SMTP: An Optimized Storage Method for Vehicle Trajectory Data Exploiting Trajectory Patterns

Hailun Wang*, Mingming Zhang*, Renyu Yang*, Xuelian Lin*, Tianyu Wo*, Rajiv Ranjan‡, Jie Xu§*
*State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
‡School of Computing, University of Newcastle upon Tyne, Newcastle, UK
§School of Computing, University of Leeds, Leeds, UK
Email: {wanghl, zhangmm, yangry, linxl, woty}@act.buaa.edu.cn; raj.ranjan@ncl.ac.uk; j.xu@leeds.ac.uk

Abstract—**Recent advances in location-acquisition and mobile sensing technologies have enabled tracking of vehicle movements (i.e., trajectory data). Massive trajectory datasets are processed routinely (often in real-time) to provide support for many new types of IoV (Internet of Vehicles) applications (e.g., traffic congestion management, and load-coordination across electric vehicle charging stations). High-volume, high-velocity data emitted by IoV applications introduces issues with efficient spatial and temporal queries over massively redundant datasets, typically represented as a collection of longitude-latitude tuples. In this paper we present SMTP, a new storage method based on the recognition of trajectory patterns to reduce the storage space for the trajectory data. An adaptive algorithm for mining trajectory patterns from the data is developed, and it recognizes frequent trajectories as patterns according to the geo-space relationships between trajectories. A combinatorial optimization algorithm is then introduced to decide which trajectory patterns should be used for trajectory storage, thereby removing redundant data and saving space. The recognized and saved patterns also help to accelerate queries to the trajectory data. Several large IoV datasets from the real world are used to validate the effectiveness of the proposed method. Experimental results show that storage space for trajectory data can be reduced by 38% while a typical query to the data can be accelerated by approximately 40%.**

Index Terms—**IoV; Trajectory Patterns; Trajectory Pattern Mining; Vehicle Trajectories**

I. Introduction

The ubiquitous exploitation of trajectory data has been driven by recent advances in location-acquisition, mobile sensing, and IoV technologies [5]. The growing use of data such as position, sensing data etc. together with rapidly urban development leads to the large volume and highly redundant trajectory data. In reality, the real-time location information of vehicles is leveraged by many taxi corporations to realize a timely scheduling of vehicles [10] [19]. To detect the number of event and speed of data, it is mandatory to investigate efficient trajectory data management algorithms and approaches. The service (e.g., Uber [4]) and infrastructure providers (e.g., Amazon Web Services [1]) can utilize our trajectory data management method to provision effective and reliable IoV application that not only reduces operational cost (especially storage overheads) but also improve query processing efficiency [20] [21].

IoV applications are typically spatio-temporal data management and query processing applications that require varying level of accuracy driven by decision making scenarios (e.g. traffic flow management, scheduling of taxi service etc). Traditionally, each GPS point within a trajectory is represented in longitude-latitude tuple *PT* denoted by <*longitude,latitude,timestamp*>. This form is intuitive but primarily suffers from two fundamental problems: imprecision stemming from record errors and space complexity due to high volume and velocity of trajectory data samples [6]. In practice, in a real-time location query, merely the road and distances are required, rather than a complete series of GPS position information. Inevitably, utilizing original GPS data would increase the query time complexity and the storage overhead.

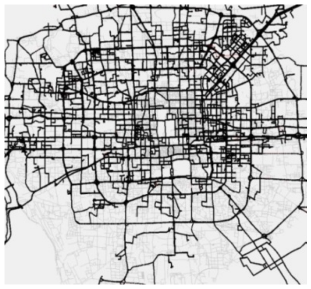


Fig. 1: Trajectories in road network of Beijing

With the increasing scale of vehicles and road networks, the probability of repetitive trajectories increases. In Figure 1, the thickness of a line represents the trajectory number in Beijing road network. Apparently, numerous trajectories manifest in the ring and backbone roads in Beijing. As vehicle trajectories are directly related to road network layout, spatio-temporal search queries in IoV application are represented as <*rid,dis,t*> (i.e., road and distance). In addition, compression approaches based on road network are further proposed in [11] [17]. However, these methods have not fully taken advantage of road networks and the lossy compression cannot satisfy different accuracy requirements in the IoV applications, resulting in ineffective query processing. In fact, *frequent trajectory* has been applied in the frequent trajectory mining and path planning [9] [14]. The most frequent subsegments that constantly appear in many trajectories can be recognized as a *trajectory pattern* [8]. Undoubtedly, defining, detecting and leveraging frequent trajectory patterns to represent the positions can mitigate the storage redundancy whilst accelerating the trajectory queries.

Frequent trajectory mining methods are proposed in existing literature [8] [13] [15] [17]. However, those approaches generate a large number of subsets of frequent trajectories, making it infeasible to apply into current large-scale IoV applications that typically contain high volume trajectory data. Therefore this raise two research challenges: *1) how to efficiently recognize frequent trajectory patterns based on large-scale tracking data; 2) how to effectively overlay the real-time vehicle trajectories over geo-space patterns for removing redundant data.* Due to the sheer volume and velocity of trajectory data, these problems continue to be intractable. In this paper we present *SMTP* for reducing the storage space of the trajectory data, based on the recognition of trajectory patterns. An adaptive algorithm to mine trajectory patterns from the data is then developed, and it recognizes frequent trajectories as patterns according to the geo-space relationships between trajectories. Afterward, we formulate and apply a combinatorial optimization to decide which trajectory patterns should be utilized to store original trajectories, thereby removing redundant data and saving space. Large-scale IoV datasets from the real world are used to validate and experimental results demonstrate that the storage space can be reduced by 38% and the query can be accelerated by approximately 40%. In particular, the contributions of this paper are as follows:

- An adaptive trajectory pattern mining algorithm based on geo-space relationships between trajectories, fully covering the road network and avoiding the generation of a large number of frequent subsets;
- An efficient combinational optimization algorithm that greedily selects trajectory patterns for roads in each trajectory;
- An implementation of a holistic data storage method that mitigates the storage redundancies and accelerates queries based on trajectory patterns.

The rest of the paper is organized as follows: Section II presents the formal definitions of concepts and problems. Section III describes SMTP in detail. The experiments are shown in Section IV. Section V reviews related work and we conclude our work in Section VI.

II. Problems Definition

In this section, we will define a set of key concepts and problems. Specifically, original trajectory T can be regarded as a sequence composed of a number of GPS points (PTs). Many road segments (RSs) constitute a road R. Road network (RN) is a directed graph that includes a set of road intersections and pertaining roads. After a map matching (MM) procedure, a GPS point PT will be transformed into map-matched point (MMP), and the trajectory T is converted to the map-matched trajectory (MMT). A trajectory pattern tuple (TPT) contains the trajectory pattern TP and distance-timestamp tuples. Based on the trajectory pattern TP, trajectory NT is represented by a TPT sequence. More details can be found in Table I. Based on these concepts, we describe the fundamental problems as follows:

Trajectory Pattern. TP=<pid,(RS_1,$RS_2 \cdots RS_p$),$attributes$>, where pid is the identifier and (RS_1,$RS_2 \cdots RS_p$) is a valid

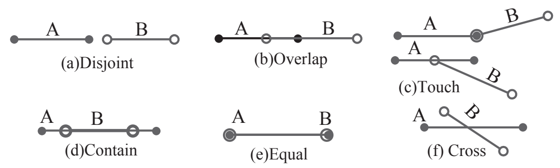


Fig. 2: The line-line relationships in geo-space

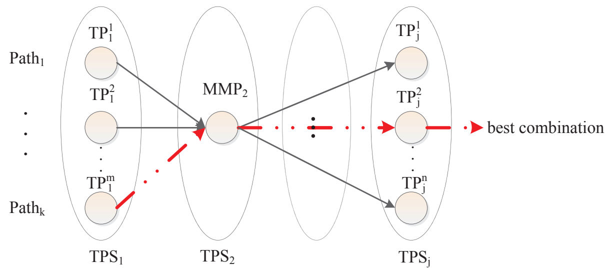


Fig. 3: Example of trajectory pattern optimization

path in RN while *attributes* include distance and direction etc. p is defined as the RS number and the distance is the accumulated road distance of the specific path. We define **min_sup** as the minimized occurrence number of the TP within the connecting $MMTs$, and **min_len** as the minimized RS number that the RS set should contains. These two metrics are significantly important and will be emphatically discussed in our paper.

Trajectory Pattern Mining. Given a road network RN, the historical trajectory set $MMTS$ will be obtained after map-matching. We can discover the trajectory patterns TPS from $MMTS$. Figure 2 demonstrates the line-line relationships in geo-space, mainly including *Disjoint, Overlap, Touch, Contain, Equal and Cross* etc. According to the definition of trajectory pattern, TP consists of road segments. The trajectory patterns will thus have those relationships correspondingly. In the frequent pattern mining, all non-empty subsets of a frequent itemset must be frequent itemsets [18]. Therefore we can conclude that each sub-trajectory of a trajectory pattern must be a trajectory pattern. This indicates that we should avoid the occurrence of substantial generations of trajectory pattern subsets during the pattern recognition from numerous historical trajectories. Moreover, different roads have distinctive traffic flows, resulting in the asymmetrical distribution of trajectories. It is obviously observable from Figure 1 that the ring roads and backbone roads in Beijing have more trajectories. Therefore uniform *min_sup* and *min_len* are extremely difficult to define. It is very likely to omit many important roads by assigning large values, or reserve infrequently-occurred roads as trajectory patterns by small values. It is significantly vital to excavate trajectory patterns by fully leveraging road network information and avoiding excess sub-trajectory patterns.

Trajectory Pattern Combinational Optimization. Given a trajectory pattern set TPS, a map-matched trajectory MMT, the problem we have to deal with is to select suitable trajectory

TABLE I: The definitions of key concepts

Abbreviation	Name	Definition
PT	GPS Point Tuple	PT=<*longitude,latitude,t*>. PT represents a GPS position at time t. PT is a longitude-latitude tuple.
T	Trajectory	T=<PT_1,$PT_2\cdots PT_n$>, n is the number of PT. A trajectory T is a sequence of GPS points ordered by time.
RS	Road Segment	RS=<*rid*,$Node_{start}$,$Node_{end}$,*attributes*>, *rid* is the identifier, and $Node_{start}$ is the start point represented in <*longitude,latitude*> tuple. $Node_{end}$ is the end point, and *attributes* include distance and direction etc.
R	Road	R=<*Rid*,(RS_1,$RS_2\cdots RS_m$),*attributes*>, *Rid* is road identifier, and RS_i is the ith road segment in road R. m is the number of road segments, and attributes include the road name, distance and direction etc. A road is defined as a sequence of RSs.
RN	Road Network	RN=G(V,E). V is the set of road intersections and Nodes represented in <*longitude,latitude*> tuple. E is the set of roads. A road network RN is a directed graph.
MM	Map Matching	A procedure that maps or overlays original trajectory T to the existing road network RN.
MMP	Map-Matched Point	MMP=<*rid,dis,t*>. *MMP* is the reuslt of PT matched to RN, *rid* is the identifier of matched RS, and *dis* is the distance from the matched point to the $Node_{start}$ of RS.
MMT	Map-Matched Trajectory	MMT=<MMP_1,$MMP_2\cdots MMP_l$>, *l* is the number of MMP. A MMT is a sequence of MMPs after T is map-matched to RN.
MMTS	Map-Matched Trajectory Set	MMTS=<MMT_1,$MMT_2\cdots MMT_s$>, MMTS is the set of MMT.
TP	Trajectory Pattern	TP=<*pid*,(RS_1,$RS_2\cdots RS_p$),*attributes*>. *pid* is the identifier, (RS_1,$RS_2\cdots RS_p$) is a valid path in RN, and *attributes* include size p, distance and direction etc.
TPS	Trajectory Pattern Set	TPS=<TP_1,$TP_2\cdots TP_t$>, TPS is the set of TP.
TPT	Trajectory Pattern Tuple	TPT=<*pid*,(<dis_1,t_1>,<dis_2,t_2>$\cdots$<dis_q,t_q>)>. dis_i is the distance from ith point to the $Node_{start}$ of trajectory pattern, and t_i is the timestamp of ith point. Including the pid of TP, TPT is a sequence of distance-timestamp tuple, namely <*distance,timestamp*>.
NT	Trajectory Based On Trajectory Pattern	NT=<TPT_1,TPT_2,$MMP_i\cdots TPT_r$>. r is the number of TPTs and MMPs. NT is a trajectory based on trajectory pattern.
CR	Redundancy Removal Ratio	CR=1-$\frac{ST'}{ST}$, ST is the storage space of trajectory T, and ST' is the storage after removing redundancy.

patterns to generate a new trajectory *NT*, substituting the original one. For each RS_i, it has a trajectory pattern set TPS_i where TPS_i contains the RS_i. The problem can be formalized as follows:

$$\textbf{Minimize: } \bigcup_{i=1}^{n} TP_i^{'}$$
$$\textbf{Subject to:} \tag{1}$$
$$TP_i^{'} = \begin{cases} TP_i^j, & if\ TP_i^j \in TPS_i = \langle TP_i^1, \cdots TP_i^m \rangle \\ & \quad and\ 1 \le j \le m \\ \varnothing, & if\ TPS_i = \varnothing \end{cases}$$

As one trajectory pattern in TPS_i, TP_i^j is used to create *NT* instead of MMP_i.

More specifically, in road network *RN*, the road segment *RS* might belong to many trajectory patterns. Consequently a *MMP* in the *MMT* can be represented by different trajectory patterns. TPS_1 shown in Figure 3 can be illustrated as an example. Some road segments may not belong to any trajectory pattern because trajectory patterns cannot cover all roads. The objective is to achieve the optimal redundancy removal effects, thus the total number of trajectory patterns should be minimized as possible (see Equation 1). Instead of *MMP*, we want to use trajectory patterns to represent the trajectory as more as possible. For a *MMT*=<MMP_1,$MMP_2\cdots MMP_j$>, we assume that the road segment in MMP_j belongs to trajectory pattern set TPS_j. In order to get the optimal combination, the algorithm needs to traverse all the paths in the candidate graphs which contain trajectory pattern sets from TPS_1 to TPS_j. The original complexity of this problem is O(m^j), where *m* is the number of trajectory patterns in candidate trajectory pattern set, and *j* is the number of *MMP* in *MMT*. The data is collected frequently in the IoV scenario, indicating a large number of GPS points within a trajectory. Consequently, the algorithm needs to figure out the targeted trajectory patterns timely in order to meet requirements of real-time traffic processing and analysis.

III. Solutions

In this section we will present our solutions ***SMTP*** in detail. We will introduce the overview followed by description of each key component.

A. Overview

As shown in Figure 4, ***SMTP*** is composed of four components: **Map Matcher**, **Trajectory Pattern Miner**, **Trajectory Optimizer** and **Querier**.

Map Matcher. The original trajectory *T* will be pre-processed and matched to the road network *RN*.

Trajectory Pattern Miner. This module excavates trajectory patterns from a large set of matched trajectories *MMTS*.

Trajectory Optimizer. Trajectory Optimizer selects the optimal trajectory patterns based on the aforementioned optimization problem to create *NT*.

Querier. Querier will perform the storage operations and execute queries based on trajectory patterns.

In particular, Map Matcher divides each trajectory into sub-trajectories according to distance and time of adjacent GPS points. During the matching procedure, Map Matcher also obtains candidate road segments *RSs* by the similarity of distance, direction and the driving rules of roads. After map-matching, the original trajectory *T* is converted to *MMT* which is represented in road network. In order to reduce the subsets of trajectory patterns and mine trajectory patterns fully, Trajectory Pattern Miner adopts an adaptive algorithm based on geo-space relationship of trajectories. We implement a

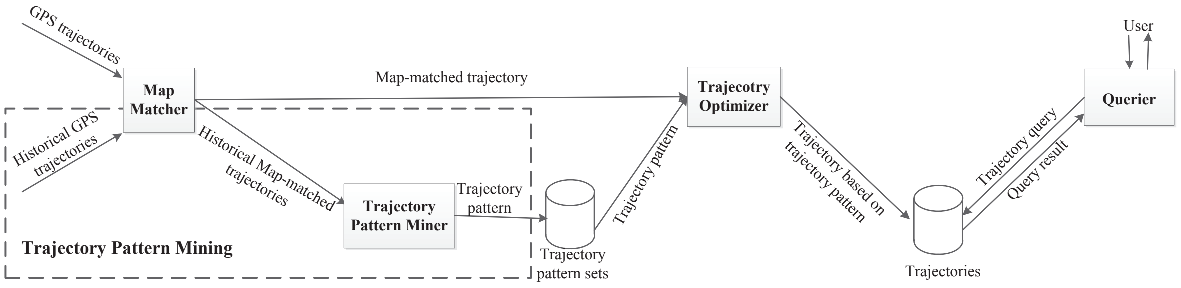


Fig. 4: System overview: components and workflow

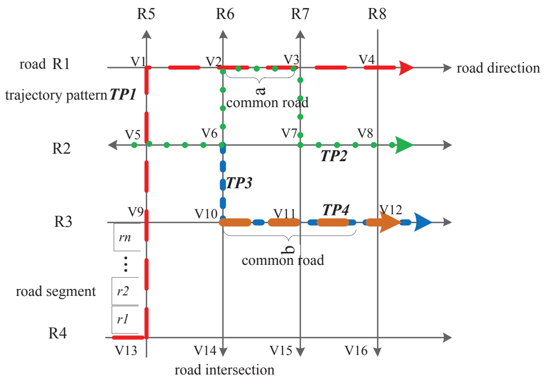


Fig. 5: Example geo-space relationships between trajectories

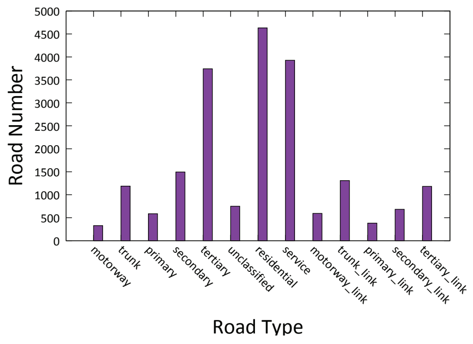


Fig. 6: Distribution of different roads in Beijing

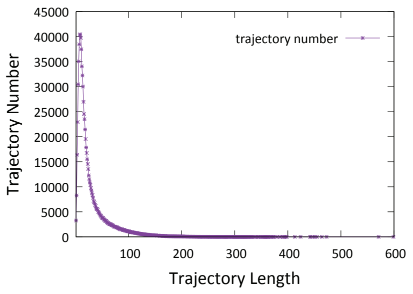


Fig. 7: Trajectory number of different length

greedy algorithm in the Trajectory Optimizer to timely choose trajectory patterns. At last, user can query the trajectories from the Querier based on trajectory patterns.

B. Trajectory Pattern Miner

According to the geo-space relationships among trajectories discussed in Section II and trajectories in Figure 1, the geo-space relationships of Disjoint, Overlap, Contain and Touch are ubiquitous. The geo-space relationships among trajectories are valuable but ignored by many trajectory pattern mining algorithms. In Figure 5, we can find several observations: the trajectory pattern TP_1 is disjoint with TP_3 and TP_4. TP_1 and TP_2 overlap in road a. TP_2 and TP_3 touch at road intersection V_6 and TP_3 contains TP_4(they both have the same road b). Typically, a vehicle often changes directions at the road intersection, resulting in the varying trajectories related to road intersection. In road network *RN*, the road intersections are key variations and connections for trajectories, significantly affecting the mining of trajectory pattern. For example, the road $V_{13}V_9$ includes road segments r_1, $r_2 \cdots r_n$, making the road $V_{13}V_9$ be a trajectory pattern and min_len=1. According to our rules, r_1, $r_2, \cdots r_n$, r_1r_2, r_2r_3, $\cdots$,$r_{n-1}r_n, \cdots$, $r_1r_2 \cdots r_n$ are trajectory patterns. Namely, $r_i \cdots r_j$ $(1 \leq i \leq j \leq n)$ is a trajectory pattern and the number of trajectory pattern generated by road $V_{13}V_9$ is $\frac{n^2+n}{2}$, namely C^2_{n+1}. Assuming min_len is m, the number is C^2_{n-m+2}.

In the Contain relationship, a trajectory pattern of size n can generate n^2 sub-trajectory patterns. Meanwhile, The connection of adjacent trajectory pattern such as Overlap and Touch can create new trajectory pattern, which facilitate the number reduction of trajectory pattern. Due to the enormous differences of traffic flows among different roads, the *min_sup* and *min_len* will be significantly impacted. According to OpenStreetMap [2], the roads of Beijing can be categorized into different highways. Through the mining from road intersections, we can reduce the generation of subsets and mine the trajectory patterns of Touch. To deal with different traffic flows, we adopt an adaptive approach to dynamically determine the value of *min_sup* according to road types. We further propose an efficient algorithm that can automatically excavate trajectory patterns by using geo-space relationships and road types.

The adaptive trajectory pattern mining algorithm starts to mine from road intersections and uses different *min_sup* according to road types. To accurately determine *min_len*, we analyze the distribution of trajectory length and select the largest concentration of length range. We count the number of trajectories in different road types and the number of different

Algorithm 1 Trajectory Pattern Mining Algorithm

```
Input: (1) MMTS;(2) RN.
Output: (1) TPS.
1: TPS ← Ø
2: //construct the whole Trie Root from MMTS
3: for each MMT in MMTS do
4:     append MMT to Trie Root
5: end for
6: //mining TP of each Inter from Root
7: for each Intersection Inter in RN do
8:     get the minimum min_sup of Inter
9:     nodeTrie←ConstructNodeTrie(Inter,min_sup, min_len)
10:    TP ← GetPath(nodeTrie, min_sup,min_len)
11:    TPS ← TP
12: end for
13: return TPS
```

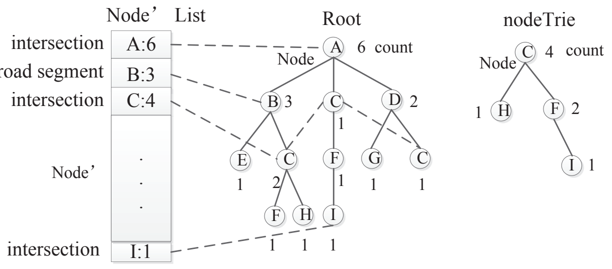


Fig. 8: An example of trajectory pattern mining

trajectory length. The road distribution is shown in Figure 6 and we calculate the number of trajectories passing different roads from the training set. The mining algorithm adjusts the *min_sup* adaptively according to the statistics. As shown in Figure 7, the length are tail-distributed and the length of most trajectories is less than 100 after the map-matching. Thus we set several parameters of *min_len* below 100.

Algorithm 1 depicts the holistic procedure. It takes the set of map-matched trajectories *MMTS* and road network *RN* as input, and set of trajectory patterns *TPS* as output.

At first, we assign to the set of trajectory pattern *TPS* an empty set (line 1), and then construct a whole Trie tree [12] Root and Node' list List from all map-matched historical trajectories (lines 3-5). Each node in Root represents a road segment and the number of trajectories passing the road segment. Each node in the list includes all the occurring positions of the road segment in Root. Secondly, we make sub-Trie nodeTrie start in each intersection Inter according to the Root and List (line 7). Afterwards the algorithm determines *min_sup* adaptively according to the sub-Trie information, such as the road type and trajectory number(line 8). Finally, the algorithm adds the trajectory which appears to be more than *min_sup* and the size is larger than *min_len* as a trajectory pattern to *TPS*(lines 10-11). At the same time, we can also obtain the trajectory pattern's road segment list, distance, size and direction etc. The most advantage of the mining algorithm is the complete utilization of intersection and road type information.

In Figure 8, the *Root* consists of road segments or intersections A, B $\cdots$ I, the Node' *List* comprises *Node* items.

Algorithm 2 Trajectory Pattern Optimizing Algorithm

```
Input: (1) MMT.
Output: (2) NT.
1: lastCandidate ← Ø, currentCandidate ← Ø, NT ← Ø
2: for each MMP in MMT do
3:     currentCandidate ← GetCandidateTP(MMP)
4:     if lastCandidate is Null then
5:         lastCandidate ← currentCandidate
6:     end if
7:     tempCandidate ← lastCandidate ∩ currentCandidate
8:     if tempCandidate is Null then
9:         NT ← one trajectory pattern in lastCandidate
10:        lastCandidate ← currentCandidate
11:    else
12:        lastCandidate ← tempCandidate
13:    end if
14: end for
15: NT ← one trajectory pattern in lastCandidate
16: return NT
```

For example, the intersection C occurs 3 times in *Root*. After creating *Root* and Node' list, the algorithm can easily construct NodeTrie of each road intersection. The road intersection C occurs 4 times in nodeTrie. Let min_sup be 2 and min_len be 2, and consequently the path CF is a trajectory pattern.

C. Trajectory Optimizer

As discussed in Section II, we can get new trajectory *NT* by combining trajectory patterns. In order to find the optimal solution, we have to traverse all the paths of the graph consisting of multiple trajectory pattern sets. To utilize more trajectory patterns in *NT* and reduce time complexity, we greedily select the candidate trajectory patterns of adjacent *MMP*. To this end, we propose an approximate algorithm based on trajectory pattern matching.

Algorithm 2 shows the details. At first, the algorithm takes map-matched trajectory *MMT* as input and trajectory based on trajectory pattern *NT* as output. The algorithm searches and finds each trajectory pattern set *currentCandidate* of *MMP* in *MMT* (line 3). It then gets the intersection with *lastCandidate* of last *MMP* in sequence (line 7). This process repeats until the intersection set tempCandidate is empty. Finally, our approach will find the longest matching trajectory pattern which contains all the prior *MMPs* (lines 8-13), and get the *NT* (line 16). The core philosophy of the greedy algorithm is that it refers to the trajectory pattern which contains more *MMPs*.

Analysis: As for the time complexity, the algorithm examines each intersection of adjacent trajectory pattern sets and the time complexity is $O(m^2)$ where m is the trajectory pattern set's number of a specific *MMP*. As a result, the overall time complexity is $O(m^2n)$, where n is the number of *MMPs* in *MMT*. Apparently, compared to the time complexity $O(m^n)$ of original algorithm, the proposed algorithm is more efficient and feasible to current IoV scenarios.

D. Querier

Based on the above design and implementations, we further extend the trajectory pattern based representation mechanism into the underlying storage and data query to realize the

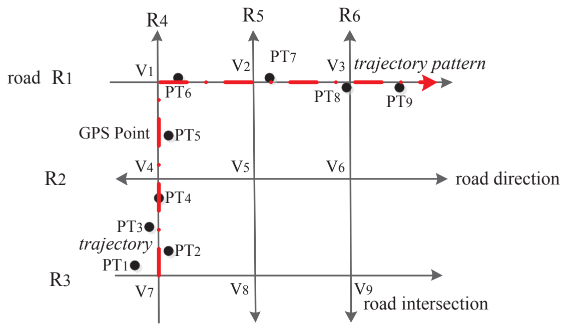


Fig. 9: An example of application

mitigation of data redundancies and the improvement of query performance. In reality, trajectory *NT* by using the trajectory patterns is able to reduce the number of data items, resulting in a sound redundancy removal effectiveness. When querying, the model can calculate the distance according to the trajectory pattern directly.

Consider a case study of the trajectory *T* containing 9 GPS points in Figure 9 to illustrate the storage space reduction. The number of data items of *T* is 27. Assume that the data type of latitude and longitude is double, while timestamp is int. Double variable and int variable need 8 bytes and 4 bytes respectively. Therefore, the total storage of trajectory *T* should be 180 bytes. If the trajectory *T* is represented in *NT*=<*pid*,(<dis_1,t_1>,<dis_2,t_2>$\cdots$<dis_9,t_9>)>, the number of data items of *NT* is 19 and the format of attributes are as below: pid and timestamp are int while distance is double, resulting in a holistic 112 bytes storage.

With regards to the query aspect of the trajectory *T path*, due to the road segments V_7V_4, V_4V_1, V_1V_2, V_2V_3 involved in the trajectory pattern, we can easily get the path between dis_1 and dis_9. The model can get the accurate position directly when querying the real-time location. In real IoV systems and applications, queries are primarily based on the road network. The road list and distance of trajectory pattern will be extremely beneficial and generally applicable to those trajectory data storage models.

IV. Experiments and Evaluation

In this section, we evaluate the effectiveness of the proposed trajectory pattern mining algorithm on its ability of reducing redundant data and improving query processing.

A. Experimental Setup

Road Network. We use the map within the 5th Ring roads of Beijing from OpenStreetMap. The longitude ranges from 116.199814 to 116.554159, and the latitude ranges from 39.750665 to 40.027734. The map of this area contains 106,792 road nodes, 28,972 roads, 151,237 road segments and 37,034 road intersections.

Datasets. Different value of *min_sup* and *min_len* will impact the effectiveness of the proposed trajectory pattern mining algorithm. Therefore, we vary *min_sup* according to the statistics of training set. With regards to mining trajectory patterns, we evaluate the effects by several *min_len*s. In this experiment, we considered 1,000,000 trajectories generated by different vehicles.

Algorithms and Implementation. All algorithms of *SMTP* are implemented in Java and run on a cluster of servers. Each server is equipped with Intel(R) Xeon(R) E5-2650 CPU(2.00GHz) and 256 GB memory. The following metrics are quantified:

1) *Road Coverage Ratio*: $RC = \frac{NRS'}{NRS}$, where *NRS* is the number of distinct *RSs* in *RN*, and *NRS'* is the number of different *RSs* in *TPS*;

2) *Redundancy Removal Ratio:* this ratio *CR* indicates the reduction degree;

3) *Query Time Ratio*: $QTR = \frac{QTP}{QLL}$, where *QTP* is the time of querying trajectory based on trajectory pattern, and *QLL* is the trajectory query time based on longitude-latitude tuple.

We discuss the impact of parameter values in different trajectory pattern mining algorithms in Section IV-B and then evaluate the optimization algorithm of redundancy mitigation in Section IV-C. We also compare the query processing efficiency of trajectory based on trajectory pattern with longitude-latitude tuple method in Section IV-D.

B. Trajectory Pattern Mining

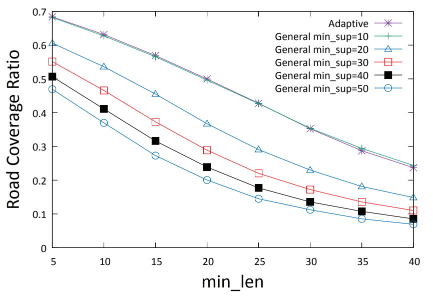
We discuss the impact of different parameter values on adaptive and general trajectory pattern mining algorithms. Figure 10(a) depicts that the road coverage ratio varies with *min_sup* and *min_len* in both adaptive and general algorithm. The road coverage ratio decreases as *min_len* increases in both algorithms. The reason is because the trajectory patterns decrease while *min_len* increases. Due to all intersections in the road network and dynamic *min_sup*, the road coverage ratio *RC* of adaptive algorithm is higher than that of general algorithm. We can conclude that the adaptive algorithm has a larger road coverage ratio than the general algorithm.

Figure 10(b) shows that the Touch relationship ratio varies with *min_sup* and *min_len* in adaptive and general algorithm. The Touch relationship ratio means that the trajectory patterns of Touch relationship account for all the trajectory patterns. Figure 10(c) is the Contain relationship ratio variation. It is observable that the Touch relationship ratio of adaptive algorithm surpasses the general algorithm, but the Contain relationship ratio of adaptive algorithm is lowest. More trajectory patterns of Touch relationship indicates that we are able to combine them to create more trajectory patterns easily, and less trajectory patterns of Contain relationship means that the mining algorithm generates less subsets. This is due to the fact that the adaptive algorithm makes use of the geo-space relationship and reduces the generation of subsets.

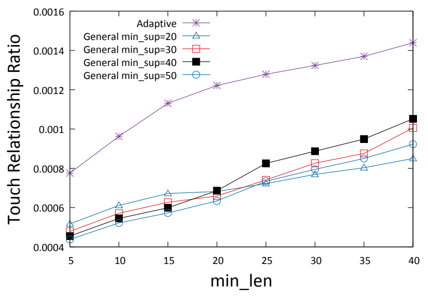
Holistically, the adaptive algorithm outperforms other approaches in road coverage aspect with less trajectory patterns generated.

C. Redundancy Mitigation

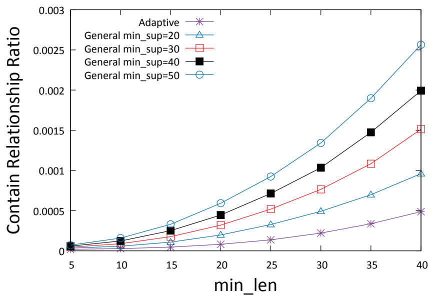
As discussed in Section II and III, a GPS point contains 20 bytes if it is represented in *PT* tuple. However, by using the



(a)



(b)



(c)

Fig. 10: Trajectory pattern mining: (a) road coverage ratio; (b) touch relation ratio; and (c) contain relation ratio

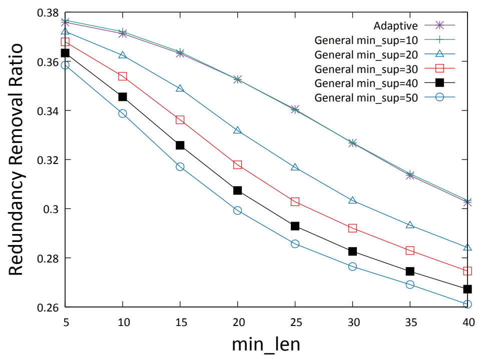


Fig. 11: Redundancy removal ratio and parameter impacts

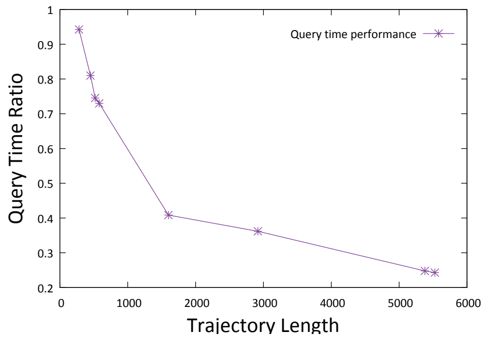


Fig. 12: Query time ratio under *path* Query

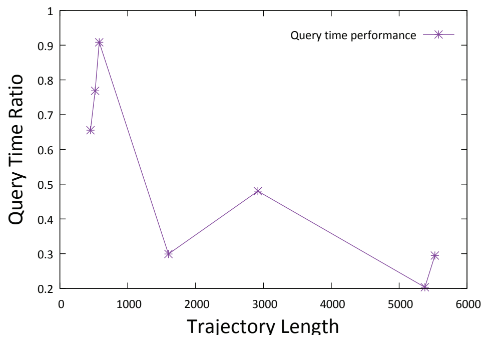


Fig. 13: Query time ratio under *where* Query

MMP, the total storage requirement of representing a location point can be reduced to 16 bytes assuming that data type of *rid* and *t* is int and *dis* is double. In particular, by using *MMP* and *PT*, the lossless redundancy removal ratio will be 1-$\frac{16\eta}{20\eta} = \frac{1}{5}$ (assuming that η is the number of GPS points in *T*, the total storage of *T* in *MMP* formulation will be 16η, and the storage of *T* in *PT* formulation will be 20η). When representing in *NT* and *PT*, the lossless redundancy removal ratio will be:

$$CR = 1 - \frac{4\varphi + 12\varrho + 16\varsigma}{20\eta} \quad (2)$$

where η is the number of GPS points in T and φ is the number of trajectory patterns referred while ϱ is the number of distance-time tuples and ς is the number of *MMP* in *NT*. The total *NT* storage can be calculated by 4φ+12ϱ+16ς, and the resultant *T* will be 20η. Additionally, due to $\varrho + \varsigma = \eta$, the *CR* can be induced to *CR* = $\frac{8}{20}$ - $\frac{(\varphi+\varsigma)}{5\eta}$. In fact,

$$\lim_{\varphi=0,\varsigma=\eta} \frac{\varphi+\varsigma}{5\eta} = \frac{1}{5} \quad (3)$$

$$\lim_{\varphi=1,\varsigma=0,\eta\to\infty} \frac{\varphi+\varsigma}{5\eta} = 0 \quad (4)$$

According to Equation 3 and 4, we can conclude that the ratio is between 20% and 40%. Obviously, the trajectory pattern sets will lead to extra storage cost. Assume min_len=10, adaptive algorithm will generate more than 120,000 trajectory patterns which take around 137 MB space. However, compared with the huge amount of trajectories in the IoV system, the overhead of these trajectory patterns and road network is acceptable and can be neglected.

Figure 11 shows the redundancy removal ratio varies with changes in *min_sup* and *min_len* in context of greedy algorithm. We can see that the redundancy removal ratio can reach about 38%, which is approximate to the theoretical boundary 40%. Based on this, it can be concluded that the greedy algorithm achieves a perfect performance.

D. Query

To evaluate the query performance, we make a comparison between the query efficiency of trajectories stored in trajectory pattern and longitude-latitude tuple methods.

Figure 12 illustrates the effectiveness of *path* query, the QTR decreases with the increment of trajectory length, and the query speed of trajectory based on trajectory pattern is observably faster than trajectory based on longitude-latitude tuple. Because the trajectory pattern has the road list and trajectory *NT* has deviated distances, the method can get the road list and distance directly. For most trajectories within the length of 1,000, comparing to the time of query based on longitude-latitude tuple, the time of path query based on trajectory pattern is merely 60% approximately. Namely the query path based on trajectory pattern can be accelerated by 40%.

Figure 13 demonstrates the comparison of *where* query. Given one location, *where* query is to find all occurring times at the location in one trajectory. The query time ratio QTP varies with trajectories as different trajectories have distinct number of same locations and the query needs different time to calculate. We can observe that the query based on trajectory pattern is generally faster than longitude-latitude tuple due to the road and distance.

In conclusion, the query based on trajectory pattern is faster than conventional trajectory based on longitude-latitude tuple which can attribute to the use of trajectory patterns.

V. Related Work

Trajectory data has been a significant research focus for years and many systems are proposed. Those approaches mainly stored trajectories in spatio-temporal databases. Geo-databases such as PostGIS [3] focus on Geo-data processing and optimizing. Distributed databases such as MD-HBase [16] based on HBase [7] which achieve high insertion and efficient geographical query by using multi-dimension index. These systems basically store the trajectory with longitude-latitude tuple structure. Without considering the high repeatability of vehicle trajectories in real world, such systems often suffer from problems of huge storage redundancy and low query efficiency.

Recently road network has been used for trajectory data storage. A nonmaterialized trajectory model, for example, is proposed in [6]. In this model, the spacial dimension is stored by road network and time dimension is stored as road offset and timestamp. Map matching trajectory compression is introduced in [11]. *Press* [17] processes trajectory by compressing its spatial and temporal information further. More specially, assuming that the trajectory is always along the shortest path, *Press* uses shortest path and Huffman coding to compress the trajectory. However, in the complex road environments, the vehicle trajectory is always changeable and the shortest path is not always suitable. Meanwhile, the adopted lossy compression approach could decrease the query accuracy.

Frequent pattern mining is an important research area in data mining. Frequent patterns can be used to accelerate the path computation on a road network [9]. Trajectory pattern is proposed in [8] and has been used widely. A driving pattern mining algorithm based on traffic flows determining the frequent support is proposed in [9] which needs extra traffic information. Methods in [13] [15] [17] treat trajectory as string and calculate the occurring frequency.

VI. Conclusions and Future Work

In this paper, based on the recognition of trajectory patterns, we have proposed the *SMTP* for storing a very large set of vehicle trajectories. The patterns are recognized according to the geo-space relationships between trajectories. And patterns are stored selectively so as to remove any redundancy included in the original trajectory data. The queries of vehicle trajectories are now pattern-based and allow potential acceleration. We have conducted extensive experiments to evaluate the performance of *SMTP* using real IoV data sets. Experiment results show that storage space for trajectory data can be reduced by 38% while a typical query to the data can be accelerated by approximately 40%. In the future we will continue to explore the possibilities of faster pattern mining methods and more efficient indexing of trajectory patterns.

Acknowledgements

This work is supported by the National Key Research and Development Program (2016YFB1000103), the National 863Project (2015AA01A202), the National Natural Science Foundation of China (61421003), and the Beijing S&T Committee - Beijing Brain Inspired Computing Program in BCBD Innovation Center.

References

[1] Amazon web services. https://aws.amazon.com/.
[2] OpenStreetMap. http://www.openstreetmap.org/.
[3] PostGIS. http://postgis.net/.
[4] Uber. https://www.uber.com/.
[5] White Paper of Internet of Vehicles(IoV). http://mddb.apec.org/Documents/2014/TEL/TEL50-PLEN/14_tel50_plen_020.pdf.
[6] H. Cao and O. Wolfson. Nonmaterialized motion information in transport networks. In *ICDT*, pages 173–188, 2005.
[7] L. George. *HBase: the definitive guide*. " O'Reilly Media, Inc.", 2011.
[8] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *ACM KDD*, pages 330–339, 2007.
[9] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *VLDB*, pages 267–282, 2007.
[10] W. Jiang, T. Wo, M. Zhang, R. Yang, and J. Xu. A method for private car transportation dispatching based on a passenger demand model. In *IoV*, pages 37–48. Springer, 2015.
[11] G. Kellaris, N. Pelekis, and Y. Theodoridis. Map-matched trajectory compression. *Journal of Systems & Software*, 86(6):1566–1579, 2013.
[12] D. E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1998.
[13] A. Kügel and E. Ohlebusch. A space efficient solution to the frequent string mining problem for many databases. *Data Mining and Knowledge Discovery*, 17(1):24–38, 2008.
[14] W. Luo, H. Tan, L. Chen, and L. M. Ni. Finding time period-based most frequent path in big trajectory data. In *ACM SIGMOD*, pages 713–724, 2013.
[15] E. Masciari, G. Shi, and C. Zaniolo. Sequential pattern mining from trajectory data. In *ACM International Database Engineering & Applications Symposium*, 2013.
[16] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi. Md-hbase: A scalable multi-dimensional data infrastructure for location aware services. In *IEEE ICMDM*, volume 1, pages 7–16, 2011.
[17] R. Song, W. Sun, B. Zheng, and Y. Zheng. Press: a novel framework of trajectory compression in road networks. *VLDB*, 7(9):661–672, 2014.
[18] A. Wasilewska. Apriori algorithm. *Lecture Notes, http://www. cs. sunysb. edu/~ cse634/lecture_notes/07apriori. pdf, accessed*, 10, 2007.
[19] H. Wei, Y. Wang, T. Wo, Y. Liu, and J. Xu. ZEST: a Hybrid Model on Predicting Passenger Demand for Chauffeured Car Service. In *ACM CIKM*, 2016.
[20] R. Yang and J. Xu. Computing at massive scale: Scalability and dependability challenges. In *IEEE SOSE*, pages 386–397, 2016.
[21] R. Yang, Y. Zhang, P. Garraghan, Y. Feng, J. Ouyang, J. Xu, Z. Zhang, and C. Li. Reliable computing service in massive-scale systems through rapid low-cost failover. *IEEE Transactions on Services Computing*, 2016.