Statistical Density Prediction in Traffic Networks

Hans-Peter Kriegel Matthias Renz Matthias Schubert Andreas Zuefle Ludwig-Maximilians University Munich, Germany http://www.dbs.ifi.lmu.de {kriegel,renz,schubert,zuefle}@dbs.ifi.lmu.de

Abstract

Recently, modern tracking methods started to allow capturing the position of massive numbers of moving objects. Given this information, it is possible to analyze and predict the traffic density in a network which offers valuable information for traffic control, congestion prediction and prevention. In this paper, we propose a novel statistical approach to predict the density on any edge in such a network at a future point of time. Our method is based on short-time observations of the traffic history. Therefore, it is not required to know the destination of each object. Instead, we assume that each object acts rationally and chooses the shortest path from its starting point to its destination. This assumption is employed in a statistical approach describing the likelihood of any given object to be located at some position at a particular point of time. Furthermore, we propose an efficient method to speed up the prediction which is based on a suffix-tree. In our experiments, we show the capability of our approach to make useful predictions about the traffic density and illustrate the efficiency of our new algorithm when calculating these predictions.

1 Introduction

Traffic control systems for large traffic networks have attracted much attention, recently. One large challenge of traffic controlling is traffic prediction, i.e. methods that are able to estimate the traffic density for a future point of time. The information provided by a traffic prediction system allows to initiate various types of traffic control methods in order to avoid congestion situations within a monitored network.

One of the most important application areas of traffic control systems are road networks. In particular at rush-hour when the risk of the occurrence of traffic jams is very high traffic control systems will become be very important. The combination of modern positioning and mobile communication systems enables us to capture real-time positions of mobile clients on a road network at a central server. These systems can be used to con-



Figure 1: Architecture of the Traffic Capture System (TCS) and the Traffic Analysis System (TAS).

tinuously track the current traffic at arbitrary locations in a traffic network. An example of a possible architecture of a traffic analysis system is illustrated in Figure 1. This architecture consists of two modules, the Traffic Capture System (TCS) and the Traffic Analysis System (TAS).

The TCS captures the position of the mobile objects traveling in the network. Each mobile object is a client equipped with a GPS-System that can capture its actual position in space and a transmitter that can determine its actual position to the next receiving antenna. The receiving antennas forward the incoming information to a central server which is a component of the Traffic Analysis System (TAS). A possible infrastructure for the communication between the mobile clients and the central server might rely on ordinary cell-phones. This method is especially interesting because modern navigation systems are already able to communicate via cell-phones.

Based on the information provided by the TCS, the central server of the TAS can continually track the positions of the captured moving objects within the road network. This information is now used to estimate future traffic situations. Thereby, further information like common driving behavior and the assumption that each mobile object moves on an intuitive path from its

starting point to its destination can be incorporated in order to improve the accuracy of traffic prediction. One possible output of a traffic analysis system is depicted in Figure 1. The output of the traffic processor in our example shows the traffic network with marked zones that indicate the location for which a large traffic volume is expected within the next half an hour. Such kind of information is reported back to the mobile clients that can subsequently use this information to plan an alternative route in order to avoid the predicted traffic jams. Furthermore, a traffic control systems can dynamically adjust the speed limits on highways in order to counter the emerging congestion.

In this paper, we propose a novel statistical approach to predict the density of any edge in a road network at some future point of time. Our method is based on short-time observations of the traffic history, i.e. the input for the traffic predictor are recent trajectories of the moving objects. The destinations and the following trajectories of the moving objects are unknown. Therefore, we need to estimate the future motion of the objects in the network. We assume that the moving objects will act rationally and choose the shortest path between their starting points to their destinations. Based on this assumption, we introduce a statistical approach for calculating the likelihood that a certain object is located at a certain network position at a certain point of time. Using the estimation for each object in the network, it is possible to estimate the traffic density at a certain position at a certain point of time. The allowed traffic capacity, i.e. the maximal traffic density that does not lead to a traffic jam, and the expected future traffic density of an edge at a certain point of time indicate the risk of a traffic jam. If the estimated traffic density exceeds the allowed traffic capacity of the edge, we can assume that a traffic jam is very likely to occur.

Formally this paper offers a solution to the following problem: Given a set of moving objects in a road network, we want to estimate the traffic density (i.e. the expected number of objects located at a road segment at the same time) for all network segments at a any future point of time. Besides the current position of each object, we additionally assume to have a short time trajectory of each object, i.e. the recently visited network segments of each object in the network. Furthermore, it is assumed that the moving objects act rationally, e.g. each object moves along the shortest path between its starting point and its destination.

