

# Adaptive Nearest Neighbor Queries in Travel Time Networks\*

Wei-Shinn Ku, Roger Zimmermann, Haojun Wang, and Chi-Ngai Wan  
Computer Science Department, University of Southern California  
Los Angeles, California 90089. USA

wku@usc.edu, rzimmerm@usc.edu, haojunwa@usc.edu, cwan@usc.edu

## ABSTRACT

Nearest neighbor (NN) searches represent an important class of queries in geographic information systems (GIS). Most nearest neighbor algorithms rely on static distance information to compute NN queries (e.g., Euclidean distance or spatial network distance). However, the final goal of a user when performing an NN search is often to travel to one of the points of the search result. In this case, finding the nearest neighbors in terms of travel time is more important than the actual distance. In the existing NN algorithms dynamic real-time events (e.g., traffic congestions, detours, etc.) are usually not considered and hence the pre-computed nearest neighbor objects may not accurately reflect the shortest travel time. In this paper we propose a novel travel time network that integrates both spatial networks and real-time traffic event information. Based on this foundation of the travel time network, we develop a local-based greedy nearest neighbor algorithm and a global-based adaptive nearest neighbor algorithm that both utilize real-time traffic information to provide adaptive nearest neighbor search results. We have performed a theoretical analysis and simulations to verify our methods. The results indicate that our algorithms remarkably reduce the travel time compared with previous nearest neighbor solutions.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Search Process*; H.2.8 [Database Management]: Database Application—*spatial databases and GIS*

## General Terms

Algorithms

\*This research has been funded in part by NSF ITR grant CMS-0219463, equipment gifts from Intel and Hewlett-Packard, unrestricted cash grants from the Lord Foundation and by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GIS'05, November 4-5, 2005, Bremen, Germany.

Copyright 2005 ACM 1-59593-146-5/05/0011 ...\$5.00.

## Keywords

Nearest neighbor query, travel time network, location-based services, advanced traveler information systems.

## 1. INTRODUCTION

Nearest neighbor (NN) queries are of significant interest for applications that work with spatial data. A sample query could be to “find the nearest gas station from my current location.” Previous work [14, 7] has resulted in efficient techniques to compute NN queries in Euclidean space. More recently, novel algorithms [8, 17, 13] have been proposed to compute NN queries in spatial networks. These methods extend NN queries by considering the spatial network distance, which provides a more realistic measure for applications where objects are constrained in their movements. However, these existing techniques only consider static models of spatial networks: pre-defined road segments with fixed road conditions are used in computing nearest neighbors. Thus, any real-time events (e.g., detours, traffic congestions, etc.) affecting the spatial network cannot be reflected in the query result. For example, a traffic jam occurring on the route to the computed nearest neighbor most likely elongates the total driving time. More drastically, the closure of a restaurant which was found as the nearest neighbor might even invalidate a query result. This motivates the need for new algorithms which extend existing NN query techniques by integrating real time event information.

Recent advances in personal locator systems (e.g., GPS), wireless communication technologies (e.g., 802.11x), and peer-to-peer networks (P2P) have created an innovative environment that allows the exchange of real time traffic information between peers. By leveraging ad-hoc networks, traffic information can be shared in a P2P manner among mobile hosts (MH) and thus local traffic information (e.g., driving speed of vehicles) can be considered when computing NN queries. Furthermore, cellular communication enables remote *traffic information server* (TIS) access such that collecting and disseminating traffic information for a much wider area becomes possible.

In this paper we propose two novel nearest neighbor query algorithms which incorporate real time traffic information. Compared with existing work, our design leverages the communication among mobile hosts and traffic information servers to adaptively compute more accurate nearest neighbor results. The contributions of our work are:

- We introduce the concept of *travel time networks* (TTN). A travel time network integrates the real time traffic information with a static data of a spatial network. The length of each edge in the travel time network represents the driving time of each road segment. As a result, NN queries are computed on

travel time networks more realistically than on spatial road networks.

- We propose a Local-bAsed greedy Nearest Neighbor query algorithm (LANN). LANN utilizes the peer-to-peer communication model to frequently update the travel time network on each mobile host. The LANN algorithm incrementally selects the road segments to the nearest neighbor based on the most current instantiation of the local travel time network. At each step local traffic information is collected to select the best road segment leading toward the nearest neighbor.
- We also propose a Global-based Adaptive Nearest Neighbor query algorithm (GANN). A traffic information server is used to aggregate and maintain valid traffic events. Mobile hosts access and retrieve traffic information and construct a global travel time network. Nearest neighbors are computed in a best-first manner on this global travel time network.
- We report experimental results based on a simulated travel time network environment.

The rest of this paper is organized as follows. The related work is described in Section 2. In Section 3 we define the travel time network and introduce the LANN and the GANN algorithms. The experimental validation of our design is presented in Section 4. We discuss the conclusions and future work in Section 5.

## 2. RELATED WORK

In this section we review previous work related to Euclidean distance nearest neighbor queries (Section 2.1), spatial network distance nearest neighbor queries (Section 2.2), and inter-vehicle communication (Section 2.3).

### 2.1 Nearest Neighbor Query Processing in the Euclidean Space

The R-tree algorithm and its extensions [6, 16, 1] are very popular index structures for spatial query processing in the Euclidean space due to their efficiency and simplicity. To find nearest neighbors, Roussopoulos et al., [14] proposed a branch-and-bound algorithm that searches an R-tree in a depth-first manner. A best-first NN algorithm [7], proposed by Gíslí et al., keeps a heap with the entries of the nodes visited so far and the algorithm always expands the first entry in the heap. The best-first NN algorithm is optimal, because it visits only the necessary nodes for obtaining nearest neighbors. Furthermore, the algorithm reports nearest neighbors in ascending order according to their distance to the query point and can be applied under conditions without knowing the number  $k$  of queried nearest neighbors in advance.

