An Efficient Algorithm for Mining Frequent Sequences In Spatiotemporal Data

Vu Thi Hong Nhan, Cheong Hee Chi, and Keun Ho Ryu
Database/Bioinformatics Database Laboratory, Chungbuk National University
Email: {nhanvth,jhchi,khryu}@dblab.chungbuk.ac.kr

Abstract

Spatiotemporal data mining represents the confluence of several fields including spatiotemporal databases, machine learning, statistics, geographic visualization, and information theory. Exploration of spatial data mining and temporal data mining has received much attention independently in knowledge discovery in databases and data mining research community. In this paper, we introduce an algorithm Max_MOP for discovering moving sequences in mobile environment. Max_MOP mines only maximal frequent moving patterns. We exploit the characteristic of the problem domain, which is the spatiotemporal proximity between activities, to partition the spatiotemporal space. The task of finding moving sequences is to consider all temporally ordered combination of associations, which requires an intensive computation. However, exploiting the spatiotemporal proximity characteristic makes this task more computationally feasible. Our proposed technique is applicable to location-based services such as traffic service, tourist service, and location-aware advertising service.

1. Introduction

New types of mobile, location-based services (LBS) have been deploying along with the development of hardware technologies including continuous miniaturization of electronics technologies in display devices, and in wireless communication. Specially, global positioning systems (GPS) are becoming increasingly available and accurate. Moving users disclose their positional information to services, which in turn use this information and other information to provide specific functionality.

Each user receives a service customized to the user's specific preference, needs, and current situation. The accumulated data is used for longer-term strategic decision making[16].

To aid this decision making and customization, data mining techniques can be applied to discover interesting knowledge about the behavior of users. Past user movement patterns are possibly used to predict current and future user locations, then users will be provided proper services customizing their references according to this estimated location context. The need to simultaneously investigate both spatial and temporal relations complicates the data mining tasks. A crucial challenge in spatiotemporal data mining is the exploration of efficient methods due to the large amount of spatiotemporal data and the complexity of spatiotemporal data types, data representation, and spatial data structure. Recently, there is an approach DFS_MINE discovering spatiotemporal patterns for weather prediction, but does not show how to apply for mining location patterns of moving objects whose positions continuously change over time [7]. In this paper, we introduce an algorithm Max_MOP for mining frequent moving patterns in mobile environment. Max_MOP find only maximal frequent patterns. We exploit the characteristic of the problem domain, which is the spatiotemporal proximity between activities, to partition the spatiotemporal space. The task of finding moving sequences is to consider all temporally ordered combination of associations, which requires an intensive computation. However, exploiting the spatiotemporal proximity characteristic makes this task more computationally feasible. That is, the antecedent and consequent of a pattern are either likely to be close together in space and not too far in time or have some repeating temporal pattern, e.g., the same weekday. This technique is applicable to information systems(e.g., tourist,

This research was supported by University IT Research Center Project in Korea.

travel, and traffic information systems supporting queries based on physical user location) or server-side selective information dissemination based approach(e.g., target advertisement based on user profiles and location information).

The remainder of this paper is organized as follows: in section 2, we survey related work. The problem statement and denotations are described in section 3. Section 4 introduces the process of discovering moving sequences in detail. Section 5 shows the experimental results proving that our method outperforms previous solutions.

2. Related work

This section reviews some previous studies related to knowledge discovery mining. Sequential patterns are described as the discovery of inter-transaction patterns in large customer transaction databases [1][6][11][12]. In this setting transactions are associated with itemsets.

One can divide approaches for finding frequent itemsets based on two criteria: by their strategy to traverse the search space and by their strategy to determine the support values of itemsets. Based on the first criteria there are two common approaches breadth-first search and depth-first search. A comparison of methods based on these approaches revealed that all of the methods have some types of data for which they perform better than the others [13]. Moving sequence mining deals with series of object location information involving temporal and spatial dimensions. Exploration of spatial data mining [3][10][11][13][14] and temporal data mining [2][15] has received much attention independently in knowledge discovery in databases and data mining research community. While spatial data mining techniques do not consider the evolution of objects over time, temporal data mining techniques alone are inefficient to discover spatial patterns of objects whose locations continuously vary over time. Spatiotemporal data mining methods proposed recently includes [7][8][17][18]. DFS_MINE in [7] discovers spatiotemporal sequential patterns. It is seeking relationships between time-varying attributes for fixed locations, not seek relationships between stable attributes of objects with varying-time locations. Periodic patterns are introduced in [17] for periodic moving objects. Some other techniques mainly focus on the models and structures for indexing moving objects [8].

