Online Hot Spot Prediction in Road Networks

Maik Häsner Conny Junghans Christian Sengstock Michael Gertz

Institute of Computer Science
University of Heidelberg, Germany

Abstract: Advancements in GPS-technology have spurred major research and development activities for managing and analyzing large amounts of position data of mobile objects. Data mining tasks such as the discovery of movement patterns, classification and outlier detection in the context of object trajectories, and the prediction of future movement patterns have become basic tools in extracting useful information from such position data. Especially the prediction of future movement patterns of vehicles, based on historical or recent position data, plays an important role in traffic management and planning.

In this paper, we present a new approach for the online prediction of so-called *hot spots*, that is, components of a road network such as intersections that are likely to experience heavy traffic in the near future. For this, we employ an efficient path prediction model for vehicle movements that only utilizes a few recent position data. Using an aggregation model for hot spots, we show how regional information can be derived and connected substructures in a road network can be determined. Utilizing the behavior of such hot spot regions over time in terms of movement or growth, we introduce different types of hot spots and show how they can be determined online. We demonstrate the effectiveness of our approach using a real large-scale road network and different traffic simulation scenarios.

1 Introduction

Driven by major advancements in GPS-based technologies, position data of mobile objects have become ubiquitous. Through tracking systems for vehicles, persons, and even animals, enormous amounts of object trajectory data are being collected and subject to various data mining tasks. The objectives of these tasks include the discovery of (periodic) movement patterns, the classification of objects based on their trajectories, and the prediction of future movement patterns based on the objects' historical trajectories (see the tutorial by Han et al. [HLT10] for an excellent overview).

In the past couple of years, especially the analysis of traffic data has been of great interest as results obtained by analyzing past and current vehicle position data provide important input to traffic management, control, and planning. One key feature of respective approaches is the detection of *hot spots*, that is, road segments and intersections that experience a high load of traffic and thus require special attention in traffic management.

One can roughly distinguish three classes of approaches to the discovery of hot spots, based on what object position data is used and whether the detection refers to past, current, or future hot spots. The first class of approaches apply real-time monitoring where only current vehicle position data is used to determine hot spots based on the density of vehicle positions on a given road network. This approach, where no predictions are made, is employed by several online traffic monitoring systems. The second class of approaches perform an analysis of historical trajectory data to determine (periodic) hot spots that occurred in the past and make predictions of future hot spots that likely occur, e.g., during rush hour [JYZJ10, LHLG07]. The third class utilizes only current position data, perhaps including a few previous object position data, and they try to predict near future vehicle paths and hot spots (e.g., [KRSZ08]). Especially this class is of interest in online scenarios where no information about historic object trajectories is available and near time traffic predictions have to be made to mitigate congestion in a timely manner.

In this paper, we present such an online approach for the prediction of near future traffic hot spots based on recent positions of vehicles moving in a road network. Our approach does not require historic object trajectories but makes predictions based on direction information derived from the most recent and current positions of vehicles, similar to the approach by Kriegel et al. [KRSZ08]. Through the prediction of possible future positions of vehicles, nodes in the road network (representing, e.g., intersections) are assigned a weight that reflects the likely traffic intensity within a specified time horizon. Instead of just focusing on such so-called hot spot nodes, our interest is in the discovery of regional information about hot spots, e.g., structures composed of an intersection and roads leading to that intersection. For this, we introduce a subgraph-based approach, with subgraphs being part of the underlying road network, to describe predicted hot spots in a more comprehensive and intuitive manner. We also introduce different types of hot spots based on the evolution of respective subgraph structures over time. For example, one can predict that a hot spot grows over time or that a hot spot moves in a particular direction. Respective results are obtained based on successive predictions of hot spots. We evaluate the efficiency and effectiveness of our approach to hot spot prediction using a real large-scale road network and simulated traffic scenarios. In summary, the paper makes the following contributions:

- An efficient approach to predict a vehicle's near time future positions based only on the very recent past positions.
- An online prediction approach to determine near future hot spot nodes that likely will experience heavy traffic, based on predicted future locations of vehicles.
- An aggregation model for hot spot nodes to determine regional structures in a road network and the evolution of such predicted hot spot structures over time.
- An extensive evaluation of our framework based on a real-world road network with more than 170,000 nodes and up to 10,000 vehicles in different traffic scenarios.

The rest of the paper is structured as follows. After a review of related work in the following section, in Section 3 we detail our online approach for predicting future positions of objects and how to derive hot spot nodes from such a prediction. In Section 4, we then discuss how regional patterns can be formed from hot spot nodes, and we introduce different types of hot spots based on their behavior over time. After the presentation of an evaluation of our approach using a real-world road network and simulated traffic data in Section 5, we conclude the paper with a summary and outlook in Section 6.

2 Related Work

Research on path prediction for moving objects has been conducted for some time already, e.g., by Pelanis et al. in [PvJ06] and Tao et al. in [TFPL04]. These approaches, however, do not consider an underlying road network that restricts the possible movement of objects. More recently, approaches for path prediction of moving objects in road networks have been proposed by, e.g.,Kim et al. in [KWK$^+$07] and Jeung et al. in [JYZJ10]. The work in [KWK$^+$07] assumes that the objects' destinations are known, which is in contrast to our approach. In addition, both approaches, and also many others that predict motion paths in road networks, utilize the complete historical trajectories of all moving objects to infer common movement or turning patterns. Our prediction instead uses a simple heuristic based on only a few of the most recent object positions to identify positions that moving objects will likely travel to in the near future, as existing path prediction approaches are prohibitively expensive and thus cannot be used in an online fashion in large-scale road networks with high numbers of moving objects.

As our focus is on the identification of *regions* with heavy traffic, our work also relates to the detection of moving clusters [KMB05], convoys [JYZ$^+$08], and flocks [GvK06]. All of these moving object patterns require at least a minimum number of moving objects to be within a certain distance from each other. We use the same requirement during the detection of hot spots. The moving object patterns listed above are usually identified while, or even after, they appeared, as no methods for their prediction have been proposed, yet. However, we are able to utilize methods for analyzing the evolution of convoys, as proposed by Aung and Tan in [AT10], in order to identify different types of hot spots, like growing or moving hot spots.

The detection of hot routes and hot spots in road networks has been addressed by Li et al. in [LHLG07] and Liu et al. in [LLN$^+$10] respectively. Both approaches provide an analysis of the current traffic situation, and thus do not perform a prediction but rather a detection of hot spots.

Most similar to our work is the approach by Kriegel et al. in [KRSZ08], where they introduce a method for traffic density prediction in road networks. The approach in [KRSZ08] uses a statistical traffic model to predict the traffic density of any edge in the road network for a given prediction time, which can be either a specific point in time in the future, or a time interval of certain length. Predictions done by our approach always refer to a time interval, not a specific future point in time. In addition, we predict traffic intensity on nodes instead of edges and use a different weighting function. Nevertheless, the traffic density in the road network predicted by the approach in [KRSZ08] is very similar to our concept of weights that are assigned to each node in the network to represent the predicted traffic intensity. Our approach goes one step further and considers regional aspects of hot spots by identifying hot spot regions in the form of concrete subgraphs structures in the road network. Overall, our approach is not tailored as much to predictions for longer periods of time as [KRSZ08] is, but instead focused on an efficient and scalable approach for predicting regional hot spots and their evolution in the near future.

