SIMILARITY SEARCH FOR TRAJECTORIES OF RFID TAGS IN SUPPLY CHAIN TRAFFIC

Sabu Augustine[1] ,M.S Samuel[2] and Sajimon Abraham[3]

[1]Department of Mathematics, Marian College, Kuttikkanam, India

[2]Department of Computer Applications, MACFAST, Tiruvalla, India

[3]School of Management & Business Studies, M.G University, Kottayam, India

ABSTRACT

In this fast developing period the use of RFID have become more significant in many application domaindue to drastic cut down in the price of the RFID tags. This technology is evolving as a means of tracking objects and inventory items. One such diversified application domain is in Supply Chain Management where RFID is being applied as the manufacturers and distributers need to analyse product and logistic information in order to get the right quantity of products arriving at the right time to the right locations. Usually the RFID tag information collected from RFID readers is stored in remote database and the RFID data is being analyzed by querying data from this database based on path encoding method by the property of prime numbers. In this paper we propose an improved encoding scheme that encodes the flows of objects in RFID tag movement. A Trajectory of moving RFID tags consists of a sequence of tagsthat changes over time. With the integration of wireless communications and positioning technologies, the concept of Trajectory Database has become increasingly important, and has posed great challenges to the data mining community.The support of efficient trajectory similarity techniques is indisputably very important for the quality of data analysis tasks in Supply Chain Traffic which will enable similar product movements.

KEYWORDS

Radio Frequency Identifier, Supply Chain, Path Encoding Scheme, Chinese Remainder Theorem, Trajectory Similarity

1. INTRODUCTION

Radio Frequency Identification (RFID) is a technology that can be used to identify, track, detect or sort large number of products moving in a chain of supply [1, 2]. The applications of RFID have become more important and diversified in recent years due to the lower cost of RFID tags and smaller tag sizes. However there are many challenges that RFID data management requires as it has to record data from multiple RFID readers, which comes data in petabytes.

In this paper, we propose a model to represent an optimized storage structure RFID information for further analysis and interpretation. Using this model we suggest a similarity search algorithm which uses the spatial (location) and time concepts of the product movement in supply chain based on the trajectory of their RFID tags.

2. RFID DATABASE MODELING

We design an RFID database model based on the concept given in [13], and implement an optimized and scalable storage system for RFID data to support mining useful information. The proposed RFID system consists of three main distinct objects: tags, antennas and readers. A tag represented as T in a location reflects (if T is a passive tag) or emits (if T is an active tag) Radio

DOI: 10.5121/ijdms.2016.8302

frequency signals within its detection area. When the antenna detects the signal, the reader reads signal and stores the unique Tag id(EPC) and the current timestamp into the RFID database. We choose paths, tags, times and other information in the model and develop path coding schemes for the movement of T in a stored data model. With this we incorporate external information like logistic hierarchies and relevant product information into the product data model and the logistic hierarchy data model respectively, as shown in Figure 1.