### 2.2 Nearest Neighbor Query Processing in the Spatial Network Space

Several spatial network nearest neighbor query methods have been proposed in recent years. Huang et al. [8] proposed an algorithm to integrate spatial and connectivity information. Their approach utilizes thematic spatial constraints to restrict permitted paths, but it is inapplicable to general query processing (e.g., nearest neighbor queries). Shahabi et al. [17] proposed a method to perform nearest neighbor queries in road networks by transforming the problem into high dimensional space. However, these solutions only extend nearest neighbor queries into the spatial network space with no consideration of traffic conditions. Similarly, Kolahdouzan et al. [9] presented a novel approach to efficiently

and accurately evaluate  $k$  nearest neighbor queries in spatial network databases using a first order Voronoi diagram, but they did not take real-time traffic information into account. The in-route nearest neighbor (IRNN) query which was addressed by Shekhar et al. [18] is designed to find in-route nearest neighbors of a query object tuple which consists of a given route with a destination and a current location on the route. According to their design, the route of the query object is predefined and IRNN can navigate the query object to the NN with the smallest detour. Nevertheless, all the solutions proposed in [18] do not consider any real-time traffic conditions.

Ding et al. [4] presented a State-Based Transportation Network (SBDTN) model which can be used to describe the spatio-temporal aspects of temporally variable transportation networks. The basic idea is to associate a temporal attribute to every node or edge of the graph system so that the state at any time instant can be retrieved. We were inspired by some ideas of [4] in implementing our travel time networks. As an extension of NN queries in the Euclidean space, Papadias et al. [13] proposed two algorithms, the Incremental Euclidean Restriction (IER) algorithm and the Incremental Network Expansion (INE) algorithm to solve the problem of finding nearest neighbors in spatial network databases. Our design extends IER to compute nearest neighbors within the travel time network. We review the IER and INE algorithms in Section 3.3 due to their relevance to our design.

### 2.3 Inter-Vehicle Communication

Inter-vehicle communication (IVC) enables a vehicle (and its driver) to communicate with other vehicles (and their drivers) exchanging real time information, such as road conditions, traffic speeds, and weather hazards. Benefiting from the power capacities of vehicles, the nodes of these networks can have a virtually unlimited lifetime.

There are many previous studies about the Media Access Control (MAC)/Physical (PHY) layer [12, 10] and network layer protocols [19, 3] for inter-vehicle communication. Considering the inter-vehicle communication at a relatively short range and low cost, Ott et al. [12] proposed the use of IEEE 802.11 networks for sending and receiving high data volumes between vehicles moving at different speeds. They demonstrated their concept with a single access point with an estimated reach of 200 meters in diameter. The speed limit for vehicles under this model is able to reach 120 km/h. In contrast, the infrastructure mode of a cellular-based network (such as utilized by the OnStar service<sup>1</sup>) links the vehicle and driver to the base station with a much longer communication range. By integrating with the Global Positioning System (GPS) system, this model enables a centralized server providing location-based services.

The design of communication protocols for IVC is very challenging due to the variety of application requirements and the tight coupling between an application and its supporting protocols [11]. Therefore, there are several very recent research projects (such as FleetNet<sup>2</sup> and CarTALK 2000<sup>3</sup>) investigating next generation IVC protocols and technologies.

## 3. SYSTEM DESIGN

We outline our system architecture in this section and propose two nearest neighbor query algorithms for travel time networks: Local-bAsed greedy Nearest Neighbor query (LANN) and Global-based Adaptive Nearest Neighbor query (GANN). LANN follows the peer-to-peer model and provides incremental results for navi-

<sup>1</sup><http://www.onstar.com/>

<sup>2</sup><http://www.fleetnet.de/>

<sup>3</sup><http://www.cartalk2000.net/>

gating a mobile host to its nearest neighbor. In contrast, GANN employs Traffic Information Servers and mobile hosts can access traffic information of a larger district. The system infrastructure and assumptions are explained in Section 3.1. Next, we introduce the concept of the *travel time network* in Section 3.2. Travel time nearest neighbor queries are discussed in Section 3.3. We propose the local-based greedy nearest neighbor query algorithm in Section 3.4. Traffic event collection and distribution via TIS is illustrated in Section 3.5. Section 3.6 covers the design of the global-based adaptive nearest neighbor query algorithm.

### 3.1 System Infrastructure and Assumptions

Figure 1 illustrates the system infrastructure of our design. We are considering mobile hosts with abundant power capacity, such as vehicles, that are equipped with a Global Positioning System (GPS) for obtaining continuous position information. In addition, mobile hosts also maintain the road network data and the set of *points of interest* (POI) in local memory (for example, stored on a CD). The road network data (e.g., the US Census TIGER data set<sup>4</sup>) covers the road segments of highways, primary roads, rural roads, etc. These different road types are defined as road class attributes in the TIGER data set. Since this road class attribute does not convey the speed limit information, we define a fixed speed limit for each road class (e.g., 65 mph for highways). Furthermore, we assume that two tiers of wireless connections are available on each mobile host. The cellular-based networks (such as utilized by the OnStar service) allow medium range connections to base-stations that interface with the wired Internet infrastructure. A second type of short-range ad hoc communication protocols (e.g., IEEE 802.11x) are also supported to communicate between neighboring peers. Mobile hosts can either broadcast requests of traffic information to peers within the communication range (local solution) or send requests to the traffic information server directly (global solution). Currently there are many real-time traffic event providers (e.g., California Highway Patrol<sup>5</sup>, SIGALERT.com real-time traffic information<sup>6</sup>, etc.) which supply traffic information of many urban areas. These web sites can be easily integrated with TIS servers through Web service interfaces in the future. A local travel time network in each mobile host is thus built by integrating the information of traffic events from peers or a TIS and the local stored road network for processing nearest neighbor queries.