3 Prediction of Traffic Intensity

The prediction of hot spots requires knowledge about locations that moving objects are likely to visit within some future time window. We need to infer this information from the trajectories observed thus far, as we assume that there is no information available about future routes or destinations of moving objects. We therefore now present a method that, at a given point in time, predicts the traffic intensity at all intersections in a road network for a time interval of a certain length. Based on this prediction, locations that are likely to be visited by a high number of moving objects within the given time interval can be identified and processed for further exploration.

In the following Section 3.1, we first explain our setting, i.e., the road network and moving objects. We then detail our approach for traffic intensity prediction in Section 3.2. Methods to infer hot spots from the traffic intensity prediction are discussed in Section 3.3.

3.1 Road Network and Moving Objects

Similar to [JYZJ10], we model a *road network* as an undirected graph $G = (V, E, D)$, where V is the set of vertices, i.e., connections between two or more road segments, E is the set of edges, i.e., road segments, and D is a function that maps each edge to a set of descriptive attributes, such as the speed limit of road segments. Note that road segments could also be modeled as directed edges that indicate, e.g., one-way roads or roads where the speed limit differs for the two directions. Our approach could be easily applied to a directed graph as well.

A set of objects O is moving within the road network. Each such *moving object* $o \in O$ has an associated unique identifier and an associated trajectory $T(o)$ containing its previously observed locations. We assume a synchronized model where moving objects report their location periodically at times $t_{-n}, \ldots, t_{-1}, t_0, t_1 \ldots$, with $n \in \mathbf{N}$, where t_0 denotes the current point in time, times with negative subscript are in the past, and times with positive subscript are in the future. We denote each such time t_i a *time step*. The duration between two consecutive time steps t_i and t_{i+1} is called the *step size*, which is application specific and can range from fractions of a second to minutes, or even hours, although the latter is uncommon in road network applications. When a moving object reports its location at time t_i, it is of the form of a *measurement* $l_i(o) = (\mathbf{x}_i(o), v_i(o), t_i)$, where $\mathbf{x}_i(o)$ is o's position and $v_i(o)$ is its velocity. An object's position is usually a 2-dimensional vector containing longitude and latitude information, and we assume it to be on an edge or vertex in the road network.

Overall, the trajectory of an object $o \in O$ is an ordered list of measurements, i.e., $T(o) =$

$\langle(\mathbf{x}_{-n}(o), v_{-n}(o), t_{-n}), \ldots (\mathbf{x}_{-1}(o), v_{-1}(o), t_{-1}), (\mathbf{x}_0(o), v_0(o), t_0)\rangle$, where t_{-n} is the oldest time step and thus represents the first measurement reported by object o. For our approach, it is not necessary to store the entire trajectory $T(o)$ of an object o. The two most recent measurements of each object are sufficient for the OPS approach. Note that a trajectory contains measurements up to the current time step t_0, but no information about the object's future positions. Predicting likely positions of an object within a certain time horizon in the future is part of our proposed method.

In a practical setting, the trajectories our approach uses for hot spot prediction represent a sample of all cars moving in the road network at any given time. As we will demonstrate in the evaluation in Section 5, a sample of all moving objects yields similar prediction results as the full set of moving objects would.

3.2 The OPS Approach

We now present *OPS*, an approach for Online Prediction of Hot Spots. Given trajectories $T(o)$ for all objects $o \in O$ at the current point in time t_0, our goal is to predict the nodes in the road network that these objects will likely visit within the next h time steps, i.e., in the time interval $[t_0, t_h]$. We predict these nodes separately for each object and denote the predicted set of nodes at time t_0 as $A_0(o)$. Note that each $A_0(o)$ is a subset of all vertices in the road network, thus $A_0(o) \subseteq V$, and $A_0(o)$ may be different for each object. In addition, a weight $w_0(v, o) \in [0, 1]$ needs to be determined for each node $v \in A_0(o)$ that indicates how likely it is for o to visit v. These weights are an essential aspect of the OPS approach, as nodes in $A_0(o)$ should not all contribute equally to the traffic intensity prediction, but based on how likely they actually are part of the future path of a moving object. By aggregating the weights $w_0(v, o)$ that all objects $o \in O$ contribute to a node v, the overall weight $w(v)$ for v is known, which is an estimator for its traffic intensity within the next h time steps.

A *hot spot* that is predicted at time t_0 is a part of the road network where a significant number of objects $o \in O$ are likely to be located in the time interval $[t_0, t_h]$. A *hot spot node* is therefore defined as a node $v \in V$ having a weight $w(v)$ that is relatively high with respect to the weights of most other nodes. We infer hot spot nodes based on the weights $w(v)$ as a last step in the presented approach (cf. Section 3.3).

In the next paragraphs, we will detail individual aspects of the algorithm, i.e., assigning and updating the weights $w(v)$ at each time step and extracting hot spot nodes based on the weights of all nodes. The complete OPS approach is given in Algorithm 1.

Weight assignment For each object o, a weight $w_0(v, o)$ that is greater than zero is only assigned to vertices that are likely to be visited by o within the next h time steps. In order to determine the subset of nodes that might be affected, i.e., the set $A_0(o) \subseteq V$, we first perform a coarse prediction of o's path during the next h time steps as follows (cf. lines 9–14 of Algorithm 1).

Let $\mathbf{d}_0(o) = \mathbf{x}_0(o) - \mathbf{x}_{-1}(o)$ be o's direction of movement at time t_0. Then, $\mathbf{x}_h(o) =$

```
Input: graph $G = (V, E, D)$, trajectories $T(o)$ for each $o \in O$, time horizon $h$
Output: set $hn(t_0)$ of all hot spot nodes predicted at the current time $t_0$
1  foreach $v \in V$ do // initialization
2      $w(v) = 0$;
3      $c(v) = \emptyset$;
4  foreach new time step do
5      foreach moving object $o \in O$ do
6          foreach $v \in A_{-1}(o)$ where $\mathbf{d}_0(o) \cdot (\mathbf{x}(v) - \mathbf{x}_0(o)) < 0$ do // remove weights
7              $w(v) -= c(v)[o]$;
8              remove $o$ from the set $c(v)$
           // compute new $A_0(o)$
9          $\mathbf{x}_h(o) = \mathbf{x}_0(o) + h \cdot v_0(o) \cdot (\mathbf{x}_0(o) - \mathbf{x}_{-1}(o))$;
10         $\mathbf{x}'_h(o)$ = node $v \in V$ closest to $\mathbf{x}_h(o)$;
11         $\mathbf{x}'_0(o)$ = node $v \in V$ closest to $\mathbf{x}_0(o)$ in direction of $o$'s movement;
12         $A_0(o)$ = all nodes visited by $A^*$ $(\mathbf{x}'_0(o), \mathbf{x}'_h(o))$ ;
13         foreach $v \in A_0(o)$ do // assign and update weights
14             $w_0(v, o) = 1 - (cost(\mathbf{x}'_0(o), v)/cost(\mathbf{x}'_0(o), \mathbf{x}'_h(o)))$;
15             $w(v) += w_0(v, o)$;
16             $c(v)[o] += w_0(v, o)$;
17     mean = mean of weights $w(v),\ \forall v \in V$;
18     std = standard deviation of weights $w(v),\ \forall v \in V$;
19     foreach $v \in V$ do
20         if $w(v) \geq$ mean $+ 3 *$ std then
21             add $v$ to the set $hn(t_0)$;
22     return $hn(t_0)$
```

