Influence-Aware Predictive Density Queries Under Road-Network Constraints

Lasanthi Heendaliya(✉), Michael Wisely, Dan Lin, Sahra Sedigh Sarvestani, and Ali Hurson

Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA
{heendaliyal,mwwcp2,lindan,sedighs,hurson}@mst.edu

Abstract. Density query is a very useful query type that informs users about highly concentrated/dense regions, such as a traffic jam, so as to reschedule their travel plans to save time. However, existing products and research work on density queries still have several limitations which, if can be resolved, will bring more significant benefits to our society. For example, we identify an important problem that has never been studied before. That is none of the existing works on traffic prediction consider the influence of the predicted dense regions on the subsequent traffic flow. Specifically, if road A is estimated to be congested at timestamp t_1, the prediction of the condition on other roads after t_1 should consider the traffic blocked by road A. In this paper, we formally model such influence between multiple density queries and propose an efficient query algorithm. We conducted extensive experiments and the results demonstrate both the effectiveness and efficiency of our approach.

1 Introduction

Sitting in road traffic congestion is obviously not a pleasant experience for a traveler. The impacts of traffic congestions indeed expand beyond the inconvenience and include environmental, economical, and safety issues [2,18]. This work aims to find solutions to traffic congestion problems by leveraging mobile devices and their popularity among the community.

Some common strategies related to relieving the traffic congestion problem include providing current traffic information (like Google Maps) [4,9,23] or modeling future traffic conditions with past data [17,20,21]. However, these existing approaches still have several limitations which, if can be resolved, will bring more significant benefits to our society. Specifically, such limitations include the following. First, existing approaches that provide only current traffic information do not offer many options for a driver. Because, based on current information, it may already be too late for some vehicles very close to the traffic congestion to divert to a new route. Therefore, several research works [13] have been proposed to predict traffic conditions with mobile object database queries. Unfortunately, most of them simplify the problem by considering objects moving on Euclidean space rather than under road-network constraints, making them hard to

C. Claramunt et al. (Eds.): SSTD 2015, LNCS 9239, pp. 80–97, 2015.
DOI: 10.1007/978-3-319-22363-6_5

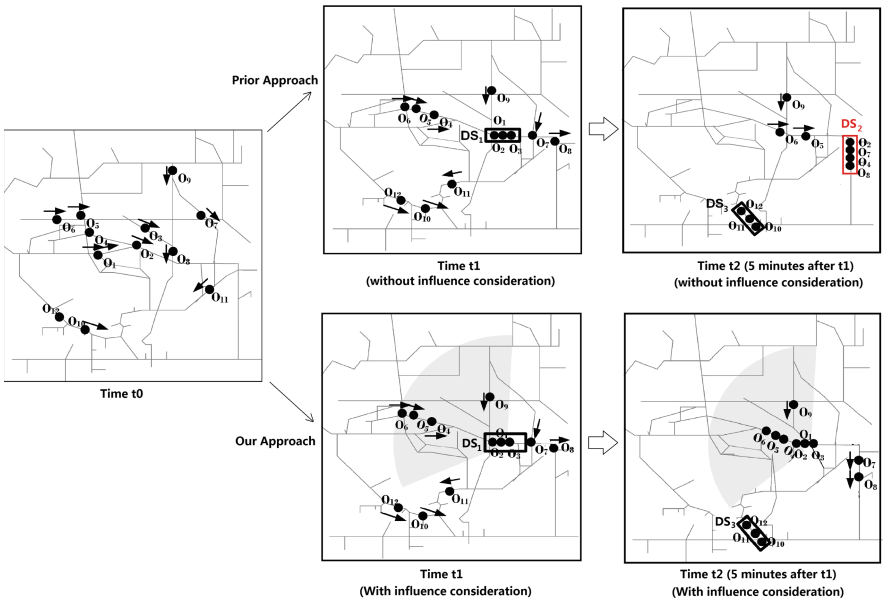


Fig. 1. An example of traffic influence effect

be directly adopted in real scenarios. Very few works predict traffic conditions under the road-network constraints. The most recent one is the Predictive Line Query (PLQ) [11]. However, PLQ only returns predicted traffic information of a user specified road. It would not be able to identify all possible dense regions (i.e., traffic congestions) automatically.

Besides the aforementioned limitations, there is another important issue that has been neglected by all existing works on predictive density queries, which is the influence of a predicted dense region on the subsequent traffic flow. Specifically, if road A is forecasted to be congested at timestamp t, the prediction of the condition on other roads after t should consider the traffic being blocked by road A during the period of congestion. To have a better understanding, let us consider a more concrete example shown in Fig. 1. In Fig. 1, the leftmost diagram shows the positions and moving directions of vehicles (denoted as black points) at timestamp t_0. Without considering traffic influence, a density query will predict a dense road segment DS_1 (highlighted by the rectangle) at timestamp t_1 and then another two (DS_2 and DS_3) at timestamp t_2 (say 5 min later). However, if observed carefully, vehicles stopped by the congestion at DS_1 are unlikely to travel to DS_2 since traffic would not be clear within 5 min. As a result, DS_2 may not have any congestion at all at timestamp t_2. This scenario explains that predicting dense areas on a given timestamp could be inaccurate unless the influence of former possible dense areas are taken into consideration. Our goal is to model such influence (as shown by the gray area) and provide more realistic traffic prediction.