### 3.2 Travel Time Networks

Leveraging existing methods, we assume a digitization process that generates a modeling graph from an input spatial network. The modeling graph contains three categories of graph nodes: the network junctions, the start/end points of a road segment, and other auxiliary points (e.g., speed limit change points). However, a digitized spatial network (as shown in Figure 2a) cannot reflect real-time traffic events and this limitation decreases the accuracy of NN query algorithms [17, 13] which utilize spatial networks. In order to obtain a more accurate estimation of the nearest POI (and its travel time) to a query point  $Q$  we propose to combine real time traffic information with spatial road networks to generate *travel time networks* (TTN). A travel time network uses the travel time between nodes as the graph edge weight, rather than the network distance as in spatial networks. For example, assume that the spatial distance of edge  $e$  is five miles (the left most road segment in Figure 2a) and the speed limit of  $e$  is thirty miles per hour (mph). Hence, we compute the minimum driving time between nodes  $A$  and  $B$  to be

<sup>4</sup><http://www.census.gov/geo/www/tiger/>

<sup>5</sup><http://cad.chp.ca.gov/>

<sup>6</sup><http://www.sigalert.com/>

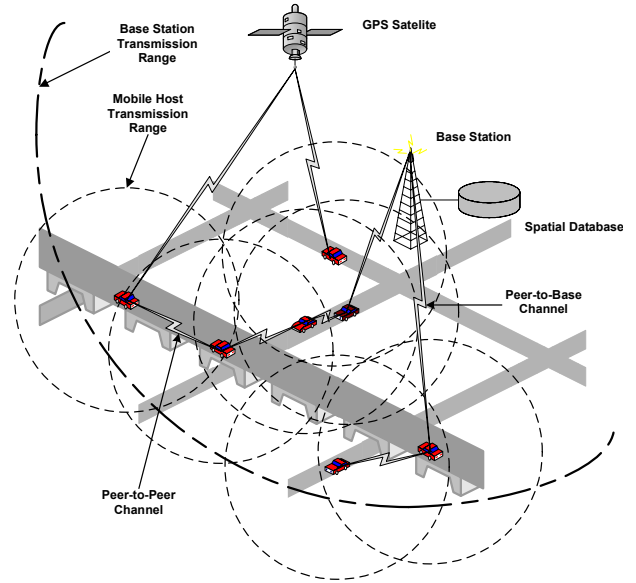


Figure 1: The system infrastructure.

ten minutes and we set this value as the distance between nodes  $A$  and  $B$  on the travel time network (the left most road segment in Figure 2b).

With travel time networks, real-time events can be readily integrated into affected road segments by converting their effect to time. For example, if the driving speed of a road segment is slower than its speed limit because of a traffic congestion, the travel time between its start and end points can be dynamically updated to reflect the congestion. In addition if a road segment is closed after a traffic accident, this road segment can be temporarily removed from its travel time network. Since turn restrictions (e.g., right turn prohibition) can be modeled by including extra nodes in the spatial network, we do not consider them as real-time events here.

### 3.3 Travel Time Network Nearest Neighbor Queries

In the real world, mobile objects often move on pre-defined networks (e.g., roads, railways, etc.). In this scenario, the spatial network distance provides a more exact estimation of the travel distance between any two objects than Euclidean distance. Papadias et al. [13] have proposed the Incremental Euclidean Restriction (IER) algorithm and the Incremental Network Expansion (INE) algorithm to solve spatial network nearest neighbor queries. Here we extend the IER algorithm to solve NN queries on travel time networks, because according to [13] IER has a better performance in denser, more regular networks (e.g., city blocks) and our algorithms are most likely to be applied in urban areas. We propose to use travel time networks to replace spatial networks and take driving time as the length between two nodes for utilizing real-time traffic information. We call the modified method *travel time network nearest neighbor query*.

**Incremental Euclidean Restriction** The IER algorithm is based on the multi-step  $k$ NN technique [5, 15]. To execute a nearest neighbor search for query point  $Q$ , IER first retrieves the first Euclidean distance nearest neighbor  $n_1$  of  $Q$  and computes the Euclidean distance  $ED(Q, n_1)$ . Next it calculates the network distance from  $Q$  to  $n_1$ ,  $ND(Q, n_1)$ . Subsequently the IER algorithm can use  $Q$  as the center to draw two concentric circles with radii  $ED(Q, n_1)$

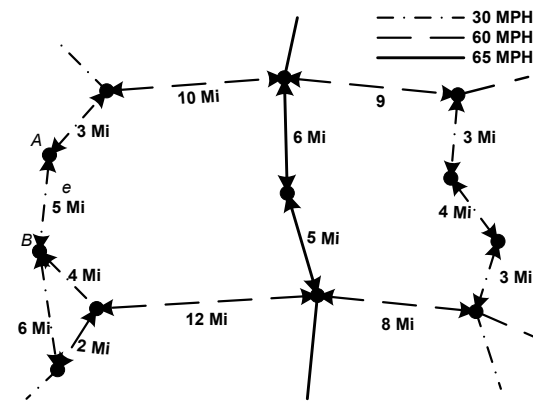


Fig. 2a. A spatial network with different road classes. Thick lines have higher speed limits.

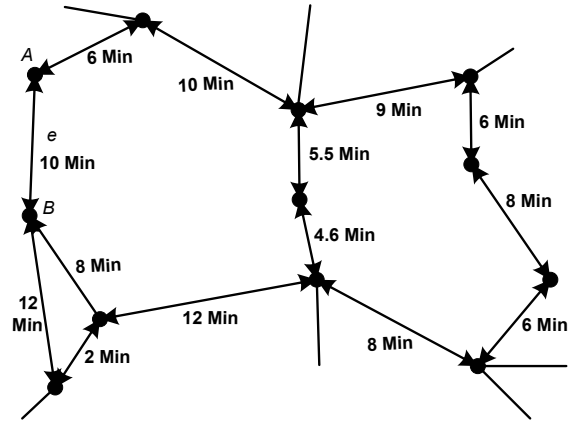


Fig. 2b. A traffic time network with the driving time between nodes.