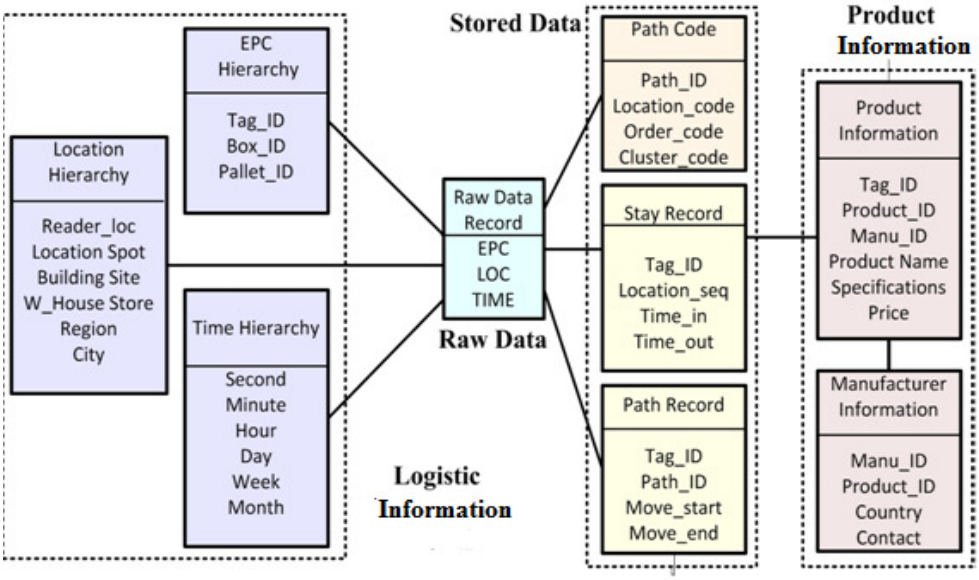

Figure. 1. RFID Database Model with External Information

To narrate the database model shown in Figure 1, as follows. An RFID reader identifies a tag and add a newRaw Data Recordusing three attributes (EPC, LOC, TIME) where EPC is a unique ID of a tag and LOC is the location of the (unique) RFID reader and TIME is the time instant of detecting the tag. For optimized storage we then convert the raw data into the Stay Recordformat which is having four attributes (tag-id, loc-code, time-in, time-out) where time-in and time-out are the time instants of the first detection and the final detection of the tag at loc-code. A stay record represents an RFID tag moving through a particular location in a specific time interval. However, using a stay record as a stored data model is simple to support the evaluation of path queries that track objects, particularly for those path queries involving many locations, which need to perform self-joining of the table many times. This will be more time consuming and may show poor search time performance. Thus, we develop various sophisticated coding techniques and include Path Record and Path Codeentities into the model. We also keep two external data structures one for logistic information and another for product information. These conceptual data structures will be used for linking external information with stored data structures for more meaningful information.

2.1 Path Encoding Scheme

We use an encoding scheme from mathematics known as path encoding scheme to have efficient storage structure optimization. The general idea discussed in [3] is about an encoding schemefor RFID tag movement where unique prime numbers q1,q0 are being used to represent all locations and order respectively. To generate a unique integer pair (Pl, Po), we rely on the *Unique Factorization Theorem* for coding locations and the *Chinese Remainder Theorem* for coding their order. However, two major difficulties in this kind of representation is that, the state-of-the-art

method in [3] which supports neither long path coding (e.g. encode a long path of more than 8 locations) nor cyclic path coding (e.g. encode a path in which a tag passed a location twice). The first problem is because of the fact that most programming languages use unsigned integers (32 bits)that only support $2^{32}-1$, which is less than the product of the first nine prime numbers$2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17 \times 19 \times 23$. Even for 64 bit unsigned integers (2^{64}-1),it can only support the first 15 prime numbers. In the case of path "L1→ L2→ L3 → L1 → L2 → L3", the method discussed in [3] fails because of the fact that the simultaneouscongruence's of Chinese Remainder Theorem is not applicable.

2.2 Problem with long path

We quote the following theorems from theory of numbers[4] in addressing the above problem

a. ***The Fundamental Theorem of Arithmetic:*** *Any natural number greater than 1 isuniquely expressed by the product of prime numbers.*

b. ***The Chinese Remainder Theorem:*** *Suppose we want to solve a system of congruence to different moduli*
$\boldsymbol{x} \equiv a_1 \pmod{p_1}$
$\boldsymbol{x} \equiv a_2 \pmod{p_2}$
……..
$x \equiv a_r \pmod{p_r}$
where$p_1, p_2, \ldots, p_r$ are positive integers such that $\gcd(p_i, p_j)=1$ for $i \neq j$
Then there exist a simultaneous solution x to all of the congruences and any two solutions are congruent to one another modulo P, where $P=p_1, p_2, \ldots, p_r$.

c. ***Euler Formula for Prime Generation:*** *For every integer* x *between* 0 *and* 40,$x^2 - x + 41$ *is a prime number.*

To solve the long path coding problem, we first partitionthe whole set of locationsinto different clusters having roughly the same number of locations. Using finite continuedfraction in Theorem 1 we are able to represent a cluster code denoted as C (havinga unique positive integer as its id) together with its respective (Pl, Po). Suppose there are two clusters coded by two positive integers C1 and C2. The sub path in cluster 1 can be computed as loc_code1 and order code 1 and similarly notations for the sub path in cluster 2. If a path goes from cluster 1 to cluster 2, we generate theFull path code P as

1/(c1+1/(loc_code1+1/(order_code1+1(c2+1/(loc_code2+1(order_code2))))))

When decoding P, we first check whether it is smaller than 1. If this is the case, then the path covers more than one cluster. We then decompress P to extract loc_code and order_code in each cluster.