In this paper, we define a new type of query, called *Influence-aware Predictive Density (IPD) Queries*. Our proposed IPD query has the following three key features that are unseen in prior works: (i) it automatically identifies and reports all possible dense areas, in terms of road segments, considering the underlying road network; (ii) it provides predicted traffic density information, which users will find more practical than the current density information; (iii) it accounts for the influence of dense regions on other nearby dense regions to produce more accurate traffic estimation. To efficiently answer the IPD queries, we propose a three-phase query algorithm along with several heuristics to further prune the search space. We have conducted experiments using real road maps and the experimental results demonstrate both effectiveness and efficiency of our approach.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 formally defines the density query problem. Section 4 presents the query algorithm. Section 5 reports the experimental results and Sect. 6 concludes the paper.

2 Related Work

Generally speaking, a density query aims to retrieve all regions with a density (i.e., the number of moving objects per square unit) that exceeds a given threshold. It is worth noting that the density query is different from the range query in that the input to the range query is the location of a query region, whereas the input to the density query is the density threshold and the size (but not the location) of the dense region. The remainder of this section discusses the past work on the density query problem and its solutions.

Existing works on density queries can be roughly classified into two categories: (i) density queries in Euclidean space; (ii) density queries in road networks. Most of existing works on density queries mainly consider objects moving freely in Euclidean space. The first work is by Hadjieleftheriou et al. [16] who proposed two versions of density queries: Snapshot Density Queries (SDQ) and Period Density Queries (PDQ). The SDQ identifies dense regions for a specific time instance in the future, while the PDQ identifies dense regions in a time interval. In their approach, the entire space is divided into a grid of equal-sized cells, and density regions are reported in terms of cells. Such approach ignores possible dense regions located in the middle of multiple cells and causes a so-called answer loss problem as pointed out by Jensen et al. in [13]. To resolve the answer loss problem, Jensen et al. [13] redefined the density query and propose a two-phase query algorithm to predict dense regions that can be located anywhere and are not constrained to partitioning cells. Unlike previous works where the dense regions are square shaped, Ni and Ravishankar [19] define a pointwise dense regions (PDR) that allows the dense region to be of any shape and any size. They also partition the space into grid while their search algorithm ensures that a 4-cell block is searched each time and hence also avoids the answer loss problem. All the aforementioned query algorithms consider the snapshot version

of the density query. The algorithms proposed in [10,22] support the continuous density queries. Similar to the snapshot queries, the continuous density queries also take the density threshold and size of the dense region as input. In order to continuously identify dense regions, the algorithm repeatedly divides the entire space into quadrants until the quadrant is no larger than the given size of the dense region. Then, each quadrant is labeled as either dense or not. This is the main limitation of such an approach.

In addition to the aforementioned density query algorithms, to which movements are considered in the Euclidean space, very few works [15] have been conducted on density queries restricting the movements to road networks. One such work is by Lai et al. [15] who propose the Effective Road-Network Density Query (e-RNDQ). The definition of density under road network constraint is now the number of moving objects per road segment rather than per square unit. Furthermore, the distance between any two neighboring objects in a dense road segment should not exceed the given distance threshold. This condition prevents skewed object distribution in a query result. Lai et al. propose a clustering-based algorithm to obtain the query results. The main limitation is that they only identify dense road segment at current timestamp but do not support any predictive queries.

In summary, our work is different from existing works in the following aspects: (i) it is the first time that the traffic influence is considered during multiple dense region exploration; (ii) it is the first work, to the best of our knowledge, that supports predictive density queries on road networks.

3 Problem Statement

Without loss of generality, we consider uni-directional or bi-directional roads with separate lanes for each direction. In other words, it is assumed that the high traffic density of one direction does not affect the traffic on the other direction. Under this assumption, in what follows we formally define our proposed *Influence-aware Predictive Density Query.*

Definition 1. *[Density] The density of a road segment r is the number of objects per unit length (e.g., meter) per lane on the road, represented as $DS(r) = \frac{N}{m \cdot len(r)}$, where N is the number of objects on road r, $len(r)$ is the length of road r, and m is the number of lanes.*

Definition 2. *[Dense Road Segment] Given a density threshold ρ, the road segment r is dense if the density(r) at time t is greater than the threshold ρ.*

We now proceed to define a new concept, namely *mutually independent dense road segments*, which is the base of the influence-aware predictive density query.

Definition 3. *[Mutually Independent Dense Road Segments] Given any two dense road segments R_a and R_b with occurrence time t_a and t_b ($t_a < t_b$) respectively, R_a and R_b are mutually independent of each other if one of the following conditions is satisfied:*

1. $t_b - t_a > \sigma$, *where* σ *is the threshold that describes the typical time taken to clear a traffic jam;*
2. *For any* $o \in O_a$, $Dist(o, R_b) > v_{max} \cdot (t_b - t_a)$, *where* O_a *is the set of objects on road* R_a *at time* t_a, $Dist(o, R_b)$ *computes the shortest road-network distance between object* o *and the closer end of road* R_b, *and* v_{max} *is the maximum moving speed of objects.*

The definition of mutually independent dense road segments aims to ensure that objects that contribute to the density of one road segment will not affect the density computation of another road. Specifically, the first condition checks if the occurrence of dense regions are far enough apart in terms of time that objects stuck on road R_a may already be freed when computing the density for road R_b. The second condition avoids considering any subsequent dense road segment caused by the similar set of objects that recently contributed to a dense road segment. For example, consider an object o passes by road R_a at t_a and then R_b at t_b. If R_a is predicted as a dense road segment at t_a, we should not consider o when computing the density of road R_b at t_b since o has stuck or been slowed down by the traffic congestion.

Definition 4. *[Influence-aware Predictive Density (IPD) Query] Given a road map* G, *a density threshold* ρ *and a time window* t_{max}, *an IPD query computes a list of predicted mutually independent dense road segments* $\{DS_1, DS_2, DS_3, ..., DS_n\}$, *where the occurrence times* $t_1 \leq t_2 \leq t_3 \leq \cdots \leq t_n (t_n \leq t_{max})$; t_i *is the occurrence time of* DS_i.