Algorithm 1: OPS approach to determine hot spot nodes in the road network G

$\mathbf{x}_0(o) + h \cdot v_0(o) \cdot \mathbf{d}_0(o)$ is the point in space that object o travels to within the next h time steps, assuming o maintains its current direction of movement and velocity. As $\mathbf{x}_h(o)$ is not necessarily on the road network G, we map it to the closest vertex in G, denoted $\mathbf{x}'_h(o)$, by performing a nearest neighbor search. As we only consider vertices for our traffic intensity prediction, we also need to map o's current location to the closest node $\mathbf{x}'_0(o)$ in G, which we do by finding the vertex that is closest to $\mathbf{x}_0(o)$ towards $\mathbf{d}_0(o)$. Figure 1 depicts an example scenario. The reported locations of o at time steps t_{-1} and t_0 are marked by blue rectangles, as is the predicted location at time t_h. In addition, nodes $\mathbf{x}'_0(o)$ and $\mathbf{x}'_h(o)$, i.e., the vertices in the road network that $\mathbf{x}_0(o)$ and $\mathbf{x}_h(o)$ were mapped to, are marked by green triangles. Object o's direction of movement $\mathbf{d}_0(o)$ is depicted as a blue arrow. Note that there are other options to compute $\mathbf{d}_0(o)$, e.g., as the average direction of the k most recent time steps.

Given $\mathbf{x}'_0(o)$ and $\mathbf{x}'_h(o)$, i.e., the start and end nodes of o's movement during the next h time steps, we find the set $A_0(o)$ of likely visited vertices using the routing algorithm

A^* [HNR68]. A^* performs a best-first search to find the lowest-cost route from $\mathbf{x}'_0(o)$ to $\mathbf{x}'_h(o)$. To compute the cost $cost(v_1, v_2)$ for the route between two nodes v_1 and v_2, function D of the graph is used to determine the speed limit of each road segment. Thus, the lowest-cost route found by the A^* algorithm is the *fastest* route. A^* uses heuristics to cut off the search on paths that are not likely to contribute to the fastest route. Thus, all nodes visited by A^* are in the vicinity of the fastest route and thus likely to be visited by o during the time interval $[t_0, t_h]$. Hence, all nodes visited by A^* during the computation of the fastest route from $\mathbf{x}'_0(o)$ to $\mathbf{x}'_h(o)$ are in the set $A_0(o)$, and thus, $\forall v \in A_0(o)$: $w_0(v, o) > 0$. In Figure 1, these nodes are marked by filled black circles.

The value of each weight $w_0(v, o)$ at time t_0 should indicate how likely o is to visit v within the next h time steps. Nodes that can be reached from the known start node $\mathbf{x}'_0(o)$ within a short amount of time are more likely to be visited, because the object o may change direction or velocity at some point in time $t_i < t_h$ and thus our predictions for likely visited nodes become less accurate as the time traveled by o since t_0 increases. Nodes in $A_0(o)$ that can be reached faster by o are therefore assigned a higher weight than those where o takes longer to reach them. We compute the weight $w_0(v, o)$ of each node $v \in A_0(o)$ as follows:

$$w_0(v, o) = 1 - \frac{cost(\mathbf{x}'_0(o), v)}{cost(\mathbf{x}'_0(o), \mathbf{x}'_h(o))} \tag{1}$$

That is, the assigned weight $w_0(v, o)$ decreases as the A^*-based cost between the start node and v increases. Note that $\forall v \in A_0(o) \setminus \mathbf{x}'_h(o)$: $cost(\mathbf{x}'_0(o), v) < cost(\mathbf{x}'_0(o), \mathbf{x}'_h(o))$, as A^* performs a best-first search and thus no node in $A_0(o)$ can have higher cost with respect to start node $\mathbf{x}'_0(o)$ than node $\mathbf{x}'_h(o)$. Therefore, $w_0(v, o) \in [0, 1]$ always holds. In the example in Figure 1, nodes v_1 and v_2 are two of the nodes that are assigned a weight by object o at that time. The weight assigned to v_1 is higher than that for v_2, i.e., $w_0(v_1, o) > w_0(v_2, o)$.

Weight update Weight assignment is performed at every time step right after new measurements arrive from all moving objects. It is therefore necessary to update the weights of all nodes $v \in V$, specifically to aggregate weights $w_i(v, o)$ that are assigned at different time steps, and to rewoke weights that were contributed by objects that, at time t_0, are no longer likely to visit v (cf. lines 6–8 and 15–16 of Algorithm 1).

At every point in time, the weight $w(v)$ of a node v is the sum of all weights $w_i(v, o)$, i.e., the weights that all objects have contributed for node v. We use the set $c(v)$ to keep track of all objects that contributed to $w(v)$. If an object o repeatedly contributes a weight greater than zero to node v, i.e., $v \in A_i(o)$ holds, in consecutive time steps t_i, these weights are added up instead of just counting the most recent weight that o contributed to v. This is because summing up the assigned weights over several consecutive time steps provides for a traffic intensity prediction that spans more than one time step, and thus helps to identify "popular" nodes. This aspect of the OPS approach is in contrast to most existing approaches, which mostly operate on snapshots and do not take the previously discovered popularity of locations into account. That is, other approaches start over every

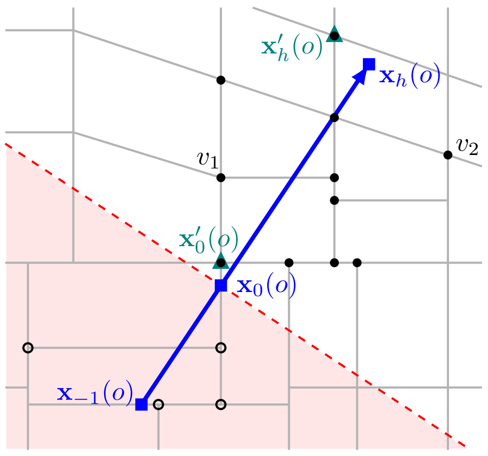


Figure 1: Example scenario depicting weight assignment, update, and revocation in OPS

time a prediction is done and thus disregard previously obtained information about the popularity of locations, whereas OPS maintains this information and uses it to reinforce the expressiveness of the prediction. This is because if an object o repeatedly assigns weights greater than zero to a node v in consecutive time steps, it indicates that it becomes more and more likely for o to visit v.