The road network is represented as a directed weighted graph G(V,E). V denotes the set of vertices that correspond to street crossings or points connecting two intersecting road segments, and E denotes the set of directed edges that connect adjacent vertices and

correspond to road segments. A weight is assigned to each edge that reflects the time any object requires to pass over the corresponding road segment. As an alternative, the distance of the way between the two adjacent vertices is used as weight and the time an object requires to traverse this segment is calculated by assuming some average speed on the road network. If the future trajectory of an object is known, its future location can be estimated by assuming an average speed at each road segment. Without this information, we would have to consider each possible location of the object for the future point of time. Since the number of possible future locations is often increasing strongly with the length of the time interval between now and the time of prediction, the prediction accuracy becomes rather small for prediction times that are not within the close future. However, assuming that each object moves along a shortest path and that the recently traversed trajectory is known for each object, it is possible to significantly restrict the potential destinations. Thus, our prediction model offers stable prediction results for a much longer period of time. The main contributions of this paper are:

- A statistical traffic model that can be used to predict the traffic density in a network at any edge.
- A method to integrate the short time history of the network to significantly improve the prediction accuracy and allows to make useful predictions over a significant period of time.
- Suitable algorithms and data structures to efficiently compute the prediction of the traffic density based on the provided information.

The remainder of this paper is organized as follows. The next section contains a brief overview over existing approaches for traffic analysis and traffic jam prediction. In section 3, we introduce our statistical approach to estimate the traffic in a network at a certain position and at a certain point of time. An efficient method to speed up the prediction which is based on a suffixtree is introduced in Section 4. In Section ??, we experimentally show the capability of our approach to make useful predictions about the traffic density. Furthermore, we illustrate the efficiency of our new algorithm when calculating these predictions and finally conclude the paper with section ??.

2 Related Work

In recent years, a lot of work has been published in the field of traffic data mining. One important problem in traffic mining is to detect areas with a significant high load of traffic. Some work has been published for the detection of traffic jams. Approaches for traffic jam detection are proposed in [?] and [?]. Both works address the problem of clustering trajectories, namely sets of short sequences of data like movements of objects. The resulting clusters indicate routes with potentially high traffic load as the clusters represent sets of objects that simultaneously move nearly the same route. While in [?] a model-based clustering algorithm is proposed that clusters trajectories as a whole, the approach proposed in [?] works on partitions of trajectories. Each trajectory is first partitioned into a set of line segments. Afterwards, similar line segments of different trajectories are grouped together in order to discover common subtrajectories from a trajectory database. An important advantage of [?] against [?] is that it can also detect routes that do not necessarily span the complete object trajectories. Normally, the trajectory of objects moving in a traffic network are very long compared to the sections which form routes with high traffic, so only reasonably small parts of a trajectory contribute to such routes.

Another approach for traffic jam detection is addressed by X. Li et al. in [?]. Their approach tries to discover hot routes in a road network. Hot routes are road segments that frequently or even regularly have a high traffic density and mostly lead to a traffic jam problem. The detection of hot routes is an important problem because each larger city has such hot routes that regularly block the traffic flow at rush hour and thus, traffic participants spend long times waiting in traffic jams. While the approaches proposed in [?] and [?] are individual traffic analysis as the traffic is computed by observing the motion of single individuals, the approach in [?] is based on the FlowScan algorithm which is a kind of aggregate traffic analysis. It is able to extract *hot routes* by means of observing the traffic flow over some adjacent road segments. The algorithm does not completely fall into the category of aggregated traffic analysis, since it considers more than the pure density of traffic on single road segments. It is merely a mixture between the individual and aggregated traffic analysis.

Further approaches concerning traffic jam detection are based on the detection of dense areas of moving objects as proposed in [?]. This approach tries to find moving clusters in moving object databases. The difference of the addressed problem compared to clustering trajectories is that the identity of a moving cluster remains unchanged while its location and content may change over time. The same usually holds for traffic jams, in particular if the traffic jam is due to an obstacle that slows down the traffic. There are normally individuals that pass the obstacle at the beginning of the traffic jam, and thus, leave the traffic jam and those which arrive at the end of the traffic jam. Consequently, the contributors of the traffic jam change over time while the identity of the traffic jam remains.

A quite similar problem is addressed in [?] where areas of moving objects that remain dense in a long period of time are detected. This approach is quite related to our approach as it addresses predictions of dense traffics where the prediction concerns any time slot in the future. Furthermore, like in our approach the predictions are made on the basis of observations of the current motion. However, there is a big difference from our approach concerning the assumption of available information of the object motions. Previous traffic prediction and traffic detection methods assume that the traffic motion and thus, the object trajectories are known in advance. However, the future trajectory of an object is usually unknown in advance in our application scenarios.

Another challenging problem is the detection of general traffic patterns. There exist several approaches for traffic prediction by means of historical observations which are based on regression analysis as proposed in [?]. Regression can be used to predict the future motion of individual objects as long as they do not move in a restricted environment as in our application. Another method concerning traffic prediction based on current traffic observations is the approach presented in [?]. In this work, the current traffic data is derived from a sensor network measuring traffic at certain locations in the traffic network. In the framework proposed in [?], the sensor network includes about nine hundred measurement stations. The data is collected in a data warehouse and used to infer interesting patterns. This kind of system may be used to learn patterns on the observed data which could be used to predict traffic jams. This method falls into the category aggregate analysis and mainly differs from our approach as it aggregates the traffic at certain road segments instead of observing single individuals.