It is worth noting that dense road segments caused by the moving objects that have already been accounted for in antecedent dense road segment are excluded from further consideration in IPD queries. In other words, only the earliest occurring dense road segment of a chain of dense segments is considered each time.

4 Influence-Aware Predictive Density Query Algorithm

In this section, we first introduce the data structure that is utilized to support the influence-aware predictive density (IPD) queries and then elaborate the query algorithm.

4.1 Data Structure

To answer IPD queries, we need indexing structures to manage the information of objects moving on road networks. There have been several such kind of indexes such as IMORS [14], ANR-tree [6], R-TPR$^{\pm}$-tree [8], and TPR^{uv} [7]. Here, we employ the most recent one, the R^D-tree [12], which is also our prior work. Although our main contribution of this work lies in the query algorithm (in Sect. 4.2) and not the data structure, we briefly introduce the data structure here to facilitate a better understanding of the whole algorithm.

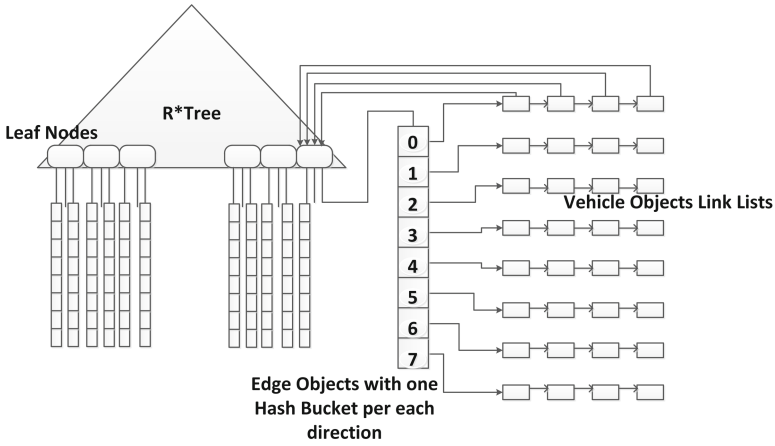


Fig. 2. The R^D-tree

The R^D-tree indexes two types of data: road-network information and object location information. The road network is represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Each edge $e = \{\mathbb{v}_1, \mathbb{v}_2\} \in \mathcal{E}$ represents a road segment in the network where $\mathbb{v}_1, \mathbb{v}_2 \in V$; $\mathbb{v}_1$ and $\mathbb{v}_2$ are starting and ending nodes of the road segment, respectively. Furthermore, each road segment is associated with two parameters: l and s, where l is the length of the edge and s is the maximum possible speed on that edge.

A moving object O is represented by the tuple $\{o_{id}, x_t, y_t, o_e^t, o_e^{t+1}, o_v^t, o_{gd}, t\}$, where o_{id} is the unique object ID, x_t and y_t are the coordinates of the moving object at the latest update timestamp t, o_e^t is the current road segment that the object is on, o_e^{t+1} is the next road segment that the object is headed to, o_v^t is the object's velocity (or speed), and o_{gd} is the object's travel destination. It is assumed that most moving objects are willing to disclose their tentative traveling destinations to the service provider (server) in order to obtain high-quality services, albeit their destinations may change during the trip.

Figure 2 illustrates the overall structure of the R^D-tree. The R^D-tree is composed of an R*-tree [3] and a set of hash tables. The road-network information is indexed by the R*-tree. Each entry in the non-leaf node is in the form of $(node\_MBR, child\_ptr)$, where $node\_MBR$ is the MBR (Minimum Bounding Rectangle) covering the MBRs of all entries in its children pointed to by the $child\_ptr$. Leaf nodes in R*-tree pointing to hash tables represent moving objects on each road segment. Each entry in the leaf node is in the form of $(edge\_MBR, obj\_ptr)$, where $edge\_MBR$ is the MBR of a road segment and obj_ptr links to a hash table storing objects moving on this edge. Each hash table has an N_d hash bucket, where N_d is the number of traveling directions. Each bucket has two linked lists that provide a finer grouping for objects based on their traveling directions. Moving objects with similar traveling directions are hashed to the same hash bucket and stored in one of the sorted linked lists

maintained in that hash bucket. Moreover, for easy update, each object also has a pointer directly linked to the edge that it is currently moving on. The details of the construction of the R^D-tree can be found in [12].

In addition to the R^D-tree, we also maintain a two-dimensional histogram that comprises of square-shaped cells covering the considered space. Each cell maintains the counts of moving objects that may cross the cell within the time period $[t_{now}; t_{now} + H]$ for equally calibrated timestamps. Here H is the horizon – the time window in which the prediction is valid. The histogram is initialized according to the moving object's estimated traveling path.

4.2 Query Algorithm

Influence-aware predictive density (IPD) queries aim to identify all dense road segments that may occur at different timestamps in the near future and also do not influence each other as defined in Sect. 3. The query algorithm consists of three phases: filtering, refinement, and refreshing.

The Filtering Phase. The filtering phase utilizes the histogram to quickly identify potential grid cells that may contain dense road segments. Recall that the histogram stores the estimated number of objects in the corresponding cell at each timestamp starting from the current timestamp. When considering whether a cell may contain a dense road segment, we do not simply use the original count of objects in the cell, as in previous works. Instead, we consider the *adjusted cell density* (Definition 5) in order to take into account the road topology. Specifically, the adjusted cell density estimates the average number of objects per unit length of road segments. For example, if a cell contains very few roads but a large number of objects, it is very likely that the cell contains a dense road segment. For time efficiency, the adjusted cell density can be generated along with the computation of the count of objects when building the histogram. For storage efficiency, the histogram can be compressed similarly as that in [13].