However, adding up all weights continuously does not yield an accurate prediction for the *future* traffic intensity, as the sum of all weights contributing to a node's overall weight $w(v)$ simply represents v's all-time likelihood to be visited by moving objects. We therefore revoke weights contributed by certain objects, to account for the event that an object o is not likely any more to visit v. Specifically, at time t_0, all weights $w_i(v, o)$ that were contributed by nodes $v \notin A_0(o)$ at time steps $t_i < t_0$ should be subtracted from $w(v)$. Note that we store the sum of all weights contributed to a node v by object o in $c(v)[o]$. Consider an example where an object o contributed the following weights to node v: $w_{-3}(v, o) = 0$, $w_{-2}(v, o) = .35$, $w_{-1}(v, o) = 0.2$, and $w_0(v, o) = 0$. Thus, at t_0, a total weight of $.35 + 0.2 = 0.55$ should be subtracted from $w(v)$.

Revoking weights in this fashion requires to determine for each object o all nodes v for which $v \notin A_0(o)$ and $v \in A_{-1}(o)$ hold, i.e., nodes that are not an element of $A_0(o)$ at time t_0, but have been an element of $A_{-1}(o)$ at time t_{-1}. In order to do this, one would have to store $A_{-1}(o)$ and compute the set difference $A_{-1}(o) \setminus A_0(o)$ for all objects o, which, for large graphs, requires a lot of additional memory and is computationally expensive. Therefore, we instead use an efficient heuristic to determine nodes where weights need to be revoked. The basic idea is depicted in Figure 1. The dashed red line is perpendicular to object o's direction of movement $\mathbf{d}_0(o)$ and crosses at o's current location $\mathbf{x}_0(o)$. All nodes v located in the shaded area below the dashed line have already been "passed" by object o and are thus not likely to be visited by o in the (near) future. Specifically, object o either did visit the nodes in the shaded area already or it could have visited them using a shorter route before t_0, as visiting any of these nodes now, i.e., after time t_0, would require

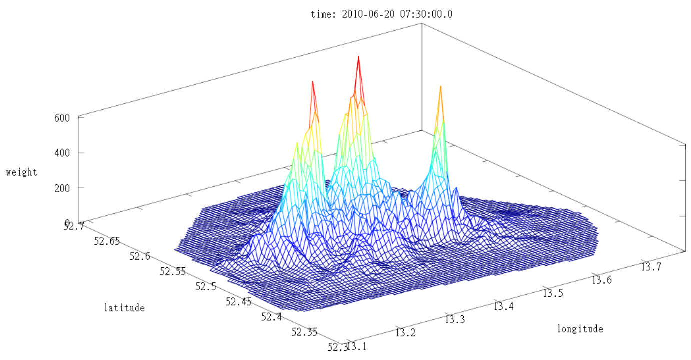


Figure 2: Traffic intensity prediction for an example data set at time $t_0 = $ 7:30am on June 20th, 2010, using $h = 120$

o to take a detour, which is much less likely.

For nodes v that are located in the shaded area, the following inequality holds:

$$\frac{\mathbf{d}_0(o) \cdot (\mathbf{x}(v) - \mathbf{x}_0(o))}{|\mathbf{d}_0(o)| \times |\mathbf{x}(v) - \mathbf{x}_0(o)|} < 0 \quad \Leftrightarrow \quad \mathbf{d}_0(o) \cdot (\mathbf{x}(v) - \mathbf{x}_0(o)) < 0 \tag{2}$$

Thus, at time t_0, we revoke weights from all nodes $v \in A_{-1}(o)$ for which inequality 2 holds. Such nodes are marked by empty circles in the example in Figure 1. Note that $\mathbf{x}(v)$ in the above inequality denotes the position of vertex v. Checking if weights need to be revoked can be done at the beginning of each time step, i.e., before the new set $A_0(o)$ is computed for all moving objects. Then, this step does not require any additional memory, as $A_{-1}(o)$ is still present at the beginning of a time step, and can be overwritten by $A_0(o)$ once all nodes $v \in A_{-1}(o)$ have been checked according to inequality 2.

3.3 Identification of Hot Spot Nodes

After updating the weights $w(v)$ of all nodes $v \in V$ in the road network at time t_0, one needs to analyze the distribution of weights over the road network in order to identify hot spot nodes. Figure 2 depicts the weights assigned to all nodes in a given road network at a specific point in time $t_0 = $ 7:30am based on an example data set of trajectory data. As is obvious from the figure, it is easy to identify areas in the road network where an exceptionally high traffic intensity is predicted for time interval $[t_0, t_h]$ ($h = 120$ in the figure). Vertices having a high weight stand out as peaks, and thus one can identify a total of three hot spots in the example in Figure 2.

While the visual identification of hot spots is straightforward, a more formal definition of hot spots is needed in order to automatically extract them from the set of all nodes in G. A hot spot is comprised of a single vertex, i.e., it is a hot spot node, or a set of connected vertices in the road network. The latter group of hot spots is discussed in Section 4, as several different types of such hot spots can be distinguished. In the following, we show how to identify hot spot nodes based on all weights $w(v)$.

Definition 3.1 ***(Hot Spot Nodes)*** *A* ***hot spot node*** *predicted at time t_0 is a vertex $v \in V$ that fulfills the following two properties:*

1. *The weight $w(v)$ of node v is significantly higher than the weight of most other vertices in the road network, based on some measure.*

2. *At least $minMO$ distinct moving objects $o \in O$ contributed to v's weight $w(v)$, i.e., $|c(v)| \geq minMO$ holds.*

We refer to the set of all hot spot nodes predicted at time t_0 as $hn(t_0)$.

The second property above ensures that a hot spot node is indeed likely to be visited by a sufficiently large number of moving objects. This is very similar to the detection of moving clusters, where a minimum number of objects is required in order to constitute a cluster. Regarding property (1), different methods can be applied to identify high weights, such as a threshold δ, which may be fixed or adjustable based on network density and number of present moving objects, or outlier detection approaches, which are perfectly suited to detect weights that are significantly higher than those of most other vertices.

In the OPS approach, we use an efficient outlier detection method from [FPP97] that labels all nodes v as outliers having a weight $w(v)$ that is more than three standard deviations above the mean weight of all nodes $v \in V$ (cf. lines 17–22 of Algorithm 1). All such nodes are added to the set $hn(t_0)$ of hot spot nodes at time t_0. Of course, depending on the desired application and the computational resources available, other outlier detection approaches may be used. However, the basic method used in the OPS approach proved to be sufficient, as we demonstrate in the evaluation in Section 5.

The set $hn(t_0)$ of hot spot nodes is updated at every time step to reflect the traffic intensity predicted in the road network at time t_0. That is, each node $v \in hn(t_0)$ is predicted to experience an amount of traffic in the time interval $[t_0, t_h]$ that is significantly larger than the amount of traffic that will be observed at most other nodes in the road network, and thus we expect the number of moving objects visiting v to be very high compared to the number of objects visiting other nodes in the road network.