Another field related to traffic mining is graph mining which has attracted a lot of attention in recent years. At first sight, graph mining seems to be closely related to traffic mining as traffic flows normally occur in a network graph, e.g. a road network or the internet. Hoverer, most graph mining approaches deals with the topological structure within graphs or subgraphs. A lot of graph mining approaches aim at finding interesting patterns within graphs. A comprehensive survey of graph mining techniques is given in [?]. A topic being related to our approach is the discovery of *center-piece* subgraphs [?] though the paper still works on static network graph topology. The center-piece subgraph problem is to find the node(s) and the resulting subgraph, that have strong connections to all or most of a given set of query nodes. Usual applications are connectivity mining in social networks, gene regularity networks and viral marketing. In a traffic network, we usually have certain places of preferred travel destinations or starting points. Such kind of *hot spots* can be malls, theme parks, city centers, commercial centers, conjunction to highways and so on. These kind of hot spots can be used as query points in order to find *center-piece sub*graphs which indicate places of expected high traffic. Similar measures like "Closeness Centrality" and "Betweenness Centrality" [?] which are traditionally used for mining in biological interaction networks can be applied to identify road segments with high risk of traffic jams.

3 Statistical Traffic Model

In this chapter, we will formalize our view on traffic networks and the traffic that can be observed on them. Furthermore, we will discuss a statistical model that allows to predict future states of the network under the knowledge of the current state and a short time history.

3.1 Traffic Density in a Network A traffic network is modeled as a graph G(V, E) where the vertices represent destinations and crossings. The edges represent ways or streets between the vertices. A walk is a sequence of edges $w = (e_1, ..., e_n)$ where successive edges are connected, i.e. $e_i = (v_l, v_k) \Rightarrow e_{i+1} = (v_k, v_m)$ with $v_l, v_k, v_m \in V$. An object o may travel on this network from one vertex v_1 to another one v_2 by following some walk $w = (e_1, ..., e_n), e_i \in E$ where e_1 is starting with v_1 and e_n is ending with v_2 .

The point of time t_i where the object reaches v_2 depends on the speed the object is traveling at on each edge. Therefore, we assume that there is a maximum speed for objects traveling on a certain edge e_i $speed_{e_i}$. Knowing the length of edge e_i $length(e_i)$, we can determine the time it takes object o to travel from v_l to v_k via e_i . Thus, given the walk $w = (e_1, \dots e_n)$ we can calculate the time it takes object o to follow w by

$$time(o, w) = \sum_{i=1}^{n} \frac{length(e_i)}{speed(e_i)}$$

To find out the position of object o at time t traveling on a walk $w = (e_1, ..., e_n)$, we have to calculate the $time(o, (e_1, ..., e_i))$ for i = 2 to i = n. We can stop at the first edge for which $time(o, (e_1, ..., e_i))$ is larger than t, because o will not reach e_i in a time shorter than t. As a result, we know that o will travel on edge e_{i-1} at time t.

After describing the movement of individual objects in the network, we will turn to describing the complete state of the network. Thus, we define the density on edge e_i at time t as the number of objects traveling on e_i at time t. Formally, the density is defined as:

DEFINITION 3.1. (TRAFFIC DENSITY) Let G(V, E) be a traffic network and let $O = o_1, ..., o_m$ be a set of objects traveling on the network. Furthermore, let $\rho : E \times O \rightarrow$ $\{0, 1\}$ be the following indicator function

$$\rho(o, e) = \begin{cases} 1 & \text{if o is on } e \\ 0 & else \end{cases}$$

Then, the traffic density on edge e is defined as:

$$density(e) = \sum_{i=1}^{m} \rho(o_i, e)$$

Clearly, it is possible to determine the density of each edge at the current time t when observing the network.

Additionally, it is possible to compute the density for any future point of time $t + \Delta t$ if all objects $O = \{o_1, ..., o_m\}$ and their corresponding paths w_{o_i} are already known at time t. As mentioned above the position of object o_i at the point of $t + \Delta t$ can be derived easily when knowing the walk o_i is traveling on.

The problem of traffic prediction is caused by the absence of knowledge about the walk w an object o is traveling on. In other words, we can observe each object on the network at time t but without knowing its route, we cannot exactly tell its position at time $t + \Delta t$.

Though we cannot tell the exact density of each edge at some future time $t+\Delta t$, it is possible to calculate an expected density employing the available knowledge about the objects and their behavior. Generally, our model for determining the expected density is based on the probability that a given object o is traveling on edge e at time $t + \Delta t$, $Pr[o, e, t + \Delta t]$.

This probability depends first of all on the existence of a walk that allows o to travel on edge e at the time of prediction $t + \Delta t$. If there is no walk allowing o to reach e in Δt time, then $Pr[o, e, t + \Delta t]$ can be considered to be zero.