Definition 5. *[Adjusted Cell Density] Let N_c^t be the number of objects in cell c at timestamp t, and l_c^t be the total length of road segments in cell c. The adjusted cell density ACD_c^t is computed as follows:*

$$ACD_c^t = \frac{N_c^t}{l_c^t}$$

The filtering phase starts checking the adjusted cell density of each cell at the earliest timestamp stored in the histogram. If a cell's density is above a threshold ρ_c, the cell will be inserted to a priority queue to be sent to the refinement phase. The challenging issue here is to determine the proper value of the threshold ρ_c. If we simply set ρ_c to the same value as the density threshold ρ given by the query issuer, we may miss the dense road segment that spans multiple non-dense cells and have many false negatives. If we set ρ_c to a very low value,

we will not miss the dense road segment but the filtering phase will lose the pruning power and keep too many non-dense cells for further examination, which in turn will increase the overall query cost. Therefore, we model this effect using the following linear regression function. The goal is to identify the best value of ρ_c that balances both the query cost and the number of false negatives. Specifically, in Eq. 1, $Cost_q$ and FN are the estimated query cost and false negatives for a given ρ_c, α is a weight value, and $Cost_{max}$ is the query cost to retrieve all objects in the entire space. Here, $Cost_q$ is the estimated cost of the second refinement phase which is determined by the number of queries on cells in the queue. The lower the ρ_c, the fewer the cells to be further examined and lower $Cost_q$. $Cost_{max}$ is used to normalize the value of the first part of the equation to the range of 0 and 1, so that it is comparable to the false negatives. FN is estimated using the number of road segments spanning cells. The goal is to minimize the "Penalty" to determine ρ_c.

$$Penalty(\rho_c) = \alpha \cdot \frac{Cost_q}{Cost_{max}} + (1 - \alpha) \cdot FN \tag{1}$$

After all cells at the same timestamp are considered, we move to the next timestamp and the same filtering process continues until all timestamps in the histogram have been examined. The final priority queue will have a list of cells ordered in an ascending order of the timestamps. Cells at the same timestamp are ordered by descending density.

More importantly, each cell c in the queue also maintains a list of influenced cells whose priority is lower than its own, along with the number of vehicles coming from the cell c to the influenced cell. This list will later be used to quickly prune non-dense cells as discussed in the refreshing phase.

The Refinement Phase. The refinement phase takes a further look at the candidate dense cells obtained from the filtering phase to see whether these cells actually contain dense road segments. The refinement phase starts from the highest prioritized candidate cell from the priority queue and moves to the next highest prioritized cell of the same timestamp, and so on. After all candidate cells at the same timestamp are examined, the *refreshing phase* (Sect. 4.2) will be activated in which the entries in the queue and their priority will be updated. Then the highest prioritized cell from the updated queue is selected and sent back to the refinement phase. The iteration between the refinement and refreshing continues until the priority queue is empty. In what follows, we elaborate the three main steps taken during the refinement.

The first step is to find the road segments containing the objects that may pass by the candidate cell at the timestamp t_d (as stored in the priority queue) when this cell may be dense. Note that these road segments may not be the road segments located in the candidate cell. This is because we are predicting future dense road segments, and objects on road segments outside the candidate cell at current timestamp may enter this cell at a future timestamp. In order to

identify these related road segments, we perform a square-shaped ring query on the R^D-tree as shown in Fig. 3, where the square shaped ring is represented by the shaded area between solid-line squares. The dimension of the square-shaped ring is determined according to the road network information. Specifically, the lengths $innerL$ and $outerL$ are the distances to the closest and the farthest objects that may be able to enter the candidate cell according to the road speed limits.

$$innerL = \frac{cell\ width}{2} + Dist(speed_{min} \cdot (t_d - t_c)) \tag{2}$$

$$outerL = \frac{cell\ width}{2} + Dist(speed_{max} \cdot (t_d - t_c)). \tag{3}$$

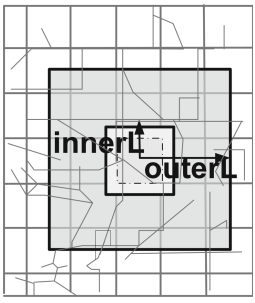


Fig. 3. Squared shaped ring query

The second step is to retrieve objects in the relevant hash bucket of each road segment found by the previous step. Recall that each road segment is associate with multiple buckets containing objects traveling towards different destinations. Intuitively, if an object is not heading toward the candidate cell, we do not need to retrieve it. To determine which bucket needs to be checked, we consider the relevancy of the object's traveling angle and the candidate cell as shown in Fig. 4. The Fig. 4 illustrates two cases where the number of hash buckets is 8 and different distances to the candidate cell (due to the difference in times when the density is computed for: t_a and $t_b > t_a$). As shown in the figure, the number of hash buckets selected to examine candidate cell A's density is 3 (hash bucket 1, 2, and 3) where that for candidate cell B is only two (bucket 0 and 1). The use of bucket selection helps improve the overall query performance by pruning irrelevant objects for further consideration.

Since objects returned from the second step may still contain objects that do not contribute to any dense road segment, the final refinement step is to compute the exact traveling routes of these candidate objects and then identify the truly dense road segments. Specifically, each road segment in the candidate cell is associated with a counter. For each candidate object's path, we increase the counter of the road segment passed by the object by one. This ensures that each

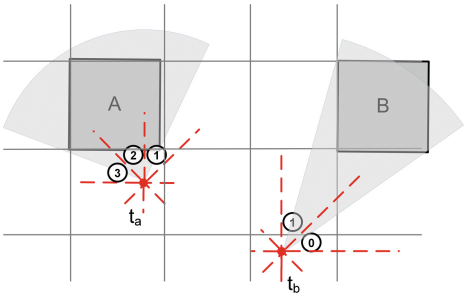


Fig. 4. Two examples of modified hash bucket selection

path is examined only once. After analyzing all the candidate objects' routes, the road segments with count above the density threshold will be reported.