We propose the following algorithms[3]to represent the above operations which will leads to storage structure optimization of RFID tags

2.3 Algorithm for Path Encoding

==========

Algorithm Encoding(P,F_i)
Input: A fullpathP, Fundamental Location Set F_i for each cluster c_i

Output: Fullpath code P

==========

Assign a positive integer $n_i$to each cluster c_i
Foreach cluster c_i**do**
Ifthere are repeated Stay Records in c_i**then**
Encode these Repeated locations by Euler Theorem
Update $E_i$by including the Euler's prime numbers
end if
loc code := Product of all prime number in Fi ∪Ei
order code := Output by using Fi ∪Eiin the Chinese remainder theorem
end for
fullpath code := Result of a finite continued fraction of {ni, loc code, order code } for all ci

==========

The idea of the decoding process is as follows. First, we decompress the full path code (if it is found to be larger than 1) to identify clusters and for each cluster C decompose loc_code into its corresponding list of all locations p. Second, we know whether there is a cycle in an encoded path by comparing p with F. If there are cycles, after sorting p, we decode those prime numbers that are not in F by reversing Euler Theorem to get their original set of prime numbers. Finally, the path can be constructed by sorting p by using order_code.

2.4 Algorithm for Path Decoding

==========

Algorithm Decoding(P,F_i)
Input: A fullpath code P, Fundamental Location Set F_i for each cluster c_i
Output: A full path code defined for all path segment p_i in the clusters

==========

The input P has to be decomposed into the ordered set in each cluster with a loc_code and order code

Foreach cluster c_i where is the order of the decompressed integer sequence from P
do
forall np≤loc_code**do**
ifthe remainder for dividing loc_code by npis 0) **then**
pi=pi+npwith an order
end if
end for
for all np belongs to p_i**do**
R(Reminder set):=R ∪{reminder after dividing order code by np}
end for
Based on the order in R Sort p_i
Ifcyclesare existing**then**
Apply Euler Theorem (Inverse) to all npin (p_i– F_i)
end if
end for

==========

3. SIMILARITY SEARCH

Similarity search is a process to find matching objects in the database which are similar to an input object supplied by the user. This section will start with a discussion on the different aspects of similarity search. The first important aspect is the concept of similarity itself. In the literature, two concepts of similarity have been applied successfully. The following sub sections present the two concepts of similarity measure in the literature [12] which arefeature vector approachand the concept of distance-based similarity.

3.1 The Feature Vector Approach

In feature vector approach the object is being described by a set of singe-valued object features. Based on these features the object is being represented in the feature space as point objects. The basic idea is illustrated in figure 2. Several application like medical imaging [6] and protein similarity has been successfully applied using this concept.

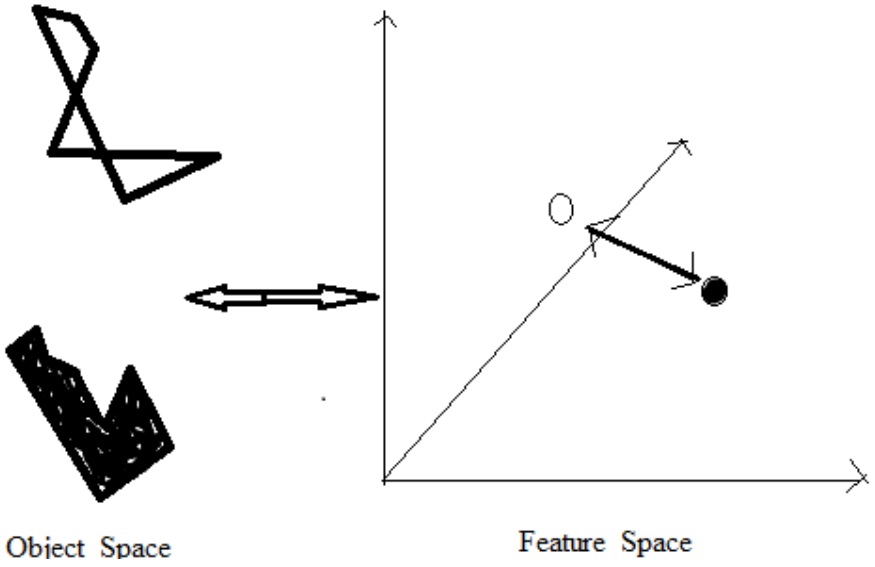

Figure 2 Similarity based on the feature vector approach.

Several measures are available to determine the distance between two points in the feature space like Manhattan distance and Euclidean distance etc. Most often it is a variant of the Lp-norms, which are explained below.

We use the following Euclidian norm to represent the length of a vector $x = (x_1, x_2, \ldots, x_n)$

$$\|x\|_2 = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{1/2}$$

If p is a real number ,$p \geq 1$, define the l_p norm of x by

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \cdots - |x_n|^p\right)^{1/p}$$

Now the distance between two vector x and y is defined by the function

$$d_p(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

The above distance measure is a metric and is known as $L_P$distance.

