Frequent Pattern Mining of Trajectory Coordinates using Apriori Algorithm

Arthur.A.Shaw
Research Scholar
National Institute of Technology
Thiruchirappalli-620015, India

N.P. Gopalan
Professor
National Institute of Technology
Thiruchirappalli-620015, India

ABSTRACT
Frequent pattern mining has been an emerging and active field in data mining research for over a decade. Abundant literature has been emerged from this research and tremendous progress has been made in numerous research frontiers. This article, provide an application of the modified Apriori algorithm in coordinate sets of trajectories to find the frequent trajectory coordinates. In this algorithm additional steps are added to prune the coordinate sets generated so that to reduce the unnecessary search time and space. This sequential pattern mining method is quite simple in nature but complex to implement. This paper explains the basics of data origination, database structure to hold the coordinate datasets and the implementation of the algorithm with the object oriented programming language by an illustration. It can be applied to interesting game domains to find the frequent trajectory of an object shot by a player which follows a trajectory path.

General Terms
Data Mining, Pattern Mining, Algorithms et. al.

Keywords
Data mining, Association mining, Frequent pattern mining, and trajectory pattern mining.

1. INTRODUCTION
Frequent trajectory coordinates play an essential role in many data mining tasks that try to find interesting patterns from trajectory databases consists coordinates of objects. With the advancements in Science and Communication technology, sensors and control equipments data accrual of moving objects is made so easy. From these huge volumes of data, knowledge about the frequent trajectory pattern can be generated and observed [1]. This can be applied in different domains to predict the further movements of the objects. Finding frequent patterns plays an important role in mining associations, correlations, and many other interesting relationships among data.

Frequent pattern mining was first proposed by Agrawal et al [2] [3] for market basket analysis in the form of association rule mining. It analyses customer buying habits by finding associations between the different items that customers place in their "shopping baskets". Since its inception, hundreds of new algorithms or improvements on the existing algorithms have been evolved to solve the mining problems more efficiently as per the user requirements. The spatial and temporal attributes are considered simultaneously for mining trajectory patterns in Anthony et al [4]. Based on this and using association mining, frequent paths of moving objects are extracted. This may be of immense help in predicting the paths frequented by moving objects. Similarly frequent paths of flights, roads of commuters and migratory birds may be investigated. More precisely this concept can be applied in case of some games, to find the path of a frequent ball movement. That is get the coordinates of the ball moved out from the bat of a player X to a destination. These coordinates can be collected for player X for a set of games. Using this coordinates, discover a frequent trajectory pattern will give a most frequent path of the ball movement for player X. Based on this knowledge the opponent player or team get adjust to it; tackle the situation face the ball from player X and have more probability to win. Similar concept is adopted in [5], [6] but by different complex approach.

The paper is organized as follows: Related works in the Apriori algorithm, frequent pattern mining and frequent trajectory mining are reviewed in Section 2; Preliminaries and problem description based on the proposed modified Apriori algorithm based method to find frequent trajectory patterns are described in Section 3; An example is given to illustrate the proposed method is in Section 4. Experimental results for showing the performance of the proposed method are provided in Section 5. Conclusions and future work are given in Section 6. Acknowledgements and references are given in Section 7 and 8 respectively.

2. RELATED WORKS
This work is mostly related to pattern discovery from sequential data, which include time series, event sequences, and spatio-temporal trajectories except time component.

Mining frequent itemsets is one of the fundamental problems in the active research area of data mining. An itemset is considered to be frequent if its support is greater than a user-specified minimum support threshold, where the support of an itemset is defined as the percentage of transactions in the database that contain the itemset. Agrawal et al [3] pioneered to mine frequent itemsets from transactional databases and proposed an Apriori approach. After this approach many other itemset mining algorithms [6]-[9] have been proposed for mining the frequent itemsets in a database. Sequential pattern mining algorithms [10]-[14] have also been proposed. Many graph mining algorithms [15]–[21] have been proposed. FFSM [16] exploits an algebraic graph framework called canonical adjacency matrix (CAM) tree to perform join-extension operations to unambiguously enumerate all frequent subgraphs and avoid subgraph isomorphism testing by maintaining an embedding set for each frequent subgraph. AGM [18] and FSG [20] use an Apriori-based approach to combine frequent subgraphs mined at the previous level to generate all candidates at the next level. The gSpan [21] generates the frequent subgraphs without candidate generation in a depth-first search manner. Yun [22] efficiently mine sequential patterns in large sequence databases by using the weight constraints. Garofalakis et al. [23] proposed an approach, called SPIRIT, to mine sequential patterns with regular expression constraints. All these, sequential pattern, itemset, and graph mining methods are not suitable for mining frequent trajectory patterns. The spatial and temporal attributes are considered simultaneously for mining trajectory patterns in certain approaches. Itemset and sequential pattern mining algorithms do not consider the spatial attribute, while the graph mining algorithms do not consider the temporal attribute instead uses the edge. For frequent pattern mining of trajectory coordinates in this approach consider the 2D coordinates (x, y).