After finding all walks $W = \{w_1, ..., w_l\}$ that allow o to be at e at time $t + \Delta t$, we can sum up the likelihoods $Pr[o, e, w_i]$ that o would take walk w_i :

$$Pr[o, e, t + \Delta t] = \sum_{i=1}^{l} Pr[o, e, w_i]$$

To determine $Pr[o, e, w_i]$, we assume that w_i is the result of a Markov chain on the network where the vertices correspond to the states and the edges to the allowed transitions. The chain is started at the current position of o. Each time o reaches a new vertex v, o has to decide for one of deg(v) + 1 options. deg(v) denotes the degree of v, i.e. the number of adjacent edges. Thus, an object can either stop traveling at the vertex v or take any of the adjacent edges to continue its journey. To find out the likelihood that o follows the walk w, we need to assume a probability distribution describing the likelihood of each of these options. Formally, we can calculate the probability that object o follows walk $w = (e_1, ..., e_n)$ where $start_i$ and end_i denote the starting and ending node of edge e_i as:

$$Pr[o, e_n] = \prod_{i=1}^{n} Pr_o[end_i | start_i]$$

 $Pr_o[end_i|start_i]$ is the likelihood that o enters e_i under the condition of being previously on node $start_i$. Let us note that we do not distinguish whether *o* intends to stop at the end_i or continues its travel. In a classical Markov chain $Pr_o[end_i|start_i]$ usually depends on the previously visited edge e_{i-1} . However, in order to keep our framework as general as possible, we do not limit our method to a certain type of distribution and thus, allow arbitrary probability distributions. For example, we might assume that the underlying probability distributions are uniform. In this case, the likelihood that ois taking walk w follows the random walk assumption, i.e. at each node an object would take any of the given options with the same likelihood with no regard of any global destination. However, since objects in a traffic network usually behave more rationally, we will introduce more sophisticated probability distributions in the next subsection.

After describing the likelihood $Pr[o, e, t + \Delta t]$ that object o will be at edge e in Δt time, we can now calculate the expected density for edge e at $t + \Delta t$.

DEFINITION 3.2. (EXPECTED DENSITY) Let G(V, E)be a traffic network and let $O = \{o_1, ..., o_m\}$ be a set of objects traveling on the network. Then, the traffic density on edge e at time $t + \Delta t$ is defined as:

$$density(e, t + \Delta t) = \sum_{i=1}^{m} Pr[o_i, e, t + \Delta t]$$

The Shortest Path Assumption Though the 3.2expected density allows us to predict the expected state of a traffic network for any future point of time, its applications pose serious problems. First of all, the prediction is strongly dependent on the underlying probability distributions. Thus, if these distributions do not model the behavior of the objects well enough, the expected density will significantly differ from the real density after a short period of time. A second problem is the computational cost of determining all walks between the current position of an object o and a future position e. The number of possibilities we have to check is exponentially increasing with Δt . Thus, finding all walks allowing o to travel on edge e an $t + \Delta t$ is very expensive for larger values of Δt .

Fortunately, the random walk assumption made above is not realistic for most traffic networks and we can employ more realistic assumptions to derive more suitable probability distributions and reduce the number of walks.

For example, a driver traveling from New York to Los Angeles would not randomly decide at each highway intersection in which direction he drives next. The reason for the more rationale behavior in traffic networks is that each object has usually a predefined destination, it wants to reach as fast as possible. Furthermore, the topology of the network is known to each object and thus, the object does not have to stray through the network until it accidentally reaches its destination. Since each object wants to reach its destination as fast as possible, we can assume that each object travels along a shortest path where each edge e is weighted by the time it takes to traverse it, i.e. $\frac{lenght(e)}{speed(e)}$. A path in contrast to a walk is not allowed to contain the same vertex twice. We will refer to this observation as the *Shortest* Path Assumption.

Though the general framework for computing the expected density can remain unchanged, the shortest path assumption has a major impact on the quality of prediction and the computational complexity.

A first implication is that in order to determine whether object o might be at edge e after the time period Δt , we only have to consider the shortest paths of the current position of o to the end vertex of e. If there is no path ending with edge e, then o travels on edge ewith a probability of 0%. If e is the last edge of some shortest path, we can calculate the time period o would travel on e. Only if $t+\Delta t$ is within this time period, it is possible to observe o on edge e at the time of prediction $t+\Delta t$. Let us note that it is not necessary to consider any other shortest path because an object traveling on any other shortest path must arrive at the end of e at the same time. To conclude the shortest path assumption significantly reduces the number of walks that have to be considered.

A further implication of the shortest path assumption is that it is possible to find meaningful probability distributions that can be used to determine the likelihood that object o travels along path p.

We know that each object o heads towards one predefined destination v_o^{dest} . Furthermore, we know that otravels along a shortest path to reach v_o^{dest} . Thus, the set of all possible paths o could follow, is the union of all shortest paths to any possible destination. Now the likelihood that o travels along path $p = (v_1, ..., v_n)$ depends on the probability that v_n is o's target, $Pr_{dest}[o, v_n]$. Without further knowledge we might assume that each destination is equally likely. Additionally, it is possible to increase the likelihood of more popular destination to integrate domain knowledge. Let us note that in the case that there is more than one shortest path leading to v_n , we assume that all paths are equally likely.