Definition 3.1: (L_p-norms) We define L_p-norms based on two assumed vectors one represented as $x = (x_1, \ldots, x_n)$, x€R^n, and other by $y = (y_1, \ldots, y_n)$, y €R^n. The L_p-norms between x and y are defined as:

$$L_p(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p) \quad ^{1/p}$$

As we know For p = 1 the norms is the Manhattan distance and for p = 2 it is the Euclidean distance..

3.2 Distance-Based Similarity

The concept of distance-based similarity is illustrated in Figure 3.

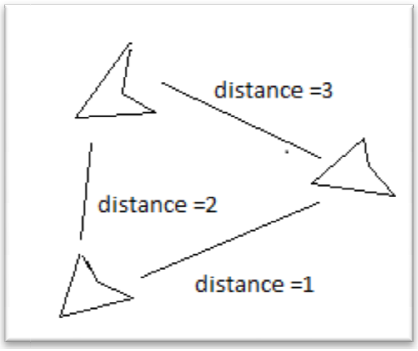

Figure 3 Distance-based Similarity

From the figure it is visible that the method is adding more flexibility by adding more complexity. Hence to ensure efficiency of the search process, careful treatment has to be made for computational complexity.

In the similarity measure, the great flexibility of the distance-based approach is founded in the only restriction which is positivity. These restrictions can be summarized by demanding the measure to satisfy the four metric properties as shown below:

i. Positivity Property: ¥x, y : $d_{sim}(x, y) \geq 0$
ii. Definiteness Property: ¥x, y : $d_{sim}(x, y) = 0$,for x = y
iii. Symmetry Property: ¥x, y : $d_{sim}(x, y) = d_{sim}(y, x)$
iv. Triangular inequality Property: ¥x, y, z : $d_{sim}(x, z) \leq d_{sim}(x, y) + d_{sim}(y, z)$

The positivity property shows that the when the distance is smaller, higher is the similarity. The symmetry property is explaining the concept that the objects which are mutually similar. The concept that no object can be very similar to two very dissimilar objects at the same time is expressed by the triangular inequality property.

3.3 Efficient Similarity Search

The size of modern databases due to the requirement of storing multi-dimensional data and the complexity of the similarity searching task make efficiency as an important issue for any similarity search application [14]. In this section, we will present two techniques to speed up the query processing in similarity search applications. The two techniques, the use of ***index structures***, and the use of a ***multi-step query processing architecture***, are not meant to be mutually exclusive. Instead, they can both be applied in parallel or at different stages of the query processing. The similarity methods proposed in this thesis are based on multi-step query processing architecture.

3.3.1 Index Structures

The use of standard index structures like ISAM, Hash tables, B-tree etc are being used to improve query processing efficiency in database systems. Numerous other index like structures like R-tree, have been proposed for many different data types and applications. The two type of index structures which are important for similarity search in structured data are: index structures for high-dimensional vector spaces and for metric spaces. When the similarity feature is based on vector space approach the first category is being used and the second is useful whenever the distance based similarity approach is followed. An overview over the existing approaches for indexing metric spaces is given in [7] and its variants M-tree [8]are specifically designed to allow dynamic updates.

3.3.2 Multi-step Query Processing

As shown in figure 4, a multi-step query processing [9] is performed in two or more steps. The filter step returns a number of candidate objects from the database which actually eliminate all unwanted objects based on a similarity measure. For those candidate objects, the exact similarity distance is then determined in the refinement step and the objects fulfilling the query predicate are reported as the result. To reduce the overall search time, the filter step has to fulfill certain constraints.

Figure 4 The Filter and Refinement step in multi-step query processing

To achieve completeness we should ensure that we eliminate unwanted objects in the filter step and at the same time not to filter out eligible objects. We will ensure that no false drop occurs during the filter step.

3.4 Requirements for Similarity Measures

In the preceding sections, we discussed several aspects of similarity search applications. From those discussions, we can now derive a few requirements as listed below which a similarity measure for structured data should fulfill.

i. The similarity measure should be adaptable to specific of the users.
ii. A clear explanation of the similarity distance formula has to be given.
iii. To apply today's large and fast growing database searches the query processing efficiency and moderate time complexity has to be ensured. The final requirement is

concerned with the efficiency of the query processing in similarity search applications.