Trajectory is defined as the path of flying object; the path that a projectile makes through space under the action of given forces such as thrust, wind, and gravity. It is also defined as the curve intersecting at constant angle; a curve or surface that intersects all of a family of curves or surfaces at a constant angle. Trajectory is a path of process or event; the way in which a process or event develops over a period of time. With the advancements in tracking the path of flying objects, a large amount of spatial-temporal data has been collected in databases. Getting knowledge from this data based on the observation of useful patterns has increased the attention of technocrats and decision makers recently.

Frequent behavior of moving objects was analyzed in the sequential pattern mining paradigm by Giannotti et al [24]. In this approach both space and time components are considered for mining trajectory patterns. The geometry of moving objects can be of any spatial type and is defined by a function from a temporal domain to a range of spatial values by Stefano et al [25]. Here the trajectory is poly-line connecting the sample points that define discrete representation of movement in a sequence of spatio-temporal segments. Components of a trajectory are defined when a trajectory starts, a spatio-temporal path it passes and ends. A graph based mining (GBM) for mining the frequent trajectory patterns in a spatio-temporal database is suggested by Lee et al [4]. This method scan the database to generate a mapping graph and trajectory information lists, from these it does a depth-first search manner to find all frequent trajectory patterns.

Finding the frequently occurring trajectory patterns will guide to analyze and predict the movement of objects. Frequent trajectory pattern mining is the process of discovering the most frequently occurring coordinates in the set of trajectories. Different trajectories may arise from different coordinates passing through different coordinates and ending at different coordinates. If all the trajectories are originating from the same coordinates passing through the same coordinates and ending at the same coordinates, then all such trajectories are frequent. But in reality, obviously it is not so. So to find the frequent trajectory pattern using data mining is nothing but finding the frequent coordinates of the trajectories which met the user defined threshold value. After the mining process the output will be a set of frequently occurring coordinates of the trajectories.

3. PRELIMINARIES AND PROBLEM DESCRIPTION

The main objective of frequent pattern mining is to find all frequent trajectory patterns in a database with respect to the user specified minimum support threshold. Initially the coordinate data set is in the form of text file consists of trajectory name and its corresponding coordinates in 2D, that is x and y values. These values are extracted from the text file and stored in a normalized relational database table consists of columns to hold trajectory number or identifier, coordinate number, x and y coordinates. There is a master table to hold the trajectory number and trajectory name. A coordinate set is a non-empty set of coordinates $\{(x_1,y_1), (x_2,y_2),\ldots,(x_n,y_n)\}$ where n is the maximum number of coordinates in the set. A coordinate sequence is an ordered list of coordinate set. For a same sequence number there will not be more than one coordinate. The sequence number is the contiguous integer. Initial processing like generating the possible one coordinates sets and their corresponding support count is found out by the programming procedural capability of the RDBMS. This will generate two output text files consists of the actual trajectory coordinates and one coordinate sets whose support count is not less than the user specified minimum support threshold. Using these files as input to an object oriented language will implement the Apriori algorithm and generate the output of a set of frequently occurring coordinates of the trajectories.