4 Regional Hot Spots and Hot Spot Patterns

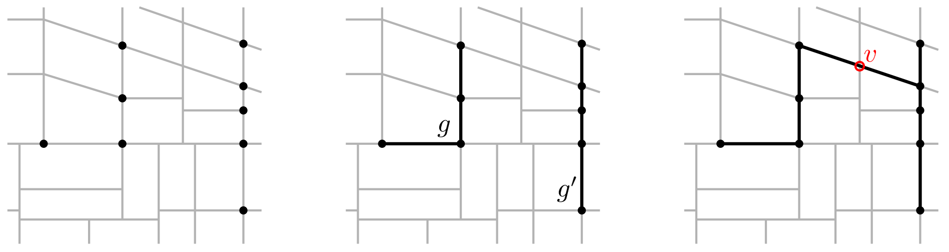
Although hot spot nodes already provide some interesting information about parts of the road network, e.g., intersections that likely will experience high traffic in the near future, there can be many such nodes at any given time. Relationships among hot spot nodes and in particular their aggregated future evolution is not that obvious. To address these issues, in the following section, we present a simple clustering approach for hot spot nodes to determine more meaningful subgraph structures that convey regional information. Using such subgraphs, in Section 4.2, we then introduce different types of predicted hot spot patterns in a road network in terms of their evolution over time.

4.1 Regional Hot Spots

Given a set of hot spot nodes $hn(t)$ determined at time t, which typically corresponds to the current time t_0. To provide the user an aggregated view on these nodes and to better determine structures and their evolution in future predictions, a set of *subgraphs*, denoted $hg(t)$ is derived from $hn(t)$ as follows:

- each node $v \in hn(t)$ belongs to exactly one subgraph $g \in hg(t)$, and
- if two nodes $v, v' \in hn(t)$ are connected in the road network, i.e., $(v, v') \in E$, then v and v' belong to the same subgraph $g \in hg(t)$.

Thus, $hg(t)$ is the set of connected subgraphs based on hot spot nodes $hn(t)$, and the subgraphs are naturally embedded in the road network (see Figure 3 for an example). Subgraphs formed in this way naturally convey regional aspects of predicted hot spots. Such a regional view, as part of the underlying road network, can be extended further by allowing the formation of larger subgraph structures and thus regions. For example, two subgraphs $g, g' \in hg(t)$ can be combined into a single subgraph if there exists a node $v \notin hn(t)$ and g, g' are connected via v, as illustrated in Figure 3(c).



(a) Set of predicted hot spot nodes

(b) Subgraphs g and g' derived from hot spot nodes

(c) Two subgraphs connected via a non-hot spot node v

Figure 3: Subgraphs (b. and c.) derived from hot spot nodes (a.) in a road network

Recall that with each node in a subgraph $g \in hg(t)$ a weight above a certain threshold is associated (cf. Section 3.3). Using an appropriate visualization, e.g., such as the one shown in Figure 2, interesting patterns can emerge from subgraph structures. For example, a subgraph may represent an intersection (with a predicted high traffic) with road segments leading to that intersection node and respective nodes of the road segments having a decreasing weight the farther away they are from the intersection. The question that remains, however, is how such patterns predicted at a point in time t now evolve over time, i.e., in subsequent predictions. This is an important aspect supported by our framework and is discussed in the following section.

4.2 Hot Spot Pattern

A set of subgraphs representing regional aspects of predicted hot spots is determined at each time step. Because of the continuous movement of objects in the road network, new subgraphs can emerge, existing ones remain for a period of time or they evolve in terms of expansion, shrinking, or movement. For example, a subgraph representing roads (or road segments) leading to an intersection may grow over time, reaching some kind of peak, and then shrinks until it disappears in the prediction of future hot spots. This is a typical example of an evolving pattern that emerges, e.g., during rush hour.

In general, for decision purposes in traffic planing and management, it is important to study the evolution of predicted regional hot spots over time, an important aspect not covered in related work.

In the following, we introduce a framework to determine important hot spot patterns that occur over time. For this, we assume a sequence of sets of subgraphs $L = [hg(t_{-k}), hg(t_{-k+1}), \dots, hg(t_0)], k \geq 2$, which has been determined up to the current time t_0. We thus investigate hot spot patterns that occur in a time interval of length $k+1$. Clearly, the larger the interval, the more (stable) patterns can be determined. Typically, the size of the interval is chosen as a multiple of the step size in the trajectory data.

The most simple type of a hot spot pattern is as follows:

Definition 4.1 ***(Strict Stationary Hot Spot)*** *Let g be a subgraph of some set of subgraphs $hg(t) \in L$. g is a* strict stationary hot spot *if for all $hg(t') \in L$, $g \in hg(t')$.*

In other words, the exact same subgraph has to appear at each point in time in the given interval. This property is independent of possible variances of the weight of the nodes in g for this time interval. An example of such a pattern is a shopping center or large parking lot where vehicles stay for some time. Such type of patterns might be rare and perhaps less interesting in a real world setting, as hot spots and subgraph structures, respectively, are more likely to expand or shrink over time.

Definition 4.2 ***(Growing stationary hot spot)*** *Let g be a subgraph of $hg(t_{-k})$. g is said to be a* growing stationary hot spot *if*

1. *there exists a subgraph g_i in each $hg(t_i)$, $-k \leq i \leq 0$, s.t. g is a subgraph of g_i, and*
2. *for each two such consecutive subgraphs g_i, g_{i+1} from $hg(t_i)$ and $hg(t_{i+1})$, respectively, we have that g_i is a subgraph of g_{i+1}.*

The concept of a shrinking stationary hot spot can be formulated in an analogous way, here with g being a shrunken subgraph observed at time t_0. These two types of hot spot patterns can also be combined to form a new pattern, first growing, then shrinking, in the given interval. Such a pattern would then formalize the scenario we gave above describing the evolution of traffic centered around an intersection.

The final structural pattern we introduce here are *moving hot spots.* Their concept is motivated by the following scenario. Assume a truck on a highway that cannot be passed by other cars. Such a scenario relates to the leadership pattern described by Andersson et al. in [AGLW08]. Over time, more and more cars will drive behind that truck that is driving towards its destination, thus forming a hot spot that moves within the road network. A precise formal definition of such a pattern, which we are only able to discover in a very restricted setting in our system, is a little bit more involved as several parameters can be introduced to tailor the properties of such a pattern. But, in general, a subgraph $g \in hg(t_{-k})$ representing such a hot spot is moving if at least l nodes of g can be found in a subsequent subgraph $g' \in hg(t_{-k+1})$, and again for this subgraph g' at least l nodes can be found in a subsequent subgraph $g'' \in hg(t_{-k+2})$, and so on. Parameter to this pattern then is the number of nodes that must be shared among subgraphs in consecutive timestamps.