3.5 Finding Similar Trajectories

Most of the trajectory similarity methods consider only spatial similarity in measuring the similarity between moving object trajectories. For example, let us assume that the number of common locationsproduct movement, visited by the trajectories is considered to be a spatial measure in finding similarity. The concept of spatio-temporal similarity is ensuring that the two trajectories have to pass through the same points at same time intervals. But in the above case two trajectories are similar spatially. Therefore we have concluded that to have actual similarity we have to consider both spatial and temporal similarity together.

Definition 3.1: Trajectory definition based on path encoding scheme

Let S be a set of trajectories in a Supply Chain traffic network, in which each trajectory is represented as

$$L_1(s_1,e_1) \rightarrow L_2(s_2,e_2) \rightarrow \cdot \cdot Li(s_i,e_i) \cdot \rightarrow L_n(s_n,e_n)$$

where n is the trajectory description length, L_i denotes a location in traffic path and s_i,$e_i$are the time instances (expressed in time units, e.g. seconds) that the moving tagbe with node L_i, and $s_i<e_i$ for each 1 <i<n. Also we assume that moving from a node to another comes at a non-zero cost.

As discussed in section 3.3.2, it is visible that since the number of trajectories are large, searching directly the similar trajectories in one step is tedious. We follow multi-step query processing concept with a filtering step followed by refinement. Filtering is the process of elimination of unwanted data and refinement is the process of tuning up the filtered candidates based on an accuracy threshold. We use spatial filtering followed by temporal refinement or temporal filtering followed by spatial refinement. We are interested in the movement of tags through selected locations as Point OfInterest (POI) or selected times as Time Of Interest (TOI).

First we filter the trajectories based on spatial similarity and then refine the result using temporal filtering. Here, since our objective is for spatio-Temporal similarity search, we propose the two step process for finding similar trajectories using a combined spatio-temporal measure which will be based on both POI and TOI.

Thus we identify the following similarity types related to RFID tag movement environment.

(i) Based on spatial filtering and then refinement using temporal distance
(ii) Based on temporal filtering and the refinement using spatial distance
(iii) Based on spatio-Temporal filtering and then refinement using spatio-Temporal distance

(i) Based on spatial filtering and then refinement using temporal distance:

Filtering is considered by spatial similarity, which is based on Point of Interest (POI). POI set may be a set of junction points in a RFID tag movement or location of some specific importance, and is usually decided by the user. This will be useful in finding out the movements of objects through known locations of interest, which may be terrorist locations, points of emergency, list of strategically important locations or famous tourist spots.

A threshold is introduced to spatial similarity measure to determine how similar the trajectories with reference to how many points it pass through and then this measure is used to filter based on

spatial distance. Here we further consider all the trajectories which confirm to the spatial threshold value which is fixed in advance.

Algorithm 1. Searching Similar Trajectories moving through certain Points of Interest based on Spatial Filtering and refinement using Temporal distance.
Input: Input trajectories TR_{IN}, spatial threshold ρ, temporal threshold δ,query trajectory tr_Q, POI set P; Output: similar trajectories TR_{OUT}

Begin
$TR_{Candidate}$= φ
TR_{OUT} = φ
n= number of Points in P
For each t_r in TR_{IN},
tr.k=0
For each p in P
If p is on $t_r$and in tr_Q*then*
tr.k = tr.k+1
End For
If (tr.k/n)> ρ *then*
$TR_{Candidate}$=$TR_{Candidate}$U{t_r}
End For
For each t_r Є $TR_{Candidate}$
If$dist_T(tr_Q, t_r, P) <δ$ *then*
TR_{OUT} = TR_{OUT} U{t_r}
End For
returnTR_{OUT}
End

Here k is a member variable stored along with each trajectory *tr* which will contain the percentage of similarity in matching with the number of points in P. This measure could be used later to cluster the similar trajectory to see how each trajectory is distant from the query trajectory. The spatial threshold ρ is an empirical value and has been fixed as per the accuracy requirement in spatial filtering. As the proposed measure finds percentage of common points visited by both the trajectories and since our experiments go for considering 10000 POI's we take the value of threshold ρ as 0.9 as we consider that the trajectories usually will not go for 10% or less points in POI.