Figure 2: A spatial network (left) and its traffic time network (right).

and  $ND(Q, n_1)$ , respectively. Due to the *Euclidean lower bound property* (i.e., for any two nodes  $i$  and  $j$ , their Euclidean distance  $ED(n_i, n_j)$  always provides a lower bound on their network distance  $ND(n_i, n_j)$ ). Objects closer to  $Q$  than  $n_1$  in the network must be within the circle making use of  $ND(Q, n_1)$  as its radius. Therefore, the search space becomes the ring area between the two circles as shown in Figure 3a. In the next iteration, the second closest object  $n_2$  is retrieved (by Euclidean distance). Since in the given example  $ND(Q, n_2) < ND(Q, n_1)$ ,  $n_2$  becomes the current candidate for spatial network nearest neighbor and the search upper bound becomes  $ND(Q, n_2)$ . This procedure is repeated until the next Euclidean nearest neighbor is located beyond the search region (as  $n_3$  in Figure 3b).

**Incremental Network Expansion** Figure 4 illustrates the INE algorithm. The points represent the nodes in the modeling graph and triangles denote POIs. The closest POI to the query point  $Q$  is  $n_5$ . The subscripts of the POIs ( $n_1, n_2, \dots, n_5$ ) are in ascending order according to their Euclidean distance to  $Q$ . The INE algorithm performs network expansion from  $Q$ , and examines the POIs in the order in which they are encountered. In this example, INE first locates the segment  $p_1p_2$  where  $Q$  finds and then retrieves all POIs on  $p_1p_2$ . Since there is no POI on  $p_1p_2$  in this example, the point  $p_1$ , which is the closest to  $Q$  is expanded. There is no POI that can be found on  $p_1p_9$  and  $p_9$  is inserted into  $Queue = \langle (p_2, 5), (p_9, 9) \rangle$ . The expansion of  $p_2$  leads to  $p_3$  and  $p_4$  and then  $Queue = \langle (p_3, 9), (p_4, 9), (p_9, 9) \rangle$  and POI  $n_5$  is discovered on  $p_2p_4$ . The network distance from  $Q$  to  $n_5$  is 8 and it provides a bound on the search space. The search terminates here since the next entry  $p_3$  in  $Queue$  has a longer distance.

### 3.4 Local-based Greedy Nearest Neighbor Queries

The travel time network enables the integration of real time traffic information into a spatial representation. Since the cellular-based communication is much more expensive than the short-range ad hoc communication, traffic information can be exchanged in a local area with a much lower cost. Based on this observation, we propose a Local-bAsed greedy Nearest Neighbor query algorithm (LANN). LANN relies on exchanging local traffic information be-

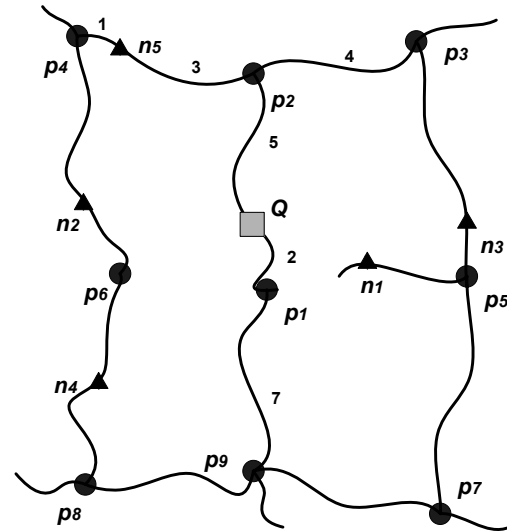


Figure 4: Nearest neighbor search in a spatial network environment with the INE algorithm.

tween peers to build a travel time network. With LANN, the mobile host first computes the nearest neighbor via executing a travel time network nearest neighbor query. Next, the mobile host incrementally updates local traffic information and selects a road segment correspondingly as the shortest path to the computed nearest neighbor. As the mobile host begins to navigate on a road segment, it broadcasts requests to collect local traffic information from peers within the ad hoc communication range. Only the traffic information of the surrounding road segments is requested. In the case that the traffic information cannot be collected from peers, the default speed limits of the corresponding road segments are used. A travel time network, which evaluates each surrounding road segment as the sum of the actual travel cost and a heuristic travel cost, is hence built up. The travel time network utilizes the travel time of sur-

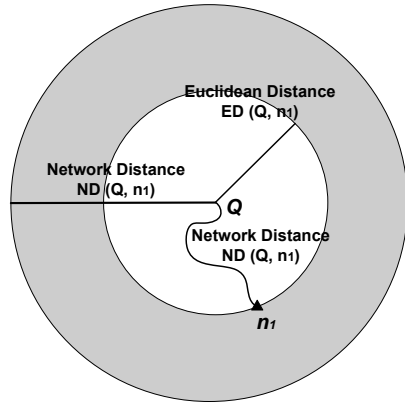


Fig. 3a. The 1<sup>st</sup> Euclidean NN.

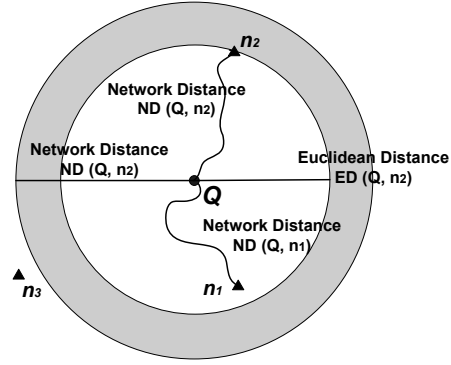


Fig. 3b. The 2<sup>nd</sup> Euclidean NN.

**Figure 3: Nearest neighbor search in a spatial network environment with the IER algorithm.**

rounding road segments as the actual cost  $g$ . The heuristic cost  $h$  is computed as the Euclidean distance from the end of each surrounding road segment to the destination point divided by a heuristic travel speed (e.g., the average travel speed on the spatial network). The mobile host selects the road segment with the cost  $f = \text{MIN}(g + h)$  as the shortest path to the computed nearest neighbor and starts to navigate on that road segment.

Figure 5 demonstrates an example of a TTN and LANN computation. Assume a mobile host at location  $A$  executes a travel time NN query and the computed nearest neighbor from the spatial road network is at location  $B$ . Figure 5a shows the corresponding spatial road network and the Euclidean distance from the end of each surrounding road segment of location  $A$  ( $AC$ ,  $AD$ , and  $AE$  in this example) to the destination  $B$ . Next, the mobile host broadcasts to collect the traffic information on road segments  $AC$ ,  $AD$ , and  $AE$  from peers. Assume the traffic speed of road segment  $AC$  is 30 mph, the speeds of  $AD$  and  $AE$  are 60 mph, and the average travel speed on the spatial network is 55 mph. A TTN is constructed based on these traffic information as shown in Figure 5b. The heuristic costs of  $AC$ ,  $AD$ , and  $AE$  are computed with the average travel speed on the spatial network. The length of each edge in the TTN represents the corresponding travel time. Hence the cost of  $AC$  is the actual cost (8 minutes) plus the heuristic cost (32.7 minutes), which in total equals 40.7 minutes. Similarly, the cost of  $AD$  is  $12 + 16.4 = 28.4$  minutes and the cost of  $AE$  is  $2 + 38.2 = 40.2$  minutes. Road segment  $AD$ , which has the minimum cost, is selected in this example.

LANN is executed in an incremental manner: when the mobile host reaches the end of the selected road segment, it broadcasts again to collect local traffic information from peers and update the travel time network. Next, the mobile host selects a road segment to continue navigating based on the updated TTN. The mobile host keeps executing the algorithm until it reaches the computed nearest neighbor. The complete algorithm of LANN is shown in Algorithm 1.

### 3.5 Traffic Event Collection and Distribution of the Traffic Information Server

In order to support the Global-based Adaptive Nearest Neighbor Queries, a traffic information server has to maintain valid traffic events for mobile hosts to access. Traffic events can be broadly

---

**Algorithm 1** Loba-based Greedy Nearest Neighbor ( $Q$ )

---

- 1: /\*  $Q$  is the query mobile host \*/
- 2: Execute a spatial road network NN query for retrieving the nearest POI and the shortest path to it.
- 3: Take the returned nearest POI as the destination,  $D$
- 4: **for** each surrounding road segment in the TTN **do**
- 5:   select the minimum of actual traffic time cost  $T_g$  plus heuristic traffic time  $T_h$
- 6: **end for**
- 7: Take the returned road segment as the current route,  $S_{route}$
- 8: **while**  $\text{Dist}(Q, D) \neq 0$  **do**
- 9:   **repeat**
- 10:     Navigate the mobile host to  $D$
- 11:     **until** the mobile host reaches the end of  $S_{route}$
- 12:     broadcast to update the TTN
- 13:     **for** each surrounding road segment in the TTN **do**
- 14:       select the minimum of actual traffic time cost  $T_g$  plus heuristic traffic time  $T_h$
- 15:     **end for**
- 16:     Take the returned road segment as the current route,  $S_{route}$
- 17:   **end while**

---

classified into four categories:

- Category 1 - Congestion Events: The real traffic speed of a road segment is much lower than the speed limit. This condition is usually caused by traffic accidents or traffic jams.
- Category 2 - Detour Events: A road segment is closed and the mobile host has to detour.
- Category 3 - Closure Events: The selected POI is closed and the mobile host has to search for another nearest POI.
- Category 4 - Recovery Events: A mobile host can recover its local TTN from previous events: a traffic congestion has been relieved, a detour has been removed, or a POI is reopened.

Since mobile hosts send requests to the TIS to acquire new traffic events, they can simultaneously upload the speed of their current road segments and report any real-time traffic events. Consequently

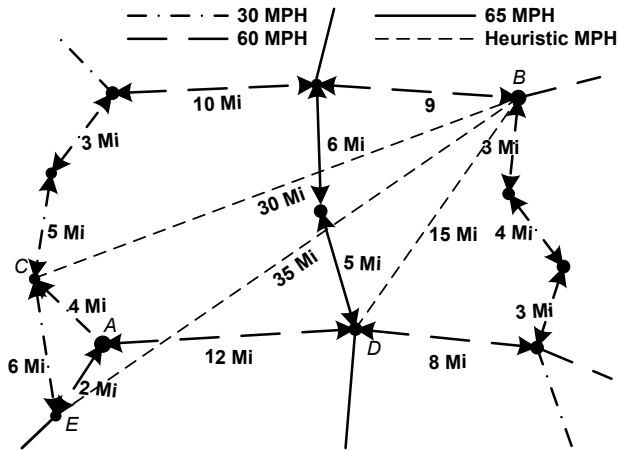


Fig. 5a. A road network with  $B$  as the nearest neighbor of POI of  $A$ .

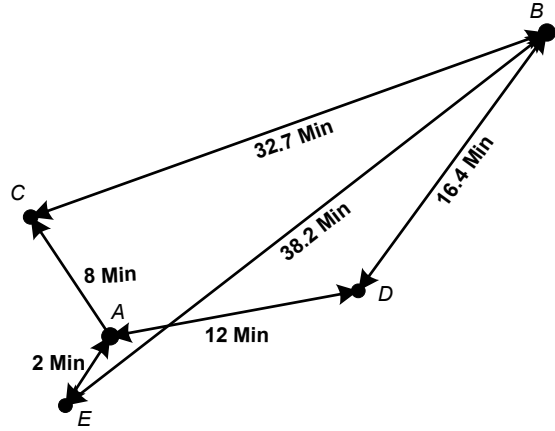


Fig. 5b. The corresponding TTN used in LANN.

Figure 5: An example of traffic time networks in LANN.

the TIS aggregates traffic congestions, accidents, and other real-time traffic events. In addition, transportation and law enforcement agencies can report road construction and accident information to the TIS. Commercial businesses (e.g., gas stations) can also report closure events to the TIS.

Every traffic event is time-stamped and a mobile host can synchronize traffic information with the TIS by checking the latest time-stamp in its local memory.

### 3.6 Global-based Adaptive Nearest Neighbor Queries

The purpose of the LANN algorithm is to ensure that a mobile host can make efficient local navigation decisions at the end of each road segment and always stays on the shortest path to a pre-selected nearest neighbor. Because of the communication range limitation of IEEE 802.11 wireless networks, a mobile host cannot access (assume one hop broadcast) real-time traffic events which happened faraway. However, these events could have significant influences on its driving time. For example, a POI may close unexpectedly and it may be too late for a mobile host to be aware of the event until it reaches the vicinity of the POI. In addition, if a mobile host holds the knowledge of global traffic events, it can update the current NN adaptively when receiving related traffic events. For example, if a road segment on the path to the current nearest neighbor is temporarily closed, this traffic event changes the TTN and may result in another POI being selected as the new nearest neighbor. Therefore, utilizing the cellular communication device on mobile hosts to access current traffic events from the traffic information server is desirable. The TIS access frequency can be decided by each mobile host and all the traffic events related to the current route can be accessed at one time.

We propose a Global-based Adaptive Nearest Neighbor query algorithm that computes the nearest neighbor in a best-first manner with a global travel time network. At the start of a trip a mobile host  $M$  executes the GANN algorithm to compute a nearest POI as the destination  $D$  and the shortest path to  $D$  as the selected route  $S_{route}$ . Afterwards the mobile host follows  $S_{route}$  for traveling to  $D$  before updating traffic events with the server. When  $M$  receives new traffic information  $T_{info}$  from TIS, it needs to determine if  $T_{info}$  has any influence (e.g., traffic jams usually slow down the

traffic on road segments) on  $S_{route}$ . If  $T_{info}$  has no influence on the current route of  $M$ , the mobile host only needs to integrate  $T_{info}$  into its local TTN for future usage. However, if  $T_{info}$  is related to the current journey of  $M$ , the mobile host has to execute more methods. As discussed in Section 3.5, there are four traffic event categories. With category 1 and 2, mobile hosts have to update their local TTN (remove the edge of the closed road segment for category 2) and recalculate the  $Dr_{time}$  from the current location to  $D$ . Then it launches a travel time network NN query with  $Dr_{time}$  as the upper search bound  $S_{bound}$ . Afterwards, GANN chooses the shortest driving time POI within  $S_{bound}$  as the new destination and navigates the mobile host there, if any new NN has been found. In category 3, a selected POI can be closed unexpectedly after a mobile host starts its trip. When receiving a POI closure event which is the current destination, a mobile host has to launch a travel time network NN query for finding a new nearest POI. With category 4, if the recovery is about a traffic congestion and related with the route to the current nearest neighbor, the mobile host only needs to update its local TTN. Otherwise, the mobile client launches a travel time network NN query with the current driving time  $Dr_{time}$  as the search upper bound  $S_{bound}$  on the updated TTN.

The complete algorithm of GANN is formalized in Algorithm 2 and its symbols are listed in Table 1.

Parameter	Description
$S_{route}$	The pre-selected route to the destination $D$
$T_{info}$	Receiving real-time traffic information from TIS
$Dr_{time}$	The driving time from the current location of the query mobile host $Q$ to the destination $D$
$S_{bound}$	The upper bound of a nearest neighbor search
$D_{new}$	The new selected POI for updating the original destination $D$

Table 1: Symbols of the GANN algorithm.

## 4. EXPERIMENTAL VALIDATION

We implemented our adaptive nearest neighbor query algorithms in a simulator to evaluate the performance of our approach. Our main objective is to increase the accuracy of nearest neighbor queries

---

**Algorithm 2** Global-based Adaptive Nearest Neighbor ( $Q$ )

---

```
1: /*  $Q$  is the query mobile host */
2: Execute a travel time network NN query for retrieving the nearest POI and the shortest path to it.
3: Take the returned nearest POI as the destination,  $D$ , and the shortest path to  $D$  as the current route,  $S_{route}$ 
4: while  $Dist(Q, D) \neq 0$  do
5:   Navigate the mobile host to  $D$  and request traffic events from the server with a frequency  $f$ 
6:   if The received information  $T_{info}$  is related with  $S_{route}$  or is a recovery event then
7:     if  $T_{info}$  is a traffic congestion recovery of  $S_{route}$  then
8:       Update the local time network with  $T_{info}$ 
9:     else
10:      if  $D$  is not closed then
11:        Update the time network with  $T_{info}$  and utilize the current location of  $Q$  for calculating a new driving time  $Dr_{time}$  to  $D$ 
12:        Execute the travel time network NN query with  $Dr_{time}$  as a search upper bound  $S_{bound}$ 
13:        if any closer POI is found then
14:          Pick the closest POI within  $S_{bound}$  as  $D_{new}$ 
15:          Replace  $D$  with  $D_{new}$  and update  $S_{route}$  with the route to  $D_{new}$ 
16:        end if
17:      else
18:        Execute the travel time network NN query with the current location of  $Q$  for finding a new nearest POI  $D_{new}$ .
19:        Replace  $D$  with  $D_{new}$  and update  $S_{route}$  with the route to  $D_{new}$ 
20:      end if
21:    end if
22:  else
23:    Update the time network with  $T_{info}$ 
24:  end if
25: end while
```

---

and decrease the driving time. First, real time traffic information can be easily integrated into the underlying network. Second, the shortest path to a destination (POI) can be generated incrementally (LANN) or dynamically (GANN). Consequently, the focus of our simulation is on quantifying the driving time variations. We have performed our experiments with both real-world and synthetic parameter sets.

## 4.1 Simulator Implementation

Our simulator consists of three main modules, the *navigation module*, the *server module*, and the *baseline module*. The objective of the navigation module is to generate and control the movements and the NN query launch of all mobile hosts. Each mobile host is an independent object which encapsulates all its related parameters (such as the movement velocity  $Move_{Velo}$  and the wireless transmission range  $TR_{Rang}$ ) and decides its movement autonomously. Spatial data (POI) indexing is provided with the well known R-tree algorithm with the quadratic splitting method [6]. All mobile hosts move inside a geographical area, measuring 4 miles by 4 miles. Additionally there are user adjustable parameters for the simulation such as the execution length, the number of mobile hosts and the number of POIs. The server module interacts with mobile hosts which execute the GANN algorithm for disseminating real-time traffic events. In addition, the purpose of the baseline module is

to simulate traditional road network NN query solutions [13] and compute a shortest path  $S_P$  to a NN without considering any related traffic events (e.g., congestions, detours, etc.). Then the driving time of following  $S_P$  can be used to compare with both the driving time of utilizing the LANN and the GANN algorithms. Table 2 lists all the simulation parameters.

Parameter	Description
$POI_{Numb}$	The number of points of interest in the system
$Mobi_{Host}$	The number of mobile hosts in the simulation area
$Move_{Velo}$	The mobile host movement velocity (MPH)
$\lambda_{Cong}$	The mean number of congestions per hour
$\lambda_{Deto}$	The mean number of detours per hour
$\lambda_{Clos}$	The mean number of POI closures per hour
$TR_{Rang}$	The transmission range of queries
$Time_{Exec}$	The length of a simulation run
$Exp_{Reqn}$	The measure of the simulation region

**Table 2: Parameters for the simulation environment.**

The simulation is initialized by randomly choosing a starting location for each MH within the simulation area. Since mobile hosts are not always searching for nearest neighbors, we assume each mobile host has two modes, *NN search mode* and *driving mode*. When a mobile host  $M$  is in the NN search mode, the navigation module navigates  $M$  to its queried nearest neighbor  $n$  via employing both the LANN and the GANN algorithms (for comparison purposes). At the same time, the baseline module computes the shortest path  $S_P$  from the start location of  $M$  to  $n$  and estimates the driving time. Afterwards, the comparison module memorizes the driving time of utilizing the LANN algorithm, the GANN algorithm, and following the pre-computed shortest path  $S_P$ . Furthermore, we employ the *random waypoint model* [2] as the mobility model for the driving mode. A MH which is in the driving mode selects a random destination inside the simulation area and progresses greedily toward it. When reaching that location, the MH pauses for a random interval and decides on a new destination for the next travel period. Both processes (NN search and driving) repeat until the end of the simulation. Users can decide the number of mobile hosts which operate in the NN search mode and the driving mode.

The movement of each MH follows the underlying road network and their travel speed  $s$  is determined by the traffic speed on the corresponding road segment. Figure 9 demonstrates the simulator interface.

### 4.1.1 Simulation Parameter Sets

In order to acquire results that closely correspond to real-world traffic conditions, we obtained our simulation parameters from data sets which report, for example, vehicle density, POI density, and traffic event statistics in the Southern California area. We term the two parameter sets based on real-world statistics the *Los Angeles County parameter set* and the *Riverside County parameter set*.

- **Mobile Hosts:** We collected vehicle statistics of Southern California from the Federal Statistics web site<sup>7</sup>. The data provides the number of registered vehicles in the Los Angeles and Riverside Counties (5,498,554 and 944,645, respectively). In our simulations we assume that about 10% of these vehicles are on the road during non-peak hours according to the traffic information from Caltrans<sup>8</sup>. We further obtained the land area of each county to compute the average vehicle density per square mile.

<sup>7</sup><http://www.fedstats.gov/>

<sup>8</sup><http://www.dot.ca.gov/hq/traffops/saferestr/trafdata/>

- **Points of Interest:** We obtained information about the density of interest objects (e.g., gas stations, restaurants, hospitals, etc.) in Southern California from two online sites: GasPriceWatch.com<sup>9</sup> and CNN/Money<sup>10</sup>. Because gas stations are commonly the target of NN queries, we use them as the point of interest type for our simulations.
- **Traffic Events:** According to the National Transportation Statistics<sup>11</sup> and data from traffic related agencies (e.g., Caltrans Office of Traffic Safety, SIGALERT.com real-time traffic information, etc.), we acquired the traffic event statistics of Southern California and categorized these events into two main categories: traffic congestions (e.g., traffic hazards, traffic collisions, etc.) and detours (e.g., road constructions, road closures, etc.). Because there is no official survey of POI closure events, we assume a low occurrence rate. The event generation module is designed to plug-in the collected traffic statistic data and produce four types of traffic/POI events: (1) traffic congestions, (2) detours, (3) POI closures, and (4) event recoveries. We categorized traffic congestions into two levels: medium and heavy. The corresponding speed are 10 to 20 miles per hour (mph) and 0 to 10 mph respectively on local routes; and additionally 20 to 40 mph and 0 to 20 mph on highways. The ratio between medium and heavy congestions is also based on the traffic statistic data. The appearance ratio of these events are based on statistical traffic data and the interval between events is based on the Poisson distribution.

The Los Angeles and the Riverside County parameter sets represent a very dense, urban area and a low-density, more rural area. Hence, for comparison purposes we blended the two real parameter sets together to generate a third, synthetic parameter set. The synthetic data set demonstrates vehicle density, POI density, and traffic event frequency in-between Los Angeles County and Riverside County, representing a suburban area. Table 3 lists the three parameter sets.

Parameter	Los Angeles County	Riverside County	Synthetic Suburbia	Units
$POI_{Numb}$	65	21	43	
$Mobi_{Host}$	1852	204	1028	
$\lambda_{Cong}$	39	11	25	$hr^{-1}$
$\lambda_{Deto}$	7	3	5	$day^{-1}$
$\lambda_{Clos}$	3	2	3	$day^{-1}$
$TR_{Rang}$	200	200	200	m
$Time_{Exec}$	10	10	10	hrs
$Expe_{Regn}$	16	16	16	$mile^2$

**Table 3: The simulation parameter sets for the Los Angeles County, the Riverside County, and the synthetic suburbia.**

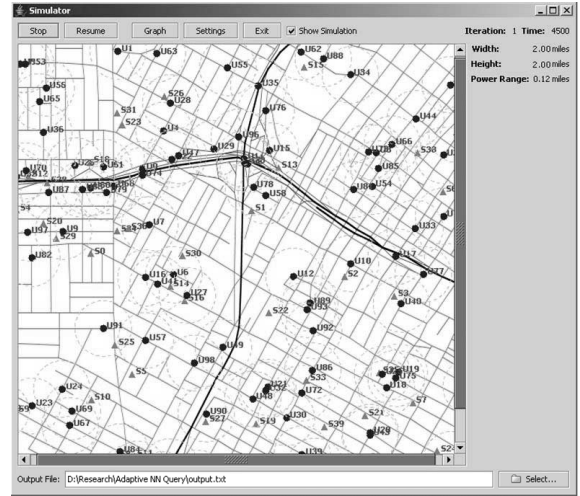
## 4.2 Implementation of Travel Time Network

We acquired our road network data from the TIGER/LINE street vector data available from the U.S. Census Bureau. The road segments belong to several different categories, such as primary highways, secondary and connecting roads, and rural roads. The segments associated with a different road classes are associated with

<sup>9</sup><http://www.gaspricewatch.com>

<sup>10</sup><http://money.cnn.com/>

<sup>11</sup>[http://www.bts.gov/publications/national\\_transportation\\_statistics/](http://www.bts.gov/publications/national_transportation_statistics/)



**Figure 9: The simulator and its visualization interface.**

different maximum driving speeds. We define a road segment as the road section between two crossroads and the driving time of a road segment can be derived from its speed limit, current traffic events on it, and its length. During the execution of a simulation, each mobile host monitors the speed on the road that it is currently traveling on and adjusts its velocity accordingly. One of the challenges when integrating road segments into a complete travel time network is to isolate intersecting paths and determine if they are indeed intersections. For example, freeways generally project many intersections in two-dimensional space, however many of them are over-passes or bridges. Our solution is to detect intersection points with the help of their endpoint coordinates. In addition, differing road classes let us distinguish over-passes from intersections.

## 4.3 Experiments

We used all three parameter sets, Los Angeles County, Riverside County, and synthetic, to simulate our two adaptive nearest neighbor query algorithms. We varied the following parameters to observe their effects on the average driving time: the wireless transmission range, the number of congestions, and the number of detours. The performance metric of the simulation is the *Driving Time Savings Rate* (DTSR) which normalizes the saved driving time to the driving time of the pre-computed  $S_P$ . Since the LANN algorithm queries for one nearest POI as the destination at the beginning and continuously searching for local optimal paths to the destination while the GANN algorithm updates its destination POI if a POI with a shorter travel time is found, we record the driving time to the present destination POI of GANN for comparing with LANN. The primary differences between the three parameter sets are vehicle density, POI density, and traffic event amount. Therefore, the simulation results reveal the applicability of our algorithms to different geographical areas. All simulation results were recorded after the system reached a steady state.

### 4.3.1 Effect of the Transmission Range

In our first experiment we varied the mobile host wireless transmission range from 20 meters to 200 meters, with all other parameters unchanged. We chose 200 meters as a practical upper limit on the transmission range of the IEEE 802.11 technology. Because of obstacles such as buildings, this range could diminish to 100 meters or less in urban areas. Figure 6 illustrates the percentage of the driving time which is saved by employing the LANN algorithm

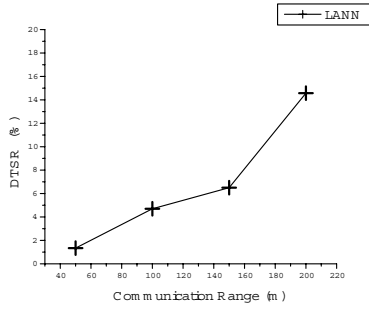


Fig. 6a. Los Angeles County.

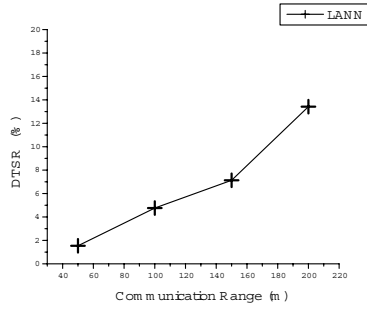


Fig. 6b. Synthetic Suburbia.

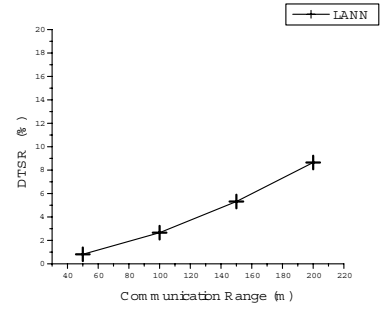


Fig. 6c. Riverside County.

**Figure 6: The percentage of driving time that are saved by the LANN algorithm as a function of the mobile host transmission range.**

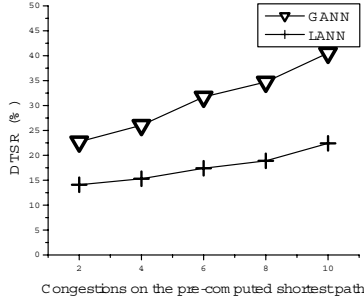


Fig. 7a. Los Angeles County.

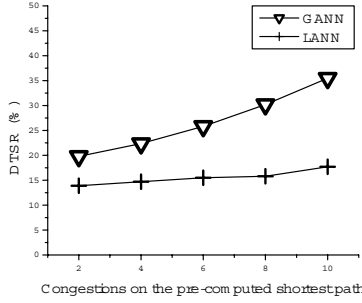


Fig. 7b. Synthetic Suburbia.