After assuming a distribution over all destinations, it is possible to derive local probability distributions that can be employed to estimate the likelihood that object o travels on a certain edge e. Therefore, we need to sum up the probabilities for each shortest path containing edge e. Formally, we can define this probability as follows:

DEFINITION 3.3. (VISITING PROBABILITY) Let

G(V, E) be a traffic network, let o be an object having the current position v_{start} and, let $sp(v_{start}, v)$ denote the set of shortest paths from v_{start} to any other vertex $v \in V$. Furthermore, let $\hat{P}_{v_{start}}(e_n) = \{p | p \in sp(v_{start}, v) \land v \in V \land e_n \in p\}$ be the set of all shortest paths beginning with v_{start} and containing the edge e_n . Now, the probability that o follows the path $p = (v_{start}, ..., v_k) Pr_o[p]$ is defined as:

$$Pr_o[p] = \frac{1}{|sp(v_{start}, v_k)|} \cdot Pr_{dest}[o, v_k]$$

where $Pr_{dest}[o, v_k]$ is the likelihood that v_k is the destination of object o. Then, the probability under the shortest path assumption that object o travels on edge e_n is defined as follows:

$$Pr_{sp}[o, e_n] = \sum_{p_i \in \hat{P}(v_{start}, e_n)} Pr_o[p_i]$$

The probability $Pr_{dest}[o, v]$ describing the likelihood of each possible destination has an important impact on the accuracy of the prediction. Furthermore, under the shortest path assumption this likelihood depends on the path $p_{history}^o$, i.e. the path o has already traversed until the current point of time. If $p_{history}^o$ is unknown, we generally have to assume that all vertices are possible destination of o. However, knowing $p_{history}^o$ allows us to prune some of these destinations. Since o is traveling on a shortest path, we can exclude all destinations for which there exists no shortest path starting with $p_{history}^o$. Thus, knowing the previous movement of each object o significantly reduces the number of possible destinations and thus, allows us to find a better estimation of $Pr_{dest}[o, v]$.

To conclude, the shortest path assumption can be derived from assuming that all objects have knowledge of the network topology and try to reach a certain destination as fast as possible. Based on the shortest path assumption, we can derive more reasonable probability distributions for the decisions each object makes at some vertex. Thus, it is possible to find a more suitable expected density for the edges in the traffic network.

4 Efficient Traffic Prediction

In the previous chapter, we defined the expected density for single edges in a traffic network at a certain time of prediction. In this chapter, we will turn to calculating the complete density in a network at some future point of time consisting of the expected densities of each edge in the network. After introducing a straight-forward method to calculate this expected network density, we will introduce a data structure allowing a much more efficient computation of density predictions.

4.1 Traffic Density Prediction The goal of our approach to is to predict the state of a traffic network for a future point in time or even a time period in the future. Therefore, we first of all formalize the expected density in a traffic network.

DEFINITION 4.1. (EXPECTED NETWORK DENSITY)

Let G(V, E) be a traffic network and let $O = \{o_1, ..., o_m\}$ be a set of objects traveling on G under the shortest path assumption. For each object $o_i \in O$, we know a short time history $p_{history}^o$ containing the path o_i has traversed before the current time t. Furthermore, the destination of each object o_i is unknown. Then, the **Expected Network Density** at time $t + \Delta t$ is defined as the set of expected densities for each edge e: density $(e, t + \Delta t)$.

The Expected Network Density consists of the complete traffic density that can be expected to be

observed at some future point of time.

In the following, we will discuss a straight-forward method for calculating the expected network density, i.e. the expected density of each edge in the network at the time of prediction $t + \Delta t$.

The basic idea of the following method is to determine all possible positions for each object *o* at prediction time $t + \Delta t$. Thus, we increase the density of each edge e by the probability $Pr_{sp}[o, e]$ if o might visit e at the time of prediction. To find out all possible positions and their corresponding likelihoods, we first of all have to determine all possible destinations. As mentioned before, the number of possible destinations depends on the path $p^o_{history}$ that o has already traversed. The refore, we start with the first known position of o, i.e. the first node in $p_{history}^{o}$, and employ Dijkstra's algorithm to determine all shortest paths to any other node in the network. Now, to determine all possible positions of o at time $t + \Delta t$, we only have to consider the shortest paths being extensions of $p_{history}^{o}$. Each of these extending paths leads to a still possible destination. Thus, we follow each of the paths p for the time period Δt and thus, determine a possible position of object o. Now, the expected density of the edge corresponding to this position is increased by $Pr_{sp}[o, p]$, i.e. the likelihood that o travels along path p. After processing each possible position for each object in the system, the expected network density is derived.

A variation of this method can be applied if we are not only interested in the traffic density at a special point of time $t + \Delta t$, but in the expected density at all points of time between t and $t + \Delta t$. In this case, the prediction of the expected density is represented by a time series displaying the expected change of traffic on a given edge. However, computing the time series is quite similar to computing a single prediction. For each path p, extending $p_{history}^o$, we traverse p and update the edges of p for the period of time o might travel on them. Whenever o could enter a new edge the expected density is increased by $Pr_{sp}[o, p]$. Correspondingly, the expected density has to be reduced each time o leaves some edge e.

Though this method can be employed to determine predictions according to the traffic model introduced in section 3, it has serious short-comings from a computational point of view. The problem is that in order to determine the possible paths of object o, it is always necessary to consider each node of the network and determine all possible shortest paths starting with its first known position. This poses an enormous computational overhead because some of the paths are computed for multiple objects. However, since the usefulness of a prediction is rather perishable, a fast computation of the



Figure 2: Example of a network graph and the corresponding suffix tree used to efficiently compute an objects probability distribution.

prediction is mandatory. Thus, in order to derive predictions in efficient time, a solution has to be found avoiding this computational overhead at prediction time and allowing efficient density predictions for the complete network.