(ii) **Based on temporal filtering and the refinement using spatial distance:**

It is usually a very rare chance of occurring such situation in practical case as finding distance between two time intervals is a meaningless process. However, it is interested to discuss the time intervals, TOI (Time of interest) as important characteristics in learning the moving behavior of moving objects on RFID tag movements. Here we consider the two trajectories are similar to each other, if they pass through the same points at the same TOI. Therefore, temporal similarity is defined based on TOI, where a TOI is the heaviest traffic time intervals on a specific RFID tag movement. The trajectories are filtered using this definition. A similar consideration is given for temporal similarity measure [10] which will consider two trajectories temporally similar with respect to a given set of TOI, only when both the trajectories alive in all the time points mentioned. This is modified by introducing a threshold δ to determine how similar the trajectories with reference to how many time points both the trajectories pass through.The temporal threshold δ is an empirical value and has been fixed as per the accuracy requirement in temporal filtering as already explained for spatial threshold-s

Algorithm 2.Objects moving through Certain Times of InterestSearching based on Temporal Filtering and refinement on Spatial Distance
Input:Input trajectory set TR_{IN}, threshold-s ρ, threshold-t δ , query trajectory tr_Q, TOI set T,
Output: Similar trajectory set TR_{OUT}

Begin
$TR_{Candidate} = \varphi$
$TR_{OUT} = \varphi$
nt= number of Time Points in T
For each t_r in TR_{IN}
$t_r.s=0$
For each t in T
If t is a time point in $t_r$andtr_Q*then*
$t_r.s = t_r.s + 1$
EndFor
If $(t_r.s/nt) > \delta$ *then*
$TR_{Candidate} = TR_{Candidate} \cup \{ t_r\}$
End for
For each $t_r \in TR_{Candidate}$
If$dist_S(tr_Q, t_r, T) < \rho$ *then*
$TR_{OUT} = TR_{OUT} \cup \{t_r\}$
End For
returnTR_{OUT}
End

Though the above method has theoretical relevance it has meagre practical application as usually initial filtering is based on location or spatial features. A drawback of this method is that the many unwanted trajectories are selected from trajectory data set by the process of temporal filtering. For example, if the time interval of the referral trajectory is much smaller than the total time interval for all moving objects, we can find that most of the trajectories are selected from the given trajectory data set. Therefore a combined measure of both spatial and temporal similarity will be more accurate and useful in terms of practical application which is discussed in the next session.

(iii)Based on spatio-Temporal filtering and then refinement using spatio-Temporal distance

Instead of the procedure followed in previous methods where we follow two stage process with initial filtering and then refinement, in this method, a combined spatial and temporal similarity together in the filtering step is being applied. Afterwards, we refine similar trajectories using spatial and temporal distance based on POI and TOI.

Algorithm 3. Similarity Searching based on Spatio-Temporal Filter and Spatio-Temporal Distance Refinement
Input: Input trajectory set TR_{IN}, spatial threshold - ρ1, ρ2; temporal threshold- δ1, δ2; query trajectory tr_Q, POI set *P*, TOI set *T* .
Output: TR_{OUT} , the set of similar trajectories

Begin

$TR_{Candidate}=\varphi$
$TR_{OUT} = \varphi$
np= number of Locations in POI
nt= number of Time Points in TOI
For each t_r in TR_{IN},
tr.k=0;
For each p in P
If p is on $t_r$and in tr_Q*then*
$t_r.k=t_r.k+1$
End for
$t_r.s=0$
For each t in T
*If*t is a time point in $t_r$andtr_Q*then*
$t_r.s= t_r.s +1$
End For
If $(t_r.k/np)>\rho1$ and $(t_r.s/nt)>\delta1$ *then*
$TR_{Candidate}= TR_{Candidate}U\{t_r\}$
End for
For each $t_r \in TR_{Candidate}$
If $(dist_T(tr_Q,t_r,P) <\rho2)$ and $(dist_s(tr_Q,t_r,T) <\delta2)$ *then*
$TR_{OUT} =TR_{OUT} U\{t_r\}$
End For
returnTR_{OUT}
End

Advantages of threshold ρ1, δ1 are that we can see the degree of similarity if the trajectories does not passes through all points in POI and all time Points in TOI. This measure could be used later for trajectory clustering purpose.

In all the above algorithms the advantages with binary encoding is that the initial filtering can be done by using binary code component of the district or road. Thus a large number of trajectories could be pruned out at the initial stage itself when the query requirement is restricted to a district or road.

3.6 Experimental Evaluation

The data set used is a simulated RFID tag movement data set[11] generated from a simulated product movement in the supply chain of an on-line shopping site.All experiments have been conducted on a Intel Core i3 machine running Windows XP, with 4 GB of RAM, and a 320 GB hard disk.