The modified Apriori algorithm to find frequent trajectory pattern mining is based on the sequential pattern mining framework. Trajectory is considered to be tracking the paths of an object. A trajectory can be represented as a set of coordinates in sequence in Euclidian space. Trajectories are considered to be tracking the paths $(P_1,P_2, \ldots, P_n)$ of flying objects $(O_1,O_2, \ldots, O_n)$, a large amount of spatial data has been collected in a database D. Given a set of trajectories $D = \{P_1,P_2, \ldots, P_n\}$, n is the number of paths of the flying object. Each trajectory path P consists of set of coordinates $\{V_1, V_2, \ldots, V_k\}$, $(1 \leq k \leq n)$ and n is the number of coordinate set, while considering the two dimension Euclidian space. Here k varies from 1 to n and is represented as $(1 \leq k \leq n)$ where k is denoted as the length of the trajectory and n is the number of coordinates in it. In other words k-coordinate set is a set of coordinates having number of coordinates n where k = n. Each coordinate in a vertex V is represented in units in X-axis and Y-axis as (x,y). A sub-trajectory path p consists of set of coordinates $\{v_1, v_2, \ldots, v_k\}$, $(1 \leq k \leq n)$ and n is the number of coordinate set. A sub-sequence coordinates set $s = \{c_1, c_2, \ldots, c_k\}$ may contain in different set of trajectory paths. The set s is maximal if it is not contained in any other trajectory path P. Hence the trajectory path P of a flying object O consists of set of coordinates $\{v_1, v_2, \ldots, v_k\}$ arranged in increasing sequence number. Such a sequence set of coordinates is called the trajectory path sequence. The support of a coordinate v is the percentage of coordinates in D that support the coordinate v. It is defined as the fraction of all the trajectories which support this coordinate. It can be written as,

Support(**v**) = Number of coordinates v in D / Total number of trajectories in D

A trajectory path or sub-trajectory path is frequent if its support exceeds the user-specified minimum support threshold value ξ. Such a set of coordinates is the large coordinate set which is also termed as frequent trajectory pattern.

Let C_k be the set of k-coordinates with each member of this set has two fields (i) coordinate set and (ii) support count. Let L_k be the set of large k-coordinate sets with each member of this set has two fields (i) coordinate set and (ii) support count. This algorithm initially scan the database D to find the distinct coordinates, to form a set C_1. Let this set be $C_1 = \{(x_1,y_1), (x_2,y_2),\ldots,(x_m,y_m)\}$, where m is the distinct coordinates in D. Then scan the database D to find the support count of each coordinates in C_1. From this set consider the coordinates whose support is greater than the user-specified minimum support threshold value ξ. Let the coordinates considered be L_1. Idea is that if the trajectory coordinate set L is frequent if and only if its subsequences L_{k-1} are frequent. This step is the first step of the algorithm.

Beginning of the second step is the generation of 2-coordinate candidate set $C_2 = L_1 \times L_1$. That will be $\{(x_1,y_1)\ (x_2,y_2),\ (x_1,y_1)\ (x_3,y_3),\ \ldots,\ (x_1,y_1)\ (x_m,y_m),\ (x_2,y_2)\ (x_3,y_3),\ (x_2,y_2)\ (x_4,y_4),\ \ldots,\ (x_2,y_2)\ (x_m,y_m),\ \ldots,\ (x_{m-1},y_{m-1})\ (x_m,y_m)\ \}$, where m denotes the number of coordinates in 1-coordinate set C_1. Simultaneously do the pruning operation by verifying a 2-coordinate candidate generated is already exists in C_2 and do not consider a 2-coordinate candidate generated is of same coordinates appearing more than once, that is $(x_1,y_1)\ (x_1,y_1)$. Again scan the database D to find the support count of each coordinates in C_2. From this set consider the coordinates whose support is greater than the user-specified minimum support threshold value ξ. Let this coordinates considered be L_2. Join all the coordinates in the set and this set is the frequent 2-coordinate sets of the trajectories in D. Similarly continue the process until k+1 times finds a set of candidates generated is empty set. Then return all the frequent trajectory coordinate sets joined in the k^{th} step. This output set is the set of frequent trajectory pattern.

3.1 Modified Apriori Algorithm – An Efficient and Scalable Method for Mining Frequent Trajectory Patterns

The Apriori Algorithm is an influential algorithm for mining frequent trajectory coordinates sets. This modified Apriori algorithm consists of two phases. As mentioned in the first paragraph of this section the frequent 1-coordinate set L_1 obtained in a text file format is given as input to the algorithm. Also the second file which consists of the coordinates of the trajectories is used for searching for a pattern and counting the support of the k-coordinate sets is used as input to this algorithm. This process will be the first phase of the algorithm. The second phase starts from candidate generation, pruning and match the coordinate for a specific pattern generated by reading the data file and find the support count are all memory resident. Then consider only the coordinates whose support count is greater than or equal to the user specified minimum support threshold value ξ. The output obtained at the end of this phase is the frequent trajectory pattern.