4.2 A Shortest Path Suffix Tree In the following, we will present a data structure that is significantly speeding up the computation of the expected network traffic density. The core idea is to store all possible shortest paths in a compact data structure. Thus, the computation of shortest paths at prediction time can be avoided.

Assuming that there exists an unique identifier denoting each node in the network, we can use these node identifiers as alphabet and represent each path as a string over this alphabet. Our algorithm needs an efficient way to determine all shortest paths being extensions of the already observed history of a given object *o*. Considering each shortest path as string, we need to find a way to efficiently determine all suffixes extending the prefix represented by $p_{history}^o$. Therefore, we propose to store all shortest paths that can be found in the given network in a suffix tree.

The suffix tree is well known in text processing and bio informatics for its space-efficient storage of massive amounts of string data. Formally, a suffix tree ST for string S = S[0..n - 1] of length n over the alphabet Ais a tree with the following properties:

- ST has exactly n + 1 leaf nodes, numbered consecutively from 0 to n
- all internal nodes (except the root) have at least two children.

- edges spell non-empty strings
- all edges from the same node start with a different element of A
- for each leaf node i, the concatenation of all edges from the root node to i matches S[i..n-1]

In order to employ the suffix tree for our problem, we store all shortest paths in the given network in the suffix tree. Therefore we use the all-pair-shortest-pathalgorithm by Floyd and Warshall to efficiently derive all possible shortest paths. Afterwards, all shortest paths are converted to strings over the alphabet of node identifiers and stored in the suffix tree. In this suffix tree, each direct son of the root represents a vertex v in the network and the corresponding sub tree represents all shortest paths starting with v. Let us note that each path in this sub tree corresponds to a shortest path and the paths to the leaf node represent shortest paths that are maximal, i.e. it is not possible to extend these paths to any longer shortest path. Each inner node v_n of the suffix tree represents a crossing in the network where some object o could arrive after traversing the path corresponding to its history $p^o_{history}$. The sons of v_n represent all possible shortest paths extending $p_{history}^{o}$. Figure ?? illustrates an example of an object traversing a network graph. The corresponding suffix tree representing all possible destinations is depicted in Figure ??.

To efficiently calculate the expected network density, it is not sufficient to directly access the information about the existence of a shortest path. Additionally, the likelihood that an object o follows some path p is of great importance. Therefore, we additionally store the probability distributions describing $Pr_o[end_i|start_i]$ in the tree, i.e. the likelihood o would turn into the direction of the node end_i after reaching $start_i$. In our model, this probability depends on the cumulated likelihood that o takes any of paths being extensions of the edge $(start_i, end_i)$. In the tree, these paths are represented by the sub tree under the node end_i . To speed up the computation of the likelihood of each path during prediction, we add up the likelihoods of possible directions right after generating the tree. Therefore, we first of all mark each ending point of each path, with the likelihood that o would take this path. Let us note that inner nodes are valid ending points as well. Afterwards, we assign the cumulated likelihood over all paths extending edge \hat{e} to \hat{e} in the tree. Let us note that for any node e in the network there usually exist multiple edges \hat{e} in tree, one for each possible prefix. Due to this modification, it is now possible to calculate the likelihood that some object o might visit edge e while traversing the tree.



Figure 3: Traffic network graph with simulated cars used as experimental test bed.

To calculate the expected network traffic density using the proposed shortest path suffix tree, we can proceed as follows. For each object o, we enter the tree traversing along the string corresponding to the already observed path of $o p_{history}^o$. After reaching the node in the tree corresponding to the current position of o, we can derive all possible positions of o at $t+\Delta t$. Therefore, we traverse every path in the sub tree under the current position of o and calculate the likelihood that o would travel this path during traversal. Traversing each path is stopped if extending the path would demand more time than Δt . Finally, we add the current likelihood to the expected density at the edge corresponding to the current position of o and continue by extending the next path in the sub tree.

To conclude, employing a shortest path suffix tree allows us to avoid shortest path computations during traffic density prediction. Furthermore, the number of edges that have to be traversed for prediction is also reduced to necessary sub paths.

5 Experimental Evaluation

In this section we show the capability of our approach to make useful predictions about the traffic density and illustrate the efficiency of our new algorithm when calculating these predictions. For all experiments, we simulated the traffic in a realistic traffic network as depicted in Figure ?? containing about 679 road segments and 533 intersection nodes. Our traffic simulator contains about 1250 cars (illustrated by small dots in Figure ??) moving from individual starting points to their destina-



Figure 4: Prediction using a spatial temporal poisson model for the entry of new cars.

tions on a shortest path. The starting points as well as the destinations are equally distributed over the entire network graph. Here, each car moves with a certain velocity which is assumed to be constant during the whole journey. The velocities of the moving cars are randomly selected for each car and it took about 60 minutes until all cars have reached their destinations. If not stated otherwise, as soon as a car has reached its destination it was removed from the network and, thus, did not contribute to the traffic anymore.

All experiments are based on java implementations. The runtime experiments were conducted on a dual core Opteron Dual Core processor with a clock time of 2.6 GHz and 32GB of RAM.