The patterns of predicted hot spots described thus far give a fairly comprehensive regional view of patterns in terms of parts of a given road network but they only consider structural properties for a given (sliding) time interval. Recall that at each time step, in addition to the weight of a node, we also maintain the set of objects that contributed to that weight (cf. Section 3.2). This information can be explored as well in the context of the above pattern. Here we only outline the basic idea. An example can be best illustrated in the context of a moving hot spot. The leadership-like pattern mentioned above has the property that mostly the same cars contribute to the weight of the nodes forming the pattern, as cars cannot pass the truck. An example where the moving objects contributing to the hot spot nodes of a pattern is not that homogeneous would be the case for the intersection. As cars pass the intersection and new cars approach the intersection, the set of moving objects contributing to respective hot spot nodes would not be as homogeneous as in the leadership pattern.

We finally want to address the computational aspects regarding the discovery of predicted hot spot patterns. Key to the complexity are (1) the size $k + 1$ of the interval (number of time steps, respectively) in which hot spot patterns are to be discovered and (2) the number of subgraphs that have been determined for each element in the sequence $L = [hg(t_{-k}), hg(t_{-k+1}), \ldots, hg(t_0)]$. Let L' denote the set of subgraphs determined at time point t_1, which comprises $[hg(t_{-n+1}), \ldots, hg(t_0), hg(t_1)]$. If there is a subgraph g corresponding to a stationary, growing, or moving hot spot in L, then it only has to be checked whether there is a subgraph g' in $hg(t_1)$ that is a respective continuation of g. In contrast to this simple case, the more complex case for which all subgraphs in L' have to be processed is when there now is a subgraph in $hg(t_1)$ such that a new stationary, growing, or moving hot spot can be formed based on the subgraphs in L'. As we will show in the following section for different traffic scenarios, the total number of subgraph structures determined at each point in time is quite moderate, compared to the total number of hot spot nodes. Thus, processing respective subgraph structures for an interval of length $k+1$ only requires minimal overhead. A key requirement, of course, is that based on the step size vehicle positions are obtained, the computation of hot spot nodes and structures at each time step has to be done within a two steps sizes. That this requirement is satisfied will be demonstrated in the following section.

5 Experimental Evaluation

We now present some experimental results evaluating the efficiency of the OPS approach as well as properties of the predicted hot spot nodes and subgraph structures. As we use synthetic data in the evaluation, we first provide details about the generated test data in Section 5.1. We then present evaluation results regarding efficiency and the predicted hot spots in Sections 5.2 and 5.3 respectively.

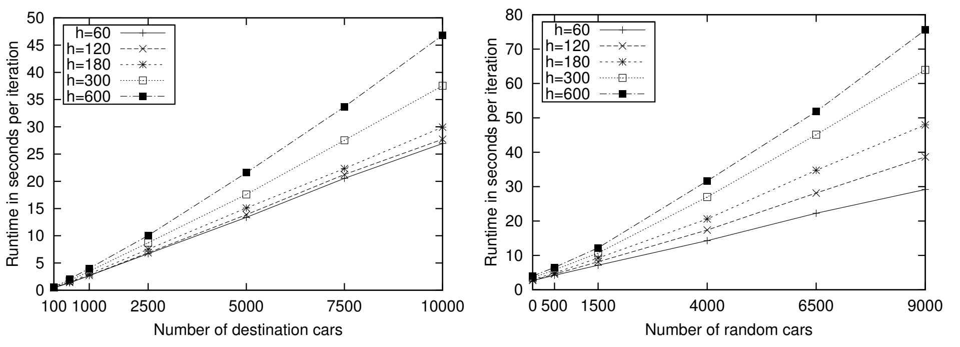
5.1 Test Data and Experimental Setup

We use synthetic trajectory data to evaluate the OPS approach, as this enables us to generate data in different scenarios. Trajectories were created using our moving object simulator called *Tragor* (Trajectory generator on road networks), which simulates the movement of cars on a world-wide road network obtained from OpenStreetMap [OSM]. In Tragor, different types of cars exist. Two of them are used in our experiments: *destination cars* drive along the fastest route from a pre-defined start position to a pre-defined destination. In contrast, *random cars* follow a random path by starting at a random position in the road network and taking random turns at each intersection. Cars in Tragor always obey the speed limit, which is provided as a property of each road segment in the OpenStreetMap data set. We chose to use Tragor because it enables us to include different kinds of hot spot patterns in our test data sets.

For each of our experiments, we generate sets of trajectories that contain the movement of n_d destination cars and n_r random cars. Thus, each set has a total of $n = n_r + n_d$ moving object trajectories. We create hot spots by assigning each destination car one of e distinct destinations, $e \ll n_d$, such that the same number of destination cars is assigned to each of the e destinations. Random cars are added as noise, since they will not contribute to any hot spots due to their random movements.

For the simulation, we do not use the entire road network, but choose two different regions of interest. The first region is the city of Berlin, Germany, which represents a dense road network, and the second one is the German State of Brandenburg, which is much more sparse than the Berlin road network. Thus, when generating trajectory data sets, the moving objects are either moving on the Berlin road network, denoted *BLN*, or on the Brandenburg road network, denoted *BB*. The BLN (BB) road network contains about $180,000$ ($408,000$) nodes and covers roughly 340 ($11,380$) square miles. We will indicate the road network used in the description of the experiments below.

All experiments were conducted on a 16 core Intel(R) Xeon(R)E5520 @ 2.27GHz with 48GB of main memory running Debian Linux. The OPS approach was implemented in



(a) Runtime in terms of number of destination cars (b) Runtime in terms of random cars using $n_d = 1000$

Figure 4: Runtime per iteration of OPS approach in BLN road network

Java.

5.2 Efficiency of OPS

In a first set of experiments, we evaluated the runtime of the OPS algorithm using 14 different sets of trajectories, seven each for the BLN and the BB road network. The test data sets contain the trajectories of between $n = 100$ and $n = 10,000$ moving objects, all of them destination cars that drive to one of $e = 10$ pre-defined destinations. Trajectories were generated for a period of one hour using a step size of 60 seconds, resulting in a total of 60 measurements for each moving object. Hot spot prediction was done for five different time horizons between 1 and 10 minutes. The results of these experiments are shown in Figure 4(a) for the BLN road network.

Runtimes depicted in Figure 4(a) are the total time for processing all new measurements in one time step, i.e., the runtime required for one iteration of the OPS approach. As can be seen in the figure, the runtime of OPS grows linearly with the number of moving objects, and is only slightly affected by the length of the prediction interval, i.e., the time horizon h. Specifically, for $h = 60$ the runtime is 27 seconds, while the runtime for a ten-times larger time horizon of $h = 600$ seconds has not even doubled, being less than 47 seconds. This demonstrates that our approach is very scalable both in terms of the number of moving objects and the length h of the prediction interval.