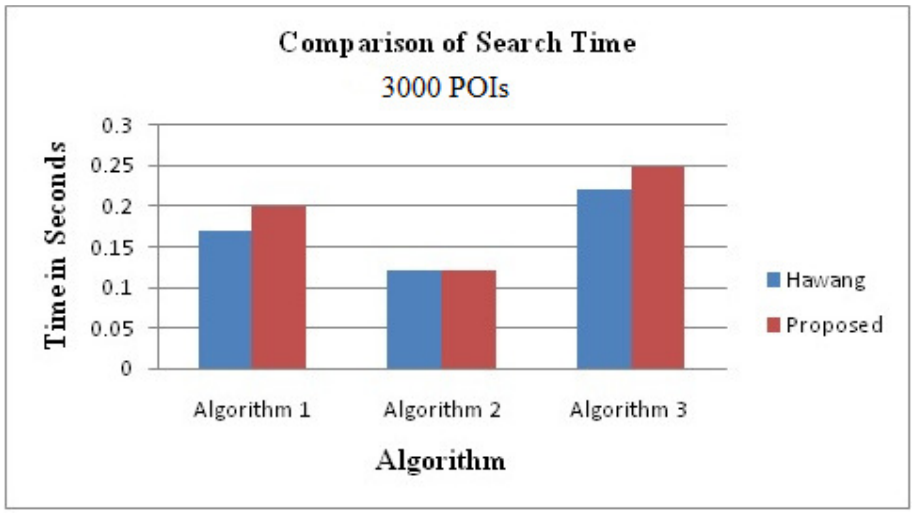

Figure 5. Search Time Performance for 1000 POI's

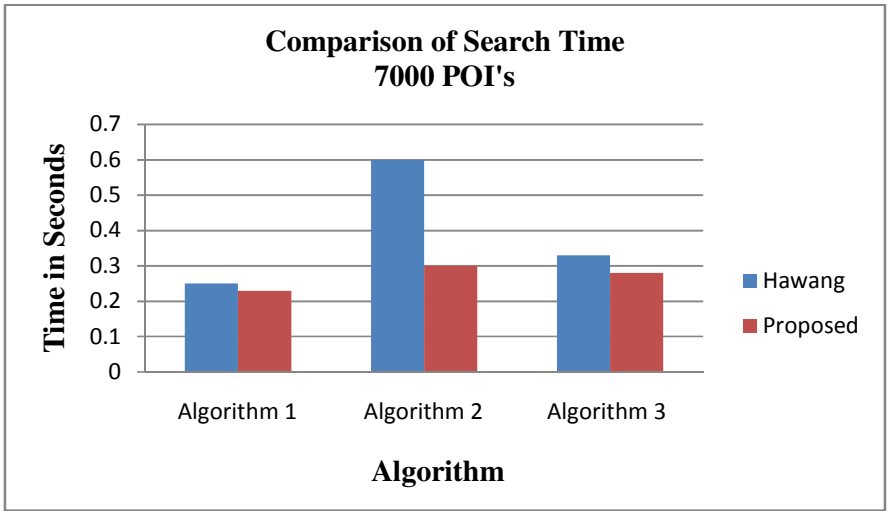

Figure 6. Search Time Performance for 7000 POI's

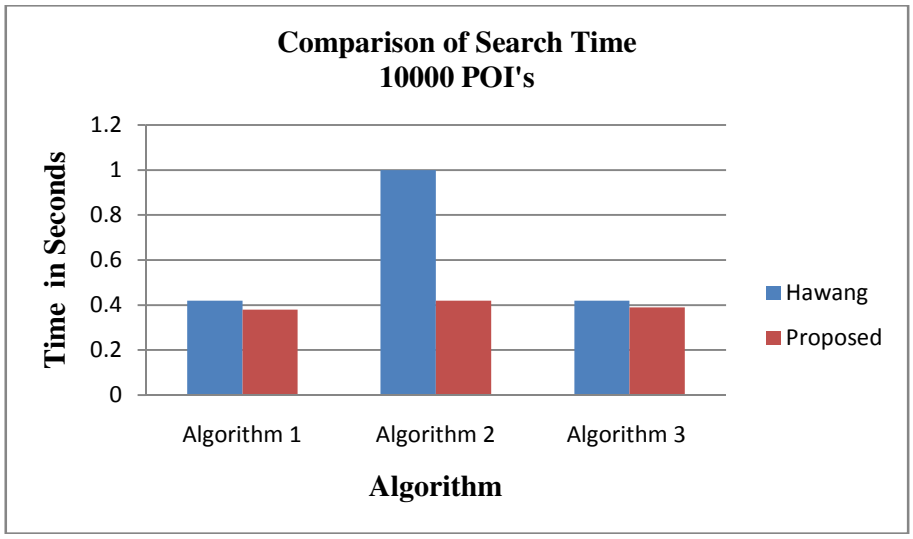

Figure 7. Evaluation of Search Time Performance for 10000 POI's

Comparison of Similarity Search Time

For comparison we have used the similar work done by Hawang [10] for different set of POI's . We have compared the search time in query processing in the above three methods with that discussed in Hawang. The results are shown in Figures 4,5 and 6. The method followed by Hawang[10] , for TOI involves so many unwanted trajectories at the filtering stage as the time interval of a query trajectory is smaller than the total life span for all moving tags. But in the above experiments one could prune out large number of unwanted trajectories when POI or TOI has taken in using the concept of path encoding. Even though, the search time as shown in Figure 4,is not showing little lower performance in two of our methods than that in [10], for higher number of POI's it is better than Hawang method. This is illustrated in Fig 5 and Fig 6. Also the additional thresholds used in the filtering step provide the possibility of clustering the trajectories for future data mining applications.

4. CONCLUSION

RFID technology has tremendous applications in the fast moving business domain of Supply Chain network where companieswant to track movements of physical items from the source of production center to the point of customer destination. This paper introduces methodology to save storage space and thereby improving the search performance by using a concept in fundamental theorem of arithmetic and properties of prime numbers. The effective representation of RFID tags in database and the similarity methods applying to trajectories of moving tags in a supply chain environment has many applications like trespassers and theft identification, detection of undue delay in traffic due to traffic congestion that will eventually open up alternate re-routing possibilities. The paper has also found that the spatio-temporal similarity measure is more realistic and suggested a measure for comparing the existing methods in an RFID tag movement environment. As a continuation work we are planning to use these measures in different types of clustering algorithms that will have extensive applications in the segregation of objects with specific moving characteristics.