5.1Experiments on Quality of the Traffic Density Prediction The first experiments concerns the quality of our traffic density predictions. The traffic density of a road segment is simply given by the number of cars that pass through this road segment at a certain point of time. In order to show the quality of the traffic density prediction, we continuously measured the traffic density *prediction error* during a certain range of time. The *prediction error* is computed by the difference between the predicted traffic density and the observed traffic density for a road segment. In our experiments, we use the parameter *prediction time* Δt which denotes the the forecasting horizon. In other words, the prediction time denotes the difference between the time the actual traffic is measured, (i.e., the time the traffic prediction is related to) and the time at which the traffic prediction was done.

Generally, in our experiments we only consider those cars that are existent in the road network at prediction time, i.e. all objects that enter the network graph



Figure 5: Average number of cars on a road segment.

after the time the prediction was done are not considered. However, in realistic scenarios new cars continuously enter the network. In order to evaluate traffic predictions under these circumstances, an additional statistical model would be required. For example, the entry of new objects in the network can be modeled using a spatial temporal Poisson-process. The prediction based on such a model is depicted in Figure ??. Obviously, the absolute prediction error increases with the number of new objects entering after the time the prediction was done. The rational of this is that we have no information of the new cars while the number of cars which are considered for the prediction diminishes. The expected number of cars which are considered for the traffic prediction is the difference between the predicted number of cars and the expected number of new cars. This number approaches zero when all objects considered for the traffic prediction have reached their destination. In the following experiments we only focus on objects which are present in the network at the time the prediction is done and do not allow objects to enter after that.

Since the prediction error is an absolute value measured in number of cars, the quality of the prediction depends on both the prediction error and the number of cars on the corresponding road segment. If not stated otherwise, we averaged the prediction error over a set of road segments. In order to achieve more representative results, we measured the prediction quality only for a subset of road segments. Here we left out those road segments that contain only very low traffic over the measured time. Thereby, we try to avoid that the quality results are inherently biased by road segments with low traffic which are expected to yield high prediction quality. Since there are a lot of such kind of road segments with little traffic in our traffic network we did not consider them in order to obtain fair qual-



Figure 6: Average prediction error in number of cars on a relevant road segment.

ity measurements. In the remainder, we will call the set of road segments taken into account for the quality measurements *relevant road segments*. This set contains twenty road segments.

The average number of cars on a relevant road segment is given in Figure ??. It shows that the number of cars decreases with the running time of the simulation as the cars which reach their destination are removed from the traffic simulation. Although the overall number of objects in the simulation in fact decreases monotonically, here we do not observe a monotone decrease in the number of cars because we only counted the objects on the relevant road segments which can fluctuate a little bit.

Figure ?? shows the average prediction error w.r.t. the prediction time Δt (i.e. the forecasting horizon). The figure presents two curves, one curve depicts the prediction error when considering the complete history of each car for the prediction. The other curve represents the prediction error when taking only the last two passed road segments into account. Both curves have similar characteristics, because within the prediction of the close future the error increases drastically with increasing prediction time. This is due to the fact that the number of possible locations for each car is initially very small and increases drastically when the car passes through the first crossings. But with ongoing prediction time, the absolute prediction error decreases again. The rationale of this effect is the decreasing number of cars in the simulation. Less cars in the simulation obviously lead to a smaller absolute prediction error. But it can clearly be observed that the prediction based on the complete history has a significantly better quality than the prediction based on only the last two road segments. Generally speaking, both predictions have a good prediction quality, at least in the first few minutes. At a

Road Segment Id	$0-\infty$	18-22 min	38-42 min
4	11.97	16.93	3.00
17	24.94	30.50	6.00
157	11.68	16.88	4.46
433	14.72	16.33	5.14
sum.	142.84	257.19	74.12

Table 1: Traffic table for the experiments shown in Figure ??.

prediction time of about 20 minutes the prediction error adds up to two of fourteen cars in average, which is a relative prediction error of about 14%. With increasing prediction time the relative prediction error increases rapidly, e.g. at a prediction time of about 40 minutes the relative prediction error reaches 37%.

We also measured the relative prediction error at the four most relevant road segments averaged over different prediction time intervals. The relative prediction errors better reflects the quality of the traffic prediction than absolute prediction errors. The results are shown in Figure ??. The relative prediction error denotes the quotient

absolute prediction error

traffic in terms of the number of cars

These results show that the relative prediction error is between 5% and 15% for short-term predictions and between 10% and 60% for long-term predictions when taking the motion history into account.

Additionally, we measured the traffic at specific time intervals in terms of number of cars. The results are given in Table ??. If we compare the resulting traffic of the single road segments to the average of all relevant road segments, we can see that the road segments selected for this experiment show a heterogeneous traffic. This experiment shows that the consideration of the history of each car has significant influence on the prediction quality for short-term predictions (about 20 minutes prediction) as well as for long-term predictions (about 40 minutes prediction).

In the next experiment we evaluated how the length of the history which was taken into account for each car influences the prediction quality. For this experiment we have run the simulation with a set of 500 cars. We measured the average prediction error for several prediction times by varying lengths of observed histories. The results are depicted in Figure **??**. An interesting observation is that the length of the history in terms of passed nodes has similar effect for short-term predictions and long-term predictions. A history longer than ten nodes does not make any difference in the prediction error.