The Refreshing Phase. The refreshing phase aims to compute quarantine areas of identified dense road segments. Since objects occurring on one dense road segment are impossible to occur on another at the same time, it is not necessary to run refreshing phase after each identified road segment. Instead, the refreshing runs after all the dense road segments have been identified for each timestamp considered to improve efficiency. It consists two main steps: (i) compute quarantine areas; (ii) rejuvenate the priority queue.

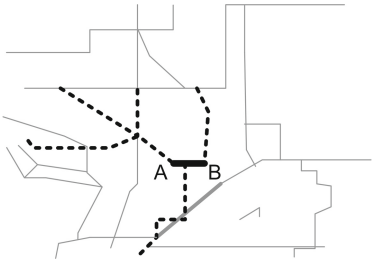
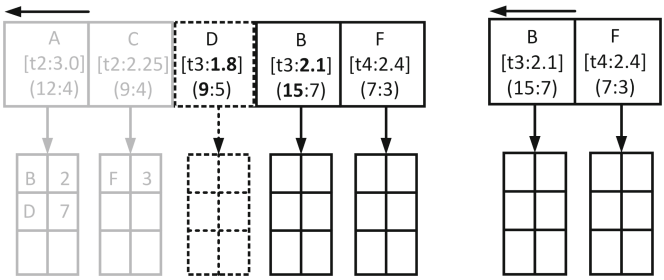


Fig. 5. Influenced road segments in the quarantine area of road $\overline{AB}$

A *quarantine area* is defined for each identified dense road segment within the same timestamp. The area contains the dense road segments and the segments that the congestion would propagated to. Figure 5 shows an example of the quarantine area regarding a dense road segment $\overline{AB}$ at time t_a. The dashed-lined road segments are the road segments that will be affected by $\overline{AB}$. More specifically, the road segments in the quarantine area contain objects stuck in $\overline{AB}$ at time t_a for the computation of their density at a near future timestamp t_b when the traffic may not be cleared. Therefore, the computation of density of road segments in the quarantine area should ignore the objects stuck in $\overline{AB}$.



(a) Original Priority Queue (b) Updated Priority Queue

Fig. 6. Updated queue

The objects on the road segments in the quarantine area are disregard from subsequent dense area identification. The formal definition of quarantine area is given by Definition 6, where the value of n is determined based on the typical time that a traffic congestion can be cleared.

Definition 6. *[Quarantine Area] Given a road network* $\mathcal{G}(\mathcal{V}, \mathcal{E})$ *and a set of dense road segments* $\mathcal{S}$*; where* $\mathcal{S} \subset \mathcal{E}$*. The quarantine area of* $\mathcal{S}$ *is a set of road segments,* $\mathcal{Q} = \bigcup_{i \in |\mathcal{S}|}(\mathcal{S}_i \cup \mathcal{S}_i^n)$*. Here* $\mathcal{S}_i^n$ *is the* n *-hop adjacent edges of* i^{th} *edge in* $\mathcal{S}$*.*

The second step of the refreshing is to rejuvenate the priority queue by discarding cells influenced by the identified dense road segment. This step leverages the influence cell list associated with each cell in the priority queue. Specifically, for each cell c that contains the identified dense road segment at the timestamp considered, we update the adjusted density for the cells that overlap with the quarantine area of cell c by decreasing the corresponding number of objects stuck in c. Figure 6 shows an example. Suppose that after the refinement phase, we know that cell A contains dense road segments while cell C does not. Since cell C has no dense road segments in it, there is no need to update its entry in the priority queue. As for cell A, it influences two cells B and D. Cell B contains two objects that will travel from cell A. Since the objects in cell A are stopped due to the high density of cell A, the total number of expected objects in cell B would be decreased, and the new adjusted density of cell B becomes $15/7 = 2.1$. Similarly, cell D's new density is 1.8. Assuming that the cell D's density is now below the density threshold ρ_c, it would be removed from the priority queue for subsequent computations. In this way, the refreshing phase helps avoid unnecessary computations.

5 Performance Study

In this section, we evaluate the performance of our proposed influence-aware predictive density (IPD) query algorithm by varying a number of parameters

on different datasets. Since the IPD query is the only approach that predicts dense road segments under road network constraints, we compare it with a baseline approach which simply examines each object's shortest path at different timestamps to directly compute the dense road segments. The baseline approach also implements the concept of quarantine area by discarding the objects on identified dense road segments from subsequent computation. Both algorithms were implemented and tested on a 2.40 GHz Intel®Xeon®E5620 CPU desktop with 11 Gigabytes of memory. The page size is 4 K bytes. The R^D-tree implementation of R^D-tree in our approach is the same as that in [12][1]. The internal nodes of a tree are pinned in a LRU buffer of 50 pages.

Table 1. Statistics of the road maps

State	Land area	Number of road segments	Average road segment length
IA	55,857	3392	356
AZ	66,455	4935	383
WA	113,594	1442	628
CA	155,779	8062	225

The datasets used for testing are generated by the commonly adopted Brinkhoff generator [5]. The generator was fed with four different US state maps: IA, WA, AZ, and CA. The states differ in total land area, number of road segments, and average road segment length, which results in different mobile object distributions. The statistics of the chosen states are given in Table 1. The number of moving objects in each dataset ranges from 10 K to 100 K. Average traveling time of each data set is 60 min. The chosen input parameters and their values are presented in Table 2 with the default value in bold. The efficiency and effectiveness are measured in terms of the average I/O cost (i.e., the number of page accesses) and the number of identified dense road segments from current time to the query life (i.e., the predictive time window), respectively.