3. Definitions

A moving database D consists of a set of time series of locations, one for each object. Assume that there are n distinct moving objects. D is defined as the union of time series of locations belonging to all the objects, i.e., $D = \bigcup_{i=1}^{n} D_i$, Each D_i is a time series containing triples (x,y,t).

The locations of moving objects are mapped to a reference map $P \subseteq R^2$. Spatial organization of map is partitioned according to a thematically significant interpretation of space. Here, we choose a thematically "neutral" partition. The means, P is represented as a grid-like partition in equal-sized cells, a finite set of areas $\{a_1,\ldots,a_n\}$ such that $\cup_{i=1}^{n} a_i = P$ and $a_i \cap a_j = \phi$, $i \neq j$.

We assume that GPS device will provide at least one location for each area crossed by an object o. A trajectory pattern is defined as a sequence of area labels. Some consecutive locations may appear several times in the same region. We eliminate such repetition since it provides no useful information. A trajectory $oid.traj = <l_1,l_2,\ldots,l_n>$ with $l_i=(x_i,y_i)$ now can be represented as following: $oid.traj = <$ *eliminate* (*eliminate* (*label* (l_1). *label* (l_2))....*label* (l_n))>.

To satisfy the antecedent and consequent of a sequential pattern being either likely to be close together in space and not too far in time, we impose a timing constraint *max_gap* on them. If the maximum time difference (*max_gap*) is ten hours, then the object must visit one area within two hour of the area of the previous location visited.

Definition 1: Moving pattern is an ordered list of areas $ms=<a_1 \rightarrow a_2 \ldots \rightarrow a_q>$ where a_i is an area indicating where the location of the object is at time t_i, and $t_{i+1}-t_i \leq max\_gap$, $2 \leq i \leq q$. A sequence is composed of k areas is denoted as k-pattern.

For a sequence s_1, if area a_1 occurs before a_2, we denote $a_1 < a_2$, we say s_1 is a subsequence of another sequence s_2 is there exists a one-to-one order preserving function f that maps areas in s_1 to s_2, that is 1) $a_i = f(a_i)$ and 2) if $a_i < a_j$ then $f(a_i) < f(a_j)$ and $t_{f(ai+1)} - t_{f(ai)} \leq max\_gap$. For example, the sequence A→B→C is the subsequence of A→B→D→C→F with assumption that $t_C - t_B \leq max\_gap$.

Let $MS=\{ms_1,\ldots,ms_m\}$ be a set of moving sequences. The support of a sequence *ms* can be defined as a fraction of sequences containing *ms*.

A user-predefined minimum support *min_sup* threshold is the lowest value that each sequence satisfies. If a sequence *ms* has support(*ms*)≥*min_sup*, then it is defined as a frequent sequence. A frequent sequence is maximal if it is not a subsequence of any other sequences.

The problem definition: given a moving object database D, a reference map $P \subseteq R^2$ decomposed according to a resolution *res*, a minimum support *min_sup*, timing constraints *max_gap*. The problem is to discover all maximal frequent sequences in D satisfying *min_sup*.

4. Frequent moving patterns discovery

This section presents the steps to find patterns in detail.

The process of generating moving sequences is described as follows. At first, the database is arranged by object identifier *oid* as the primary key and effective time t as the assistant key. The reference map is decomposed into an array $n_x \times n_y$ of equal-sized cells *res*. The search space has an origin at a point with coordinate (x_0, y_0) and the size $[S_x, S_y]$. Each cell *cid* corresponds to one page with distinct address *pageid*. Now every moving object trajectory is converted into a set of cells *cids*, each page is only allowed to contain one point for each trajectory. Therefore, we have to summarize the trajectories by eliminating the locations falling into the cell in which its previous location is available as mentioned previously.