REFERENCES

[1] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. RFID Data Processing With a Data Stream Query Language. In: Proc. of ICDE, 2007.

[2] Palmer M, Principles of Effective RFID data management, Enterprise System, March 2004.

[3] C-H Lee, C-W Chung, Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID, In: Proc. of SIGMOD 2008.

[4] Hardy, G. H.; Wright, E. M. , An Introduction to the Theory of Numbers, 1979

[5] R. Sedgewick, P. Flajolet, An Introduction to the Analysis of Algorithms.

[6] Flip Korn., Nikolaos Sidiropoulos., Christos Faloutsos., Eliot Siegel. and Zenon rotopapas. Fast and effective retrieval of medical tumor shapes. IEEE Transactions on Knowledge and Data Engineering, 10(6), 1998, 889– 904.

[7] Chavez, E., Navarro, G., Baeza-Yates, R. and. Marroquin, J.L. Searching in metric spaces. ACM Computing Surveys, 33(3), 2001, 273–321.

[8] Paolo Ciaccia., Marco Patella., and Pavel Zezula. Mtree: An efficient access method for similarity search in metric spaces. In Proceedings of 23rd International Conference on Very Large Databases, Athens, Greece, 1997, 426–435.

[9] Papadopoulos, A. and Manolopoulos, Y. Structure based similarity search with graph histograms. In Proceedings of International Workshop on Similarity Search, IEEE Computer Society, 1999, 174–178.

[10] Jung-Rae Hwang., Hye-Young Kang2. and Ki-Joune Li2. Searching for Similar Trajectories on RFID tag movements using Spatio-Temporal Similarity. Book on Lecturer Series in Computer Science ,Volume 4152, Springer-Verlag, 2006, 282-295.

[11] F. Wang and P. Liu. Temporal Management of RFID Data. In: Proc. of VLDB, pages 1128-1139, 2005.

[12] Flip Korn., Nikolaos Sidiropoulos., Christos Faloutsos., Eliot Siegel. and Zenon rotopapas. Fast and effective retrieval of medical tumor shapes. IEEE Transactions on Knowledge and Data Engineering, 10(6), 1998, 889– 904.

[13] Wilfred Ng, Developing RFID Database models for Anaysisng Moving Tags Supply Chain Management , Lecture Notes in Computer Science, Version 6998, 2011, p 206-218.

[14] Sajimon Abraham, Sojan Lal P., 2010. Trajectory similarity of Network Constrained Moving Objects and Applications to Traffic Security, Pacific Asia International Workshop On Security Informatics(PAISI 2010) held in 21 June , Hyderabad , India, LNCS 6122, pp. 31–43, Springer-Verlag Berlin Heidelberg.