Key Concepts :

•Frequent Coordinate sets: Coordinate sets whose support count is not less than the minimum support value ξ, denoted by L_i for i^{th}-Coordinate set.

•Apriori Property: Any subset of frequent coordinate set must be frequent.

• Prune Operation: Deleting the generated candidate coordinates which are not required for further processing.

•Join Operation ($\cup_k L_k$): To find L_k, a set of candidate k-coordinate set is generated by joining L_{k-1} with itself.

Input: A trajectory of 1-coordinate sets data file and a minimum support value ξ. A data file which consists of the actual coordinates of the trajectories is used for searching for a pattern and counting the support of the k-coordinate sets.

Output: Frequent trajectory coordinates which satisfies the minimum support value ξ.

Method: Pseudo-code of modified Apriori algorithm :

```
$C_k$: Candidate coordinate set of size k

$L_k$: Frequent coordinate set of size k
$L_1$ = {frequent coordinate sets};
For (k = 2; $L_k$ != $\emptyset$; k++) do begin
   $C_k$ =  Generate_Candidate_Coordinates ($L_{k-1}$);
    For each trajectory coordinate sets in $C_k$ do
        Increment the support count of all candidate
        coordinates in $C_k$ that exists in the paths ($P_1$,$P_2$,
        …, $P_n$) in the data file;
        $L_k$ = Take only the candidate coordinates
        whose support count is greater than or equal to
        the user specified minimum support threshold
        value $\xi$ from $C_k$;
End
Return (Frequent trajectory coordinates = $\cup_k L_k$);

Generate_Candidate_Coordinates ($L_{k-1}$)
 A set of candidate k-coordinate sets is generated by
  joining $L_{k-1}$ with itself
   i.e., $C_k$ =  $L_{k-1} \times L_{k-1}$;
  Prune if the candidate k-coordinate set generated
  currently is already generated or else
  if the candidate coordinates generated currently
  consists of more than one same coordinate;
Return($C_k$);
```

4. AN EXAMPLE

Given below, is a simple example to illustrate the process of the modified Apriori Algorithm construction. Assume there are six trajectory transactions shown in Table 1. Here TID represents trajectory identifier. The trajectory column represents the two dimension coordinates of a trajectory. Normally a trajectory is represented by its coordinates along with its time component. For finding the frequent trajectory patterns the time component is not considered. More over for finding the frequent trajectory patterns the coordinates of a trajectory are enough.

Table 1

TID	Trajectory
T1	(1,1), (1,2), (2,2), (2,3), (3,4), (3,5)
T2	(1,1), (1,2), (2,2), (2,3), (3,4), (3,5)
T3	(1,3), (1,2), (2,2), (2,3), (3,4), (2,5), (1,5)
T4	(2,1), (2,2), (2,3), (3,4), (2,5)
T5	(1,1), (1,2), (2,2), (2,3), (3,4), (3,5)
T6	(2,1), (2,2), (2,3), (3,4), (2,5), (1,5)

The frequent one trajectory coordinate is constructed in the following way. Here support of a trajectory coordinate is the occurrence of the coordinate in all the trajectories. For example the support of coordinate (1,1) is 2/6. That is 33.33%. Initially assume the minimum support count is set at 50%. That is if a coordinate occur in 50% and more of all the trajectories is the user specified minimum support threshold value. In short in this example the coordinates whose support count is 3 and more will be considered. First, the trajectory database is scanned to find the support of 1-coordinate sets. Let this coordinate set be C_1. That is the occurrence of each coordinates in the trajectory database D. All the 1-coordinate sets in the trajectory data set D with their support count are shown in Table 2.

Table 2

1-Coordinate set	Support count
(1,1)	2
(1,2)	4
(2,2)	6
(2,3)	6
(3,4)	6
(3,5)	3
(1,3)	2
(2,5)	3
(1,5)	2
(2,1)	2

C1

From this coordinate support count are compared with the minimum support count and consider only the coordinates which satisfies the criteria. Let this 1-coordinate set be L_1 and is shown in Table 3.