While a sequence as an object of pattern mining is clearly defined in the transaction database, a sequence as an object of moving pattern mining is not. In order to obtain significant sequences, we impose timing constraints on object locations. Only when the time between two locations stay within *max_gap* can a sequence be produced. Consequently, the database D is transformed into a database in the form of a set of data-sequences, each for a moving object *oid*. This operation is tackled by the function MovingSequenceExtraction().

Max_MOP() described in figure 2 mines only maximal frequent patterns, instead of all frequent patterns, we save a lot of space while maintaining all the necessary information.

The data structure includes a list of minimal infrequent patterns *MinInfreqList* and a list of all maximal frequent patterns *MaxFreqList* in memory and a list *CandList* keeping patterns to generate patterns with higher length in the next step.

```
Function MovingSequenceExtraction()
Input:
  - D: moving object database
  - P: the reference map
  - res: resolution
  - ,max_gap: timing constraints
Output: set of moving sequences (MS)
Method:
  SortedDB← sort database D by (oid,t);
  DIR[1:n_x,1:n_y]←PartitionMap(P,res);
  MS←ϕ;
  For all moving objects oid in SortedDB do
  Traj←ϕ; //set of moving sequences for 1 object
  Seg←DIR[l_1.x;l_1.y];//locations within max_gap
  PrevLoc←1;//previous location of object
  For (i=2; i<=oid.numberOfLocation(); i++)
  If (l_i.t - l_prevLoc.t) > max_gap
        Traj←Traj∪Seg;
        Seg← DIR[l_i.x; l_i.y];
        PreLoc ← i;
  Else If (DIR[l_i.x; l_i.y]≠DIR[l_i-1.x; l_i-1.y] )
        Seg←Seg∪DIR[l_i.x; l_i.y];
        PrevLoc ← i;
  Traj←Traj∪ seg;
  MS ← MS ∪ ms;
Return MS
```

Figure 1: Algorithm for transforming moving database into a set of moving sequences

To find all 1-patterns, a pass is made over the transformed database to count all single cells, listing all frequent cells. From the frequent cells, a set of candidate 2-patterns is created. Another pass is made to gather their supports. The frequent 2-sequences are used to discover patters with longer length, till no frequent sequences are found. There are two main steps as follows:

Candidate generation: given a set of frequent (k-1)-patterns F_{k-1}, the candidates for are created by joining F_{k-1} with itself. A sequence s_1 joins with s_2 if the subsequence obtained by dropping the first element of s_1 is the same as the subsequence obtained by dropping the last element of s_2. The candidate sequence generated by joining s_1 with s_2 is the sequence extended s_1 with the last element in s_2.

Support counting: Count all moving sequences *oid* that contain the candidate sequences. To check if a moving sequence contains a candidate sequence we apply the definition of subsequences mentioned previously with respect to *max_gap*.

We prune *k*-candidates that have an infrequent subset before counting their supports by keeping a list all minimal infrequent patterns in memory *MinInfreqList*. Initially, *MinInfreqList* contains all infrequent 2-patterns. When a new candidate pattern *S* is generated, check if it is a superpattern of any of the patterns in this list. A pattern *S* is inserted in it when all three of the following conditions hold: *S* is not already in *MinInfreqList*, *P* is not a superpattern of some minimal infrequent pattern already in in *MinInfreqList*, and *S* was scanned in the database and was found to be infrequent. After inserting we remove all superpatterns of *S* in *MinInfreqList*. The structure is kept as a list of sequences sorted in increasing order of length

```
Function Max_MOP()
Input:
    - MS: Set of moving sequences
    - max_gap: maximum time constraint
    - min_sup: minimum support
Output: MaxFreqList
Method:
F1←a set of frequent cells;
CandList ← a set of 2-patterns;
MaxFreqList← a set of 2-patterns;
While( CandList ≠φ )
        Temp← generate candidate patterns by
        joining CandList with itself and Prune
        by looking for a subpattern in
        MinInfreqList;
            If no such patter exists, then
            update MinInfreqList;
        For all moving sequences oid in MS do
            Increment counter for all
            c∈Temp contained in oid and
            satisfy min_gap and max_gap
        CandList← a set of c∈ CandList with
        c.counter≥min_sup;
        Update MaxFreqList for all c∈
        CandList with c.counter≥min_sup;
return MaxFreqLis;
```

Fig.2. Algorithm for all maximal frequent moving pattern mining

A candidate pattern is inserted in *MaxFreqList* if all of the three conditions hold: *S* is not already in *MaxFreqList*, *S* is not a subsequence of some maximal frequent sequence already in *MaxFreqList*, and *S* was scanned in the database and was found to be frequent. In this case, we need to insert the new pattern in *MaxFreqList*. After inserting we remove all subpatterns of *S* in this list. *MaxFreqList* is initialized with all frequent 2-patterns generated as in the first algorithm. The result of this algorithm is kept in *MaxFreqList*.

Now we make a comparison between DFS_MINE and Max_MOP() to see their difference and show that our method is superior to the previous one. DFS_MINE follows the concepts of Depth-First-Search, that is, it tries to discover frequent sequences of length without exhausting. But it does not aim at minimizing the database scans. It generates k-sequences by using a (k-1)-pattern and intersecting with all frequent 1-sequence and not in the useless set of (k-1)-sequence. The intersection of an item with a (k-1)-sequence involves inserting the item in all possible position in the (k-1)-sequence. Consequently, comparing with DFS_MINE generating candidate patterns in Max_MOP() is simpler and also limits the number of candidates at each step and thus save time.

5. Experimental results

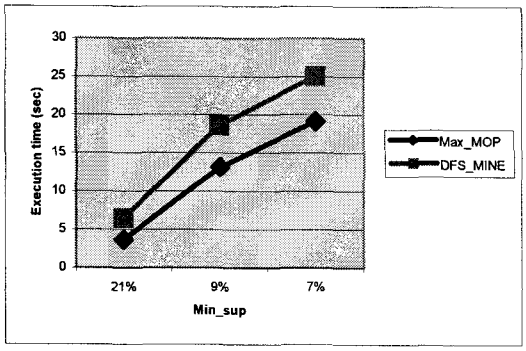
To carry out performance studies we implemented Max_MOP and DFS_MINE [7] in C++. The experiments were performed on a PC Pentium IV, 2.00GHz processor with 256MB RAM.

The experiments were performed on synthetic datasets generated by a generator that simulates the moving objects. The parameters used for synthetic database generation are shown in the table below.

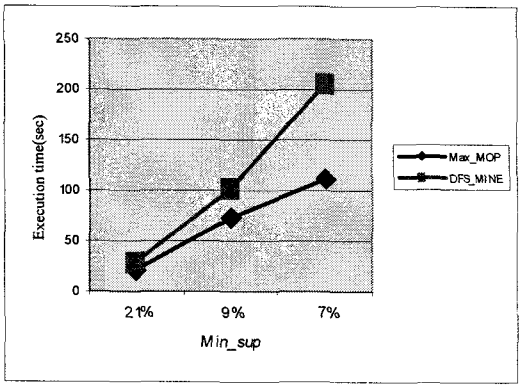
D	#of distinct objects
T	#of time points of a object trajectory
max_gap	Timing constraint
res	Space constraint
L	The average size of the potentially maximal patterns

Datasets are generated by the following principle: on average 70% object trajectories have the maximal length *L* in which average 30% trajectories have the same trip; the rest of object trajectories with the length less than *L* is also generated using that rule. Here we fix the *max_gap* with 13 hours and the time points of each trajectory are changed from *L*-1 to *L*+10.

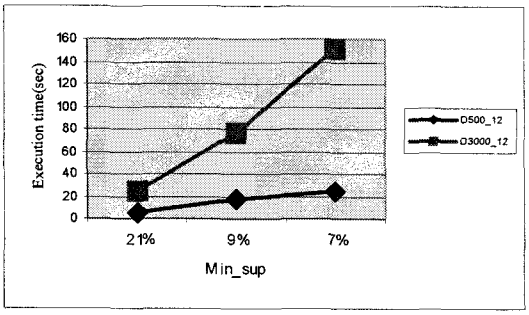
Dataset D500_L10 means 500 moving object trajectories generated and an average size of their maximal length of 10. We generate four datasets with different maximal length and number of objects including D500_10, D3000_10, D500_12, and D3000_12.



(a) D500_L10



(b) D3000_L10



(c)

Fig.3 Comparison between Max_MOP and DFS_MINE

Figure 3 shows the runtime behavior for different datasets by fixing the number of objects and changing the maximal lengths or vice verse according to different *min_sup* values. As the graphs in figure 3a and 3b shown our algorithm is less time-consuming than DFS_MINE. The reason is explained in the previous sections that the number of generated candidates using our methods is less than that of DFS_MINE. MaxMOP also saves storage space.

Moreover, Max_MOP is also scalable as indicated in figure 3c. When the number of object increases from 500 to 3000 (a factor of 6) with a fixed maximal length, the required time increased by a factor less than 6. Our other tests also return the same result.

6. Conclusions

In this paper, we offered the algorithms MaxMOS for mining all maximal frequent moving pattern in mobile environment. We presented a way to define patterns as objects of moving pattern mining. Patterns are derived based on spatio-temporal distances of moving objects, space constraint and timing constraint. At first, the object space is decomposed into areas. The moving object database is preprocessed using a clustering method, that is, time attributes are discretized. Finally, our proposed algorithm enumerates all maximal frequent moving sequences. We conducted an experiment to study the performance of them and also compared with the typical technique proposed previously DFS_MINE. From the experimental results, we prove that our proposals are quite efficient to apply to LBS applications such as tourist services, safety-related services, location-aware advertising services and so on.

We are implementing a moving patterns discovery prototype with visualized interface, generating rules applied in queries. Besides that, we will also apply an approximate counting method to discover pattern with less time-consuming requirements and find patterns regarding to multi-scales of the data model.

References

1. R.Agrawal & R.Srikant. Mining Sequential Patterns. Research report, 1995.
2. S.Chakrabarti, S.Sarawagi, & B.Dom. Mining Surprising Patterns using Temporal Description Length. In Proc. of 24th VLDB, 1998.
3. K.Koperski, J.Han. Discovery of Spatial Association Rules in Geographic Information Databases. In Proc. of 4th Int. Symp. on Advances in Spatial Databases, 1995.
4. J.F.Roddick & B.G.Lees. Paradigms for Spatial and Spatio-Temporal Data Mining. Geographic Data mining and Knowledge Discovery, 2001.
5. J.F.Roddick & M.Spiliopoulou. A Survey of Temporal Knowledge Discovery Paradigms and Methods. IEEE Trans. on Knowledge and Data Engineering, 2002.
6. R.Srikant, & R. Agrawal. Mining Sequential Patterns: Generalizations and Performance

Improvements. In Proc. Int'l Conf. Extending Database Technology, p.3-17, 1996
7. I.Tsoukatos & D.Gunopulos. Efficient Mining of Spatiotemporal Patterns. In Poc. On SSTD, LNCS, p.425-442, 2001.
8. Y.Wang, E.P.Lim, S.Y.Hwang. On Mining Group Patterns of Mobile Users. LNCS, DEXA 2003.
9. W.Wang, J.Yang, R.Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. 23rd VLDB conference, 1987.
10. W.Wang, J.Yang. R.Muntz. STING+: An Approach to Active Spatial Data Mining, Int.l Conf on Data Engineering, ICDE, 1999.
11. M.Zaki. SPADE: An efficient Algorithm for Mining Frequent Sequences. Machine learning Journal, 2001.
12. J.Hipp, U.Guntzer, G.Nakhaeizadeh. Algorithms for association rule mining- A general survey and comparison. ACM SIGMOD 2000.
13. K. Koperski, J.Han, & J.Adhikary. Mining Knowledge in Geographical Data. Communications of ACM , 1998
14. H.J.Miller & J.Han. Geographic data mining and knowledge discovery: an overview. In Miller, H.J. and Han, J. (eds) Geographic data mining and knowledgediscovery, 2001.
15. J.F.Roddick & B.G.Lees. Paradigms for spatial and spatio-temporal data mining. In H.G. Miller and J. Han (eds), Geographic Data Mining and Knowledge Discovery. 2001.
16. C. S. Jensen, A. Kligys, T. B. Pedersen, & I. imko. Multidimensional Data Modeling for Location- Based Services. In Proc. of GIS'02, pp. 8—9, 2002.
17. N.Mamoulis, H.Cao, & G.Kollios. Mining, Indexing, and Querying Historical Spatiotemporal Data. KDD, 2004.
18. S.Brakatsoulas, D.Profer, & N.Tryfona. Modeling, Storing, and Mining Moving Object Databases.