The efficiency of our approach is mostly due to the use of the A^* algorithm for predicting likely visited nodes and our heuristic for revoking weights. Only nodes that are relevant for the hot spot prediction are considered during weight assignment and update, and thus the total number of weights maintained in the graph over time is small, yielding a low runtime. To evaluate this aspect further, we performed a similar set of experiments with test data sets where we added random cars for noise. Specifically, we used $n_d = 1000$ destination cars and added between $n_r = 500$ and 9000 random cars, resulting in a total number of moving objects of up to $10,000$, i.e., $n = n_d + n_r = 1000 + 9000 = 10,000$.

The results of this set of experiments are shown in Figure 4(b).

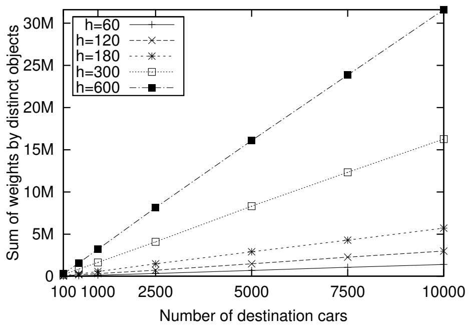
The increase of the runtime is similar to the one observed in the experiments using just destination cars. However, two differences can be observed: (1) the individual runtimes are higher than those in the respective previous experiments, and (2) the increase in runtime becomes steeper as the ratio of random cars to destination cars increases. Both observations are caused by the random movement of the majority of the cars, i.e., all random cars. Due to their random movement, each car o has very different sets $A_i(o)$ of likely visited nodes at each time step t_i, resulting in more computational effort to maintain the weights assigned to all nodes. This is because for a random car, a higher number of distinct nodes get assigned weights, whereas destination cars tend to assign weights to a similar set of nodes at each time step, resulting in a lower number of weights to be maintained overall. We will demonstrate this fact in further experiments below. Regarding the second observation above, the additional computational effort due to the random cars becomes more predominant when the ratio of random cars and destination cars increases, as is the case in our experiments, where $n_d = 1000$ is fixed and the number n_r of random cars increases.

Both observations above support our claim that OPS is efficient because of its ability to focus on nodes that are relevant to the hot spot prediction. Random cars interfere with this approach by adding irrelevant weights and nodes, thus reducing the efficiency of OPS. Runtimes for real-world trajectory data will usually be more similar to the ones shown in Figure 4(a), as real-world cars rarely exhibit random movement patterns, but instead drive fairly targeted to their destination.

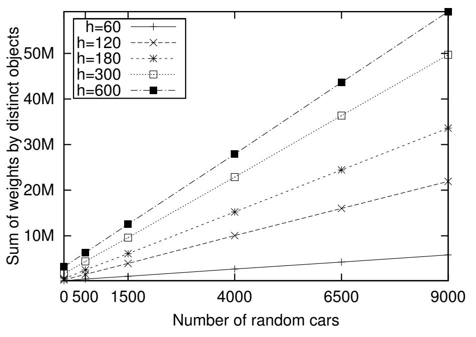
We performed the same experiments as above, i.e., using just destination cars and using an increasing number of random cars, for trajectories on the BB road network, and observed runtimes that were 10 to 15 times faster than those for the BLN road network. This is because the BB road network is much more sparse, resulting in a much lower number of nodes being visited by the A^* algorithm, and thus less computational effort overall.

To further explore the efficiency of OPS in terms of the number of weights that are assigned to nodes, we collected additional measurements during the experiments described above. Each node v in the road network keeps track of the moving objects that contributed to v's weight $w(v)$. By summing up, over all nodes, the number of objects that contributed to the node's weight, we gain insights in how the number of destination cars and random cars affects the distribution of weights in the road network. For this, we determined at the end of each iteration of OPS the sum of how many distinct moving objects contributed to the weight of each node. That is, we summed up the number of objects stored in the set $c(v)$, i.e., $|c(v)|$, over all $v \in V$. We denote this sum as $\sum(|c(v)|)$. In the following Figure 5, the maximum value of $\sum(|c(v)|)$ out of all 60 iterations is given.

In Figure 5(a) it can be seen that for higher numbers n_d of destinations cars and longer prediction horizons h the sum of assigned weights increases. This is expected, as for higher values of h a car is able to visit more nodes during the prediction time interval. Additionally, we can observe in Figure 5(b) that for high numbers of random cars, the number of assigned weights is 10 to 20 times higher than in a scenario with an equal number of destination cars. This again supports our claim that the weight assignment performed by destination cars, which behave very similar to cars in a real-world setting, is

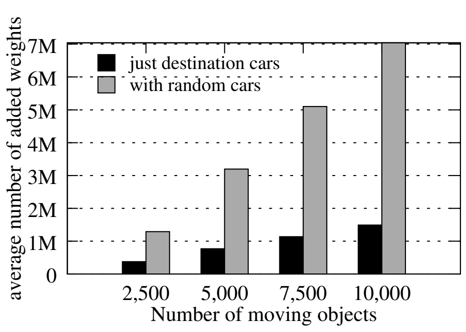


(a) $\sum(|c(v)|)$ for varying number of destination cars and no random cars

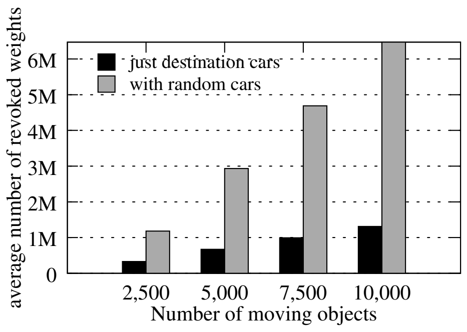


(b) $\sum(|c(v)|)$ for $n_d = 1000$ destination cars and a varying number of random cars

Figure 5: The number of weights assigned to nodes by distinct objects was determined for different numbers of moving objects n_d and n_r



(a) Average number of cars that newly contributed to a node's weight in each iteration



(b) Average number of moving objects and nodes for which weights where revoked in each iteration

Figure 6: Average number of weights that where added or revoked in each iteration of OPS

much more focused, yielding expressive weights at each node in the road network. Also, the much higher number of assigned weights in the presence of random cars justifies the increased runtime observed in the second set of experiments, whose results were depicted in Figure 4(b).