Table 3

1-Coordinate set	Support count
(1,2)	4
(2,2)	6
(2,3)	6
(3,4)	6
(3,5)	3
(2,5)	3

L1

Up to this point the first phase of the algorithm is done, each coordinate in L_1 is a member of the candidate C_1. Two output files created based on the first phase of the algorithm are, (i)the frequent 1-coordinate set L_1 obtained in a text file format without support count and (ii) data file which consists of the coordinates of the trajectory paths (P_1,P_2, …, P_n).

Next step is the second phase of the algorithm starts by generation of 2-coordinate frequent pattern after reading the frequent 1-coordinate set L_1 from the text file. In this step generate the various possible 2-coordinates that can occur from the coordinate set L_1. Let this set be C_2. Do a pruning operation, which is from the set of 2-coordinates in C_2 check for the coordinates that are already in C_2 also for similar coordinates, that is (1,2), (1,2) . If there is any such coordinate combinations exists then delete the entry. For lack of space the actual 2-coordinate sets generated and 2-coordinate sets pruned are not shown.

Then read a 2-coordinate set from the set C_2 and open the data file and search for the coordinate set pattern. Here the coordinate set

Till now the second iteration of the second phase of algorithm is done, each coordinate in L_2 is a member of the candidate C_2. Next step is the generation of 3-coordinate frequent pattern. In this step generate the various possible 3 coordinates that can occur from the coordinate set L_2 and do a pruning operation as before. Let this set be C_3.

Then read a 2-coordinate set from the set C_2 and open the data file and search for the coordinate set pattern. Here the coordinate set pattern (1,2),(2,2) means for example, either (1,2),(2,2) or (2,2),(1,2). Both are similar patterns. If similar patterns are found then increment the support count of the corresponding each 2-coordinate set C_2 just generated. Continue the process until all the 2-coordinate sets are read and compared with the data file. This is shown in the Table 4.

Table 4

2-Coordinate set	Support count
(1,2),(2,2)	4
(1,2),(2,3)	4
(1,2),(3,4)	4
(1,2),(3,5)	3
(1,2),(2,5)	1
(2,2),(2,3)	6
(2,2),(3,4)	6
(2,2),(3,5)	3
(2,2),(2,5)	2
(2,3),(3,4)	6
(2,3),(3,5)	3
(2,3),(2,5)	3
(3,4),(3,5)	3
(3,4),(2,5)	3
(3,5),(2,5)	0

C2

Compare the support count of coordinates in C_2 with the minimum support count and consider only the coordinates which satisfies the criteria. Let this 2-coordinate set be L_2 and is shown in Table 5.

Table 5

2-Coordinate set	Support count
(1,2),(2,2)	4
(1,2),(2,3)	4
(1,2),(3,4)	4
(1,2),(3,5)	3
(2,2),(2,3)	6
(2,2),(3,4)	6
(2,2),(3,5)	3
(2,3),(3,4)	6
(2,3),(3,5)	3
(2,3),(2,5)	3
(3,4),(3,5)	3
(3,4),(2,5)	3

L2

Till now the second iteration of the second phase of algorithm is done, each coordinate in L_2 is a member of the candidate C_2. Next step is the generation of 3-coordinate frequent pattern. In this step generate the various possible 3 coordinates that can occur from the coordinate set L_2 and do a pruning operation as before. Let this set be C_3.

Then read a 3-coordinate set from the set C_3 and open the data file and search for the coordinate set pattern. Here the coordinate set pattern (1,2),(2,2),(2,3) means for example, either (1,2), (2,2),(2,3) or (1,2),(2,3),(2,2) or (2,2),(1,2),(2,3) or (2,3),(2,2),(1,2). All are similar patterns. If such similar patterns are found then increment the support count of the corresponding each 3-coordinate set C_3 just generated. Continue the process until all the 3-coordinate sets are read and compared with the data file. Compare the support count of coordinates in C_3 with the minimum support count and consider only the coordinates which satisfies the criteria. Let this 3-coordinate set be L_3 and is shown in Table 6.