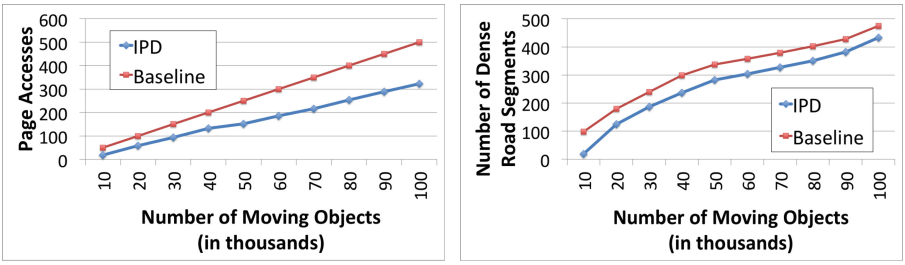
5.1 Effect of Number of Moving Objects

In the first round of experiments, we evaluate the query performance by varying the number of moving objects from 10 K to 100 K while keeping other parameters as default (in Table 2). As shown in Fig. 7(a), it is expected that the query cost increases with the number of moving objects since more data need to be retrieved from disk and examined. Our IPD query algorithm significantly outperforms the baseline approach and the performance gap increases with the number of moving objects. This can be attributed to our proposed filtering algorithm and bucket selection algorithm that help reduce the number of objects to be examined and

[1] R^D-tree adopts its R*-tree simulator from [1].

Table 2. Parameters and their values

Parameters	Values
Number of mobile objects	10 K, 20 K, ..., **50 K**, ..., 100 K
Road network topology	IA, AZ, **WA**, CA
Predictive time window (minutes)	10, 20, **30**, 50
Cell density threshold (ρ_c)	**0.05**, 0.1, 0.15, ..., 1
Road density threshold (ρ)	0.2, 0.4, 0.6, 0.8,**1**
Grid size (d)	10, 20,**30**, 40, 50
Vehicles equipped with the system	25 %, 50 %, 75 %, **100 %**
Timestamp interval (minutes)	**5**



(a) I/O cost (b) Number of Dense Road Segments

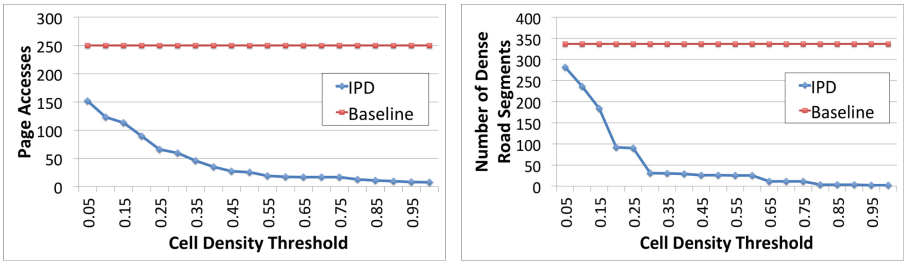
Fig. 7. Effect of the number of moving objects

the refreshing phase that further prunes the candidate cells which have been influenced by the identified road segments.

Figure 7(b) compares the number of dense road segments found by the two approaches. From the figure, we can observe that our IPD query reports fewer number of dense road segments than the baseline approach. This is expected as discussed in Sect. 4.2. Specifically, we adopt a cell density threshold in the filtering phase. Higher ρ_c will prune more cells and yield better query performance but may introduce false negatives. Fortunately, we also observe that the percentage of such difference becomes smaller for bigger datasets. In a small dataset, such effect is more severe since missing one object may make a road segment to be non-dense easily. In the next experiment, we will take a closer look at how the cell density threshold affect the performance.

5.2 Effect of Cell Density Threshold

We now study the effect of cell density threshold ρ_c. Recall that ρ_c is used to prune those cells that are highly unlikely to contain dense road segments. As aforementioned, higher ρ_c may yield fewer number of candidate cells to be further examined and hence reduce query cost. However, it is possible that when



(a) I/O cost (b) Number of Dense Road Segments

Fig. 8. Effect of the cell density threshold

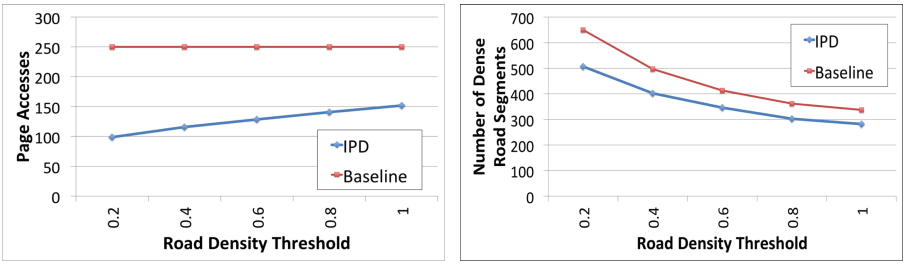
ρ_c is high, the cells with low density and contain part of a dense road segment (i.e., a dense road segment that spans multiple cells) may be left out, resulting in false negatives. Figure 8 reports the experimental results when varying the cell density threshold from 0.05 to 1. As expected, the query cost of our approach decreases quickly when the cell density threshold increases. Meanwhile the number of missing dense road segments increases. In order to minimize false negatives, we adopt the cell density threshold as 0.05 in our experiments.

5.3 Effect of Road Density Threshold

In this set of experiments, we study the effect of the road density threshold given by the query. The results are shown in Fig. 9. We can observe that our IPD algorithm again outperforms the baseline approach in terms of query efficiency and identified similar number of dense road segments in all cases. Moreover, we also observe that the query cost of our IPD algorithm increases with the road density threshold while the baseline approach has constant performance. This is because the baseline approach always checks all objects' travel paths when computing the density of road segments. In our approach, the higher the road density threshold, the fewer the number of dense road segments. Correspondingly, there will be fewer number of quarantine areas and fewer number of objects that can be pruned.

