu A., Sharma L. K., Ramakrishan G. 2010. Efficient Trajectory Pattern Mining for both Sparse and Dense Dataset. Int. J. of Computer Applications (0975 - 8887)Volume 9– No.5. 45-48
[15] New- Daniel Dobkin and Steven Weigand. Tags vs. the World: HF and UHF Tags in non-ideal environments. WCA RFID SIG, June 2005.