Table 6

3-Coordinate set	Support count
(1,2),(2,2),(2,3)	4
(1,2),(2,2),(3,4)	4
(1,2),(2,2),(3,5)	3
(1,2),(2,3),(3,4)	4
(1,2),(2,3),(3,5)	3
(1,2),(3,4),(3,5)	3
(2,2),(2,3), (3,4)	6
(2,2),(2,3), (3,5)	3
(2,2),(2,3), (2,5)	3
(2,2),(3,4),(3,5)	3
(2,3),(3,4),(3,5)	3
(2,3),(3,4),(2,5)	3

L3

Table 7

5-Coordinate set	Support count
(1,2),(2,2),(2,3),(3,4),(3,5)	3

L5

The same process is continued and at the end of the fifth iteration the result obtained is shown in table 7 above. Next step is the generation of 6-coordinate frequent pattern. In this step generate the various possible 6-coordinates that can occur from the coordinate set L_5. Let this set be C_6. Since there is only one row in L_5, from that no candidate generation is possible. So the set C_6 will be null and that makes L_6 an empty set. This results the termination of for loop in the Apriori algorithm. Now the Apriori algorithm will return the set {(1,2), (2,2), (2,3), (3,4), (3,5)} as frequent trajectory coordinates.

5. EXPERIMENTAL RESULTS

Based on the modified Apriori Algorithm method of finding the frequent trajectory patterns experiments were made to find out the performance of the construction algorithm and the results were shown in the charts. Since data mining operations are performed in data warehouse, the processing performed is similar to batch processing. So in this case no frequent new or modified transaction will occur. The process we perform is exactly as mentioned in section 3.

The experiments were performed with the instructions of Net Beans IDE 6.7.1 with JavaFX on a PC having an Intel Pentium D CPU with a 3.40 GHz processor and 1 GB RAM and running the Microsoft Windows XP Professional version 2002 operating system. The dataset used in this experiment is the synthetic dataset obtained from Yi-An chen [4]. Each transaction in this dataset consisted of the x, y coordinates of the trajectories with time interval. There were 83 transactions with 2045 coordinates in the dataset. The maximal length of a transaction was 964 and the average length of the transactions was 24.64. The transactions were generated from the synthetic dataset with the following parameters. The number of trajectories: 50–300, The length of the reference space: 30–70, The average length of the trajectories: 10 – 100, The maximum time span: 30, The number of potential frequent patterns: 1000, The average length of the potential frequent patterns: 25. The minimum support was set as 50%. A comparative study is made after the implementation of the modified Apriori based mining (ABM) with the graph based mining [4] (GBM) across their execution time in milliseconds (ms) over various minimum support threshold is obtained and is rounded up to nearest tenth places and tabulated in table 8. With these results a graph is plotted and shown in Figure 1. The experiment is executed with various trajectories and their corresponding execution time milliseconds (ms) are noted and is rounded up to their nearest tenth places and tabulated in table 9. Based upon these values a graph is plotted and shown in Figure 2. Based on these results, it is clear that the modified Apriori based mining is better than existing graph based mining.

Table 8

Minimum support threshold	Execution time(ms)	
%	ABM	GBM
10	900	1200
20	850	950
30	730	830
40	650	775
50	550	650

Table 9

No of Trajectories	Execution time(ms)	
	GBM	ABM
500	600	450
1000	650	500
1500	700	530
2000	800	600
2500	830	630
3000	860	650