We also kept track of how many objects newly contributed to a node's weight in each iteration and for how many objects weights were revoked from nodes. We again summed up these values over all nodes. The results are depicted in Figure 6, and show the average over all 60 iterations. These results are in line with all previous experiments, confirming that random cars indeed exhibit a significant fluctuation in terms of the distinct nodes to which they assign weights. Both Figures 6(a) and 6(b) show that in the scenarios with a significant number of random cars, a much higher number of weights gets added and revoked in each iteration, compared to scenarios where only destination cars move in the road network.

	t_{10}		t_{20}		t_{30}		t_{40}	
n	$\lvert hn \rvert$	$\lvert hg \rvert$	$\lvert hn \rvert$	$\lvert hg \rvert$	$\lvert hn \rvert$	$\lvert hg \rvert$	$\lvert hn \rvert$	$\lvert hg \rvert$
1000	319	51	383	52	526	56	638	50
2000	328	50	478	52	571	61	660	54
3000	358	46	502	48	625	52	683	49

Table 1: Number of predicted hot spot nodes and subgraphs using $\varepsilon = 1$ and minPts $= 3$

5.3 Evaluating Predicted Hot Spots

Next, we performed a set of experiments to evaluate the predicted hot spot nodes and subgraphs. For this, we used test data sets containing $n = n_d =$ 1000, 2000, or 3000 destination cars that are driving to $e = 10$ distinct destinations in the BLN road network. It is important to note that we derived the data sets with fewer objects from the data set containing all 3000 moving objects, and thus, the hot spots should be similar throughout all three data set. We again used a step size of 60 seconds and simulated traffic for one hour, resulting in 60 measurements in each trajectory. All cars start at different positions, but as each 10% of the present cars share a destination, we expect them to "meet" at major roads and follow the same few access roads to any of the given destinations. In our experiments, we determined hot spot nodes after 10, 20, 30, and 40 time steps using a time horizon of $h = 120$ seconds. From the set of hot spot nodes, we derived according subgraphs as detailed in Section 4.1 and depicted in Figure 3(c). That is, we consider two hot spot nodes to belong to the same subgraph if they are connected directly or via at most one non-hot spot node.

We used the well-known clustering algorithm DBSCAN [EKSX96] to derive the subgraphs. The distance measure we used is the number of "hops" in the road network, i.e., the number of edges one has to use in order to get from one node to another. Thus, using $\varepsilon = 1$ in DBSCAN, we achieve exactly the kind of subgraphs we are looking for. We also made use of DBSCAN's minPts parameter, which controls the minimum number of nodes that have to be contained in a subgraph in order for this subgraph to be considered a hot spot region. We did this because heavy traffic usually spans more than one node, i.e., more than one intersection in the road network. Thus, we used minPts $= 3$ and minPts $= 5$ in our experiments. The results of these experiments are listed in Tables 1 and 2.

As is obvious from the figures, the number of predicted hot spot nodes and subgraphs does not increase with increasing number of cars. This demonstrates that hot spots are reliably predicted even for a sample of all moving objects, i.e., based just on the trajectories of 1000 or 2000 cars instead of all 3000. Our results also show that a reasonably small number of subgraphs is derived from a fairly large number of predicted hot spot nodes. The figures in Table 1, i.e., using minPts $= 3$, show that about 50 subgraphs are detected at any time. These subgraphs represent groups of cars that are driving towards the same destination. The individual groups either approach the destination from different directions, or they are at different distances to the destination. That is, one groups is further away from the destination than another, and thus they form separate hot spots.

	t_{10}		t_{20}		t_{30}		t_{40}	
n	$\|hn\|$	$\|hg\|$	$\|hn\|$	$\|hg\|$	$\|hn\|$	$\|hg\|$	$\|hn\|$	$\|hg\|$
1000	319	5	383	6	526	4	638	5
2000	328	3	478	4	571	3	660	4
3000	358	4	502	4	625	2	683	4

Table 2: Number of predicted hot spot nodes and subgraphs using $\varepsilon = 1$ and minPts $= 5$

6 Conclusion and Future Work

In this paper we presented an approach for online prediction of hot spots in road networks. As a first step, our OPS approach determines individual hot spot nodes by using an efficient heuristic to predict the traffic intensity at all nodes in the road network. For this, likely visited nodes are predicted for each moving object based on its two most recent positions. Weights are assigned to each of the likely visited nodes such that the sum of all weights for one node represents the expected traffic intensity at that node during a time interval of length h starting at the time of prediction. Based on the hot spots nodes that are predicted by OPS, we then inferred subgraphs that represent hot spot regions within the road network and identified different types of hot spot patterns. In our extensive evaluation using a real large-scale road network, we demonstrated the efficiency and effectiveness of our approach, especially in terms of the number of considered nodes. We also presented an evaluation of identified hot spot nodes and the according derived subgraphs.

As part of our ongoing work, we try to adjust the OPS approach to accommodate objects with different reporting rates. In addition, we experiment with other path prediction approaches to determine if they provide better estimates for the likelihood that a moving object will visit a certain node in the future. Such an adjustment to our approach may further improve the prediction of location, extend, and intensity of hot spots. We also currently work on predicting the life-span of hot spots as well as their evolution from one type to another. This comprehensive prediction of hot spots in road networks may benefit motorists as well as practitioners in application domains like logistics and urban planning.

References

[AGLW08] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting Leaders and Followers among Trajectories of Moving Point Objects. *GeoInformatica*, 12(4), 2008.

[AT10] Htoo Htet Aung and Kian-Lee Tan. Discovery of Evolving Convoys. In *Proceedings of the International Conference on Scientific and Statistical Database Management*, pages 196–213, 2010.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[FPP97] David Freedman, Robert Pisani, and Roger Purves. *Statistics*. W. W. Norton & Company, 1997.

[GvK06] Joachim Gudmundsson and Marc van Kreveld. Computing Longest Duration Flocks in Trajectory Data. In *Proceedings of the ACM international symposium on Advances in geographic information systems*, pages 35 – 42, 2006.

[HLT10] Jiawei Han, Zhenhui Li, and Lu An Tang. Mining Moving Object, Trajectory and Traffic Data (Tutorial at the 15th International Conference Database Systems for Advanced Applications, DASFAA), 2010.

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[JYZ$^+$08] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of Convoys in Trajectory Databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[JYZJ10] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Statistics-Based Path Prediction and Predictive Range Querying in Spatial Network Databases. *The VLDB Journal*, 2010.

[KMB05] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, volume 3633 of *Lecture Notes in Computer Science*, pages 364–381, 2005.

[KRSZ08] Hans-Peter Kriegel, Matthias Renz, Matthias Schubert, and Andreas Züfle. Statistical Density Prediction in Traffic Networks. In *Proceedings of the SIAM International Conference on Data Mining*, pages 692–703, 2008.

[KWK$^+$07] Sang-Wook Kim, Jung-Im Won, Jong-Dae Kim, Miyoung Shin, Junghoon Lee, and Hanil Kim. Path prediction of moving objects on road networks through analyzing past trajectories. In *KES*, pages 379–389, Berlin, Heidelberg, 2007. Springer-Verlag.

[LHLG07] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. Traffic Density-Based Discovery of Hot Routes in Road Networks. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, volume 4605 of *Lecture Notes in Computer Science*, pages 441–459, 2007.

[LLN$^+$10] Siyuan Liu, Yunhuai Liu, Lionel M. Ni, Jianping Fan, and Minglu Li. Towards Mobility-based Clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 919–928, New York, NY, USA, 2010. ACM.

[OSM] OpenStreetMap. `http://planet.openstreetmap.org`.

[PvJ06] Mindaugas Pelanis, Simonas Šaltenis, and Christian S. Jensen. Indexing the past, present, and anticipated future positions of moving objects. *ACM Trans. Database Syst.*, 31(1):255–298, 2006.

[TFPL04] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.