Figure 7: Relative traffic density prediction error averaged over certain intervals of prediction time.



2 Nodes History Average Prediction Error per Road Segment 3 Nodes History 4 Nodes History 5 Nodes History 6 Nodes History 5.5 8 Nodes History 10 Nodes Histor 12 Nodes History 14 Nodes History 16 Nodes History [number of cars] 18 Nodes Histor 1.0 Full History 0.5 0.0 0.03 0.43 2.03 3.63 7.23 19.63 Prediction Time ∆t [min]

Figure 8: Average prediction error in number of cars for varying motion history taken into account.

Figure 9: Performance of one traffic density prediction in terms of accessed network nodes.

5.2 Experiments Concerning the Efficiency In the next experiments we show the performance comparison between the proposed prediction strategies. In particular, we compare the first solution where the future path probabilities for each car are computed at run-time with the approach using the pre-computed suffix-tree. The performance is measured in terms of the number of network nodes which have to be accessed to predict the traffic density at each road segment for one certain point of time in the future. Additionally we measured the absolute runtime required to make the prediction. The results of the number of accessed network nodes are shown in Figure **??**.

Furthermore, we evaluated the scalability of our traffic prediction approach when using the suffix-tree in order to accelerate the prediction. Figure ?? demonstrates the time required to make a 4-minute traffic-prediction for the entire road network. We measured the runtime for varying number of cars in the traffic network. Obviously, the prediction runtime increases linearly with increasing number of moving objects.



Figure 10: Performance of the traffic density prediction in terms runtime for varying number of cars.

6 Conclusions

In this paper, we proposed an approach for density prediction in traffic networks. We introduced a statistical model that is used to predict the traffic density on any edge of the network at some future point of time. Furthermore, we showed how short-term observations can be used to improve the prediction quality and how the traffic densities can be computed in an efficient way. We experimentally demonstrated that our approach achieves high prediction qualities in particular when taking the history of the moving individuals into account. However, we observed that only a quite small history suffices to reach adequate prediction qualities for both short-term and long-term predictions. In addition, our runtime experiments showed that the computation of the traffic predictions can be made in reasonable time. In the future, we plan to extend our statistical prediction model by taking further observable or even learnable motion parameters into account.

7 Extras

References

- D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. In ACM Comput. Surv., 38(1), New York, NY, USA, 2006.
- [2] D. Delling, P. Sanders, D. Schultes, and D. Wagner. Highway hierarchies star. In *In Proc. 9th DIMACS Implementation Challenge*, 2006.
- [3] E. Dijkstra. A note on two problems in connexion with graphs. In Numerische Mathematik, 1:269–271, 1959, 1959.
- [4] L. Fu, D. Sun, and L. R. Rilett. Heuristic shortest path algorithms for transportation applications: state of the art. In *Computers in Operations Research*, 33(11):3324-3343, 2006.
- [5] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In Proceedings of the 5th ACM International Conference on Knowledge Dis-

covery and Data Mining (SIGKDD), San Diego, CA, 1999.

- [6] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proceedings of the 33nd International Conference on Very Large Data Bases (VLDB), Vienna, Austria, 2007.*
- [7] M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. Tsotras. On-line discovery of dense areas in spatiotemporal databases. 2003.
- [8] N. Jing, Y.-W. Huang, and E. A. Rundensteiner. Hierarchical optimization of optimal path finding for transportaton applications. In Proceedings of the 5th International Conferenc on Information and Knowledge Management (CIKM), Rockville, MD, 1996.
- [9] S. Jung and S. Pramanik. Hiti graph model of topographical road maps in navigation systems. In Proceedings of the 12th International Conference on Data Engineering (ICDE), New Orleans, LA, 1996.
- [10] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. pages 364– 381, 2005.
- [11] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In Proceedings of the 22st International Conference on Data Engineering (ICDE), Atlanta, GA, 2006.
- [12] J. Lee, J. Han, and K. Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the SIGMOD Conference, Beijing, China*, 2007.
- [13] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In Proceedings of the 10th International Symposium on Spatial and Temporal Databases (SSTD), Boston, MA, 2007.
- [14] O. Mason and M. Verwoerd. Graph theory and networks in biology. 2006.
- [15] R. K. Oswald, W. T. Scherer, and B. L. Smith. Traffic flow forecasting using approximate nearest neighbor nonparametric regression. In *Final project of ITS Center project: Traffic forecasting: non-parametric regressions, December*, 2000.
- [16] S. Pallottino and M. G. Scutella. Shortest path algorithms in transportation models: classical and innovative aspects. In *Technical Report TR-97-06*, 14, 1997.
- [17] P. Sanders and D. Schultes. Highway hierarchies hasten exact shortest path queries. In In Proc. 17th European Symposium on Algorithms (ESA), 2005.
- [18] S. Shekhar, C.-T. Lu, S. Chawla, and P. Zhang. Data mining and visualization of twin-cities traffic data. In *Technical Report TR 01-015, Dept. of CSE, Univ. of Minnesota*, 2000.
- [19] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA, 2006.