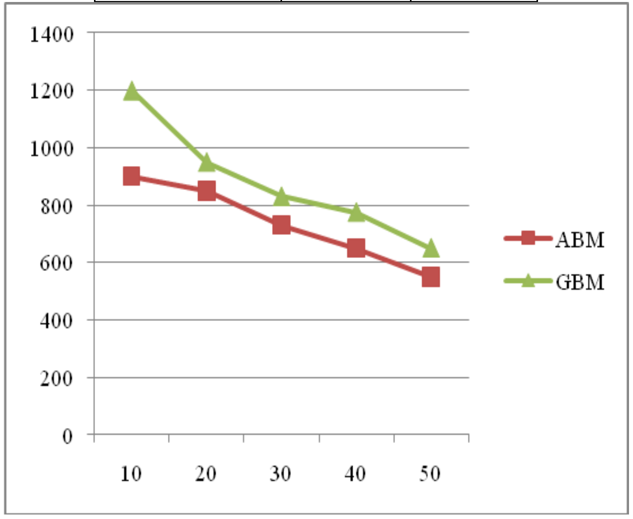

Figure 1 Execution time Vs minimum support threshold

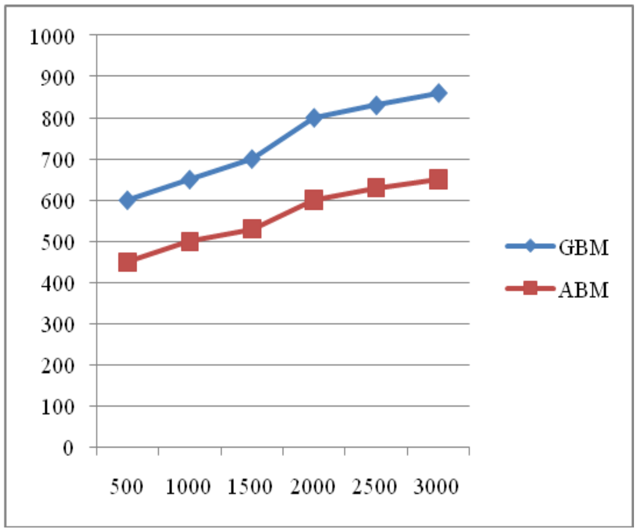

Figure 2 Execution time Vs various number of trajectories

6. CONCLUSION AND FUTURE WORK

In this paper, proposes the Apriori Algorithm based frequent trajectory pattern mining algorithm to efficiently and effectively handle the trajectory database transaction. Prior to that the trajectory dataset is extracted from a text file and is imported to a Oracle database after doing the initial data cleaning process. Initial frequency count is done in Oracle database using its programming feature. Then the data is written in the operating system then further processing is done to find the frequent trajectory pattern. Advantage of this method is later iterations are much faster than the initial iterations of the algorithm. The results obtained by this method are more accurate and reliable. This algorithm uses large coordinate set property. Each iteration in this algorithm can be parallelized so that execution time can be reduced. More over this algorithm is easy to implement. Disadvantage of this method are, it uses a generate, prune and test approach generates candidate coordinate sets (1-coordinate, 2-coordinate, 3-coordinate,…), to check the generated sequence of coordinates are already generated or not, and tests if they are frequent by scanning the database and counting their support each time. Generation of candidate coordinate sets is expensive (in both space and time). Since generation and pruning steps are in memory resident, it needs more RAM. Another disadvantage is it needs n+1 database scans, n is the length of the coordinates in the longest pattern.

The same algorithm can be applied to mine frequent curve patterns, also to mine frequent patterns of molecules which are dealt in biomedical applications. The same algorithm can be modified little to accommodate the frequent trajectory pattern mining in 3D space. Further enhancements of this algorithm can be done to improve performance by pruning method. Also other enhancements like the Partitioning technique, Sampling approach, Dynamic itemset counting and Integrating mining with relational database systems can be done to improve the performance. Based on the observed frequent trajectory pattern and apply the advanced techniques to predict the path of the object. In this way the frequently occurring trajectory patterns will help to analyze and predict the future movement of objects.

7. ACKNOWLEDGMENTS

The main author is grateful to the research supported by the TATA Consultancy Services scholarship.

8. REFERENCES

[1] Arthur.A.Shaw, Mining Frequent Curve Patterns using Apriori Algorithm. In: Proceedings of the International Conference on Innovative Research In Engineering And Technology, ICIRET 2010, Coimbatore, India.

[2] Jiawei Han, Hong Cheng,Dong Xin, Xifeng Yan (2007) Frequent pattern mining: current status and future directions. In the *Journal of Data Min Knowl Disc* (2007) 15:55–86, Springer Science+Business Media, LLC 2007.

[3] Agrawal R, Imielinski T, and Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993ACM-SIGMOD International conference on management of data (SIGMOD'93), Washington, DC, pp 207–216.

[4] Anthony J.T. Lee, Yi-An Chen, Weng-Chong Ip (2009). Mining frequent trajectory patterns in spatial–temporal databases. In the *Journal of Information Sciences* 179 (2009) 2218–2231.

[5] United States Patent Application Publication – Baseball Practice Systems, Pub. No.: US2009/0163301 A1, Inventors: John Flading, Marietta, GA (US) and Larry Duan Cripe, Seattle, WA (US).

[6] United States Patent Application Publication – Trajectory Detection And Feedback System For Tennis, Pub. No.: US2008/0200287 A1, Inventors: Marty, Alan W. (Menlo Park, CA, US) and Edwards, Thomas A. (Menlo Park, CA, US).

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.C. Hsu, Mining frequent patterns without candidate generation, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000, pp. 1–12.

[8] T. Hu, S.Y. Sung, H. Xiong, Q. Fu, Discovery of maximum length frequent itemsets, *Information Sciences* 178 (1) (2008) 69–87.

[9] J.X. Yu, Z. Chong, H. Lu, Z. Zhang, A. Zhou, A false negative approach to mining frequent itemsets from high speed transactional data streams, *Information Sciences* 176 (14) (2006) 1986–2015.

[10] M.J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, *Machine Learning* 11(5)(2001)31–60.

[11] J, Ayres, J.E. Gehrke, T. Yiu, J. Flannick, Sequential pattern mining using a bitmap representation, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 429–435.

[12] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M.C. Hsu. FreeSpan: frequent pattern-projected sequential pattern mining, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 355–359.

[13] J. Pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal and M.C. Hsu. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth, in: Proceedings of the IEEE

International Conference on Data Engineering, 2001, pp. 215–224.

[14] R. Srikant, R. Agrawal, Mining sequential patterns: generalizations and performance improvements, in: Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, 1996, pp. 3–17.

[15] E. Gudes, E. Shimony, N. Vanetik, Discovering frequent graph patterns using disjoint paths, IEEE Transactions on Knowledge and Data Engineering 18(11) (2006) 1441–1456.

[16] J. Huan, W. Wang, J. Prins, Efficient mining of frequent subgraphs in the presence of isomorphism, in: Proceedings of the IEEE International Conference on Data mining, 2003, pp. 549–552.

[17] Y. Huang, H. Xiong, W. Wu, P. Deng, Z. Zhang, Mining maximal hyperclique pattern: a hybrid search strategy, *Information Sciences* 177 (3) (2007) 703–721.

[18] A. Inokuchi, T. Washio, H. Motoda, An Apriori-based algorithm for mining frequent substructures from graph data, in: Proceedings of the European Conference on Principles and Practice of Knowledge in Databases, 2000, pp. 13–23.

[19] R. Jin, C. Wang, D. Polshakov, S. Parthasarathy, G. Agarwal, Discovery frequent topological structures from graph datasets, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005, pp. 606–611.

[20] M. Kuramochi, G. Karypis, Frequent subgraph discovery, in: Proceedings of the IEEE International Conference on Data Mining, 2001, pp. 313–320.

[21] X. Yan, J. Han, gSpan: graph-based substructure pattern mining, in: Proceedings of International Conference on Data Mining, 2002, pp. 721–724.

[22] U. Yun, A new framework for detecting weighted sequential patterns in large sequence databases, Knowledge-Based Systems 21 (2) (2008) 110–122.

[23] M. Garofalakis, R. Rastogi, K. Shim, Mining sequential patterns with regular expression constraints, IEEE Transactions on Knowledge and Data Engineering 14(3) (2002) 530–552.

[24] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, Trajectory Pattern Mining, In: Proceedings of the 13th ACM SIGKDD International Conference on KDD'07, USA, pp. 330–339.

[25] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto and Christelle Vangenot, A conceptual view on trajectories, In: *Elsevier Data & Knowledge Engineering* 65 (2008) 126-146.

9. AUTHORS PROFILE

Arthur.A.Shaw received the MCA degree from Madurai Kamaraj University, through AC College of Engineering and Technology, Karaikudi. Obtained first class with distinction in M. Tech., from MS University, Tirunelveli.

He worked as business application developer in 3GL and 4GLs during 1992 to 1998. Migrated to CRM as Clarify solution architect. Consultant for HP – Cupertino, QualComm – San Diego, F5 Networks – Seattle and Sprint PCS – Kansas City in US between 1998 and 2002 and served as Assistant Professor of Computer Applications in Engineering College affiliated to University of Kerala from 2005 to 2008. Currently pursuing research in Data Mining at National Institute of Technology, Tiruchirappalli 620015 from 2008 onwards. His research interests include database management, Software Engineering, MIS and Computer languages.

N.P. Gopalan received the M.Sc. from Madras University in 1978 and the PhD in applied mathematics from Indian Institute of Science, Bangalore, in 1983. Currently he is a Professor with the Department of Computer Applications in the National Institute of Technology Tiruchirapalli, Tamil Nadu, India. His research interests include algorithms, combinatorics, data mining, and distributed, parallel and grid computing.