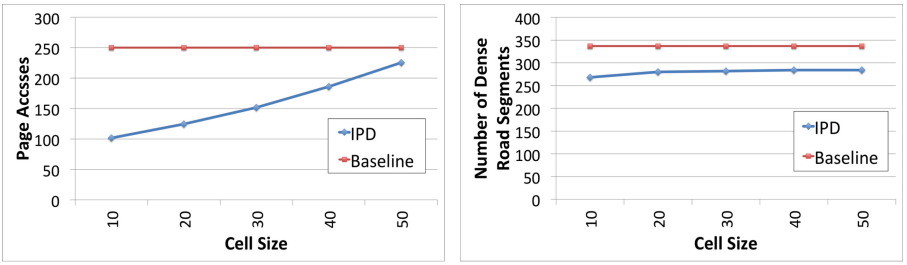
5.4 Effect of Cell Size

We proceed to study the effect of the cell size. As shown in Fig. 10, the cell size does not affect the baseline approach since the baseline approach does not utilize cells for pruning. As for our IPD algorithm, it achieves better performance when the cell size is small. This is probably because the adjusted density of smaller cells is more useful for determining whether there is a potential dense road segment. When the area of a cell is large, the road topology in the single cell becomes more complex and the pruning becomes less effective. On the other hand, when the cell is large, there is fewer chances to have dense road segments across multiple



(a) I/O cost (b) Number of Dense Road Segments

Fig. 9. Effect of the road density threshold



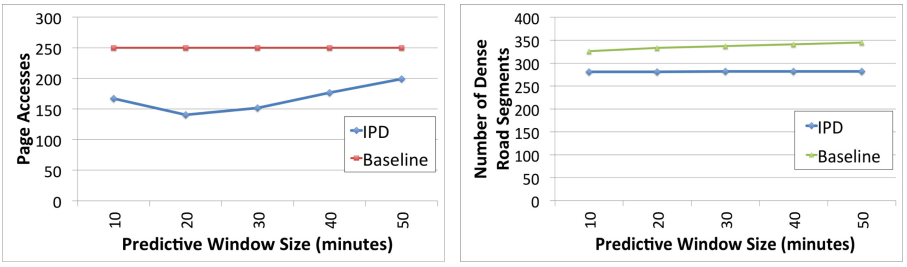
(a) I/O cost (b) Number of Dense Road Segments

Fig. 10. Effect of the cell size

cells and hence it helps reduce false negatives. In our experiments, we choose the cell size to be 30 as it balances both performance and accuracy.

5.5 Effect of Predictive Time Window

The effect of predictive time window was also studied. The results are shown in Fig. 11. It also shows the efficiency of the IPD algorithm. In fact, the IPD has



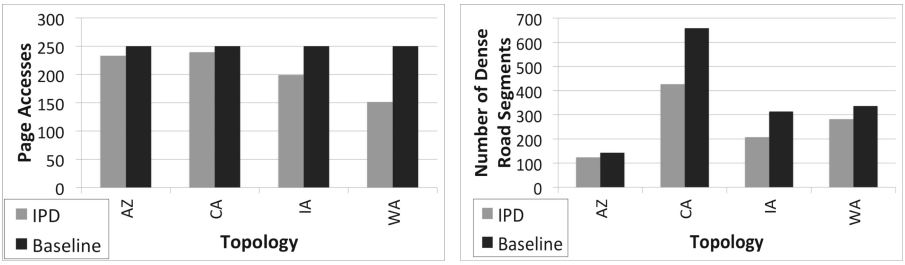
(a) I/O cost (b) Number of Dense Road Segments

Fig. 11. Effect of the predictive time window

lower page accesses compared to that of the baseline algorithm. The IPD shows an increase in the page accesses for longer predictive window lengths. This can be expected as the area covered by the square-ring is also increased. The IPD exhibits fewer unidentified road segments. In fact, the number of unidentified dense road segments increases when the predictive time window is lengthier. It is again due to relaxation of the cell threshold.

5.6 Effect of Road Network Topology

This round of experiments evaluate the effect of the road topology on the query performance. As shown in Fig. 12, our approach always achieves better query efficiency than the baseline approach when different road maps are considered. In addition, the results also indicate that the map topology does affect the performance. In general, maps with fewer roads across multiple cells tend to yield better performance. Moreover, we also observe that the numbers of dense road segments identified by the two approaches demonstrate the same trend, which proves the effectiveness of our approach.

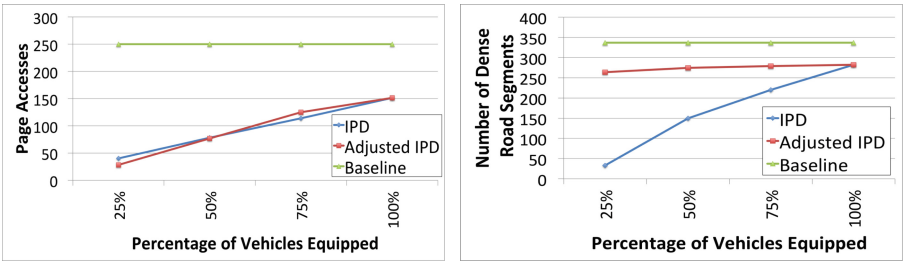


(a) I/O cost (b) Number of Dense Road Segments

Fig. 12. Effect of the road network topology

5.7 Effect of Percentage of Vehicles Equipped with the System

In the last set of experiments, we aim to examine an interesting and realistic scenario when not all vehicles subscribe to traffic prediction services. That means, the system will estimate the traffic based on a subset of vehicles. It is expected the fewer the number of vehicles equipped with the system, the less accurate the traffic prediction will be. To compensate for the missed information, we adjust the system parameters by lowering both the cell density threshold and road segment density threshold. For example, given a density threshold $\rho = 1$, we set $\rho_c = 0.0375$ and $\rho' = 0.75$. Our findings are reported in Fig. 13, where "Adjusted IPD" refers to the approach with adjusted new threshold. We can observe that the adjusted IPD has similar query cost as the original one but much better accuracy in terms of the number of dense road segments being identified.



(a) I/O cost (b) Number of Dense Road Segments

Fig. 13. Effect of the percentage of vehicles equipped with the system

6 Conclusion

In this paper, we define a new type of density query, namely Influence-aware Predictive Density (IPD) queries, with the goal to take into account the impact of a traffic congestion on the traffic flow, i.e., objects stuck in the traffic congestion should not be counted into the subsequent traffic prediction before the traffic is cleared. To the best of our knowledge, it is the first time that traffic influence is considered for predicting potential traffic congestions under the road network constraints. We propose an efficient query algorithm that leverages multiple pruning techniques. Our experimental results have demonstrated both the efficiency and effectiveness of our approach compared with the baseline approach.

References

1. Achtert, E., Kriegel, H.-P., Schubert, E., Zimek, A.: Interactive data mining with 3d-parallel-coordinate-trees. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2013)
2. Barth, M., Boriboonsomsin, K.: Real-world carbon dioxide impacts of traffic congestion. Transport. Res. Rec. J. Transport. Res. Board **2058**, 163–171 (2008)
3. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles (1990)
4. Bok, K.S., Yoon, H.W., Seo, D.M., Kim, M.H., Yoo, J.S.: Indexing of continuously moving objects on road networks. IEICE Trans. Inf. Syst. **E91–D**, 2061–2061 (2008)
5. Brinkhoff, T.: A framework for generating network-based moving objects. GeoInformatica **6**, 153–180 (2004)
6. Chen, J.-D., Meng, X.-F.: Indexing future trajectories of moving objects in a constrained network. J. Comput. Sci. Technol. **22**(2), 245–251 (2007)
7. Fan, P., Li, G., Yuan, L., Li, Y.: Vague continuous K-nearest neighbor queries over moving objects with uncertain velocity in road networks. Inf. Syst. **37**(1), 13–32 (2012)

8. Feng, J., Lu, J., Zhu, Y., Mukai, N., Watanabe, T.: Indexing of moving objects on road network using composite structure. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part II. LNCS (LNAI), vol. 4693, pp. 1097–1104. Springer, Heidelberg (2007)
9. Feng, J., Lu, J., Zhu, Y., Watanabe, T.: Index method for tracking network-constrained moving objects. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part II. LNCS (LNAI), vol. 5178, pp. 551–558. Springer, Heidelberg (2008)
10. Hao, X., Meng, X., Xu, J.: Continuous density queries for moving objects. In: Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE 2008 (2008)
11. Heendaliya, L., Lin, D., Hurson, A.: Continuous predictive line queries for on-the-go traffic estimation. In: Hameurlain, A., Küng, J., Wagner, R., Decker, H., Lhotska, L., Link, S. (eds.) TLDKS XVIII. LNCS, vol. 8980, pp. 80–114. Springer, Heidelberg (2015)
12. Heendaliya, L., Lin, D., Hurson, A.R.: Predictive line queries for traffic forecasting. In: Database and Expert Systems Applications (2012)
13. Jensen, C.S., Lin, D., Beng, C.O., Zhang, R.: Effective density queries on continuously moving objects. In: Proceedings of the 22nd International Conference on Data Engineering (2006)
14. Kyoung-Sook, K., Si-Wan, K., Tae-Wan, K., Ki-Joune, L.: Fast indexing and updating method for moving objects on road networks. In: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops (2003)
15. Lai, C., Wang, L., Chen, J., Meng, X., Zeitouni, K.: Effective Density queries for moving objects in road networks. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J.X. (eds.) APWeb/WAIM 2007. LNCS, vol. 4505, pp. 200–211. Springer, Heidelberg (2007)
16. Gunopulos, D., Hadjieleftheriou, M., Kollios, G., Tsotras, V.J.: On-line discovery of dense areas in spatio-temporal databases. In: International Symposium on Advances in Spatial and Temporal Databases, SSTDn (2003)
17. Min, W., Wynter, L.: Real-time road traffic prediction with spatio-temporal correlations. Transp. Res. Part C **19**(4), 606–616 (2011)
18. Morgan, L.: The effects of traffic congestion (2014)
19. Ni, J., Ravishankar, C.V.: Pointwise-dense region queries in spatio-temporal databases. In: IEEE 23rd International Conference on Data Engineering (2007)
20. Quek, C., Pasquier, M., Lim, B.B.S.: Pop-traffic: a novel fuzzy neural approach to road traffic analysis and prediction. IEEE Trans. Intell. Transp. Syst. **7**(2), 133–146 (2006)
21. Smith, B.L., Williams, B.M., Oswald, R.K.: Comparison of parametric and non-parametric models for traffic flow forecasting. Transp. Res. Part C **10**(4), 303–321 (2002)
22. Wen, J., Meng, X., Hao, X., Xu, J.: An efficient approach for continuous density queries. Front. Comput. Sci. **6**(5), 581–595 (2012)
23. Yiu, M.L., Tao, Y., Mamoulis, N.: The Bdual-tree: indexing moving objects by space filling curves in the dual space. VLDB J. **17**(3), 379–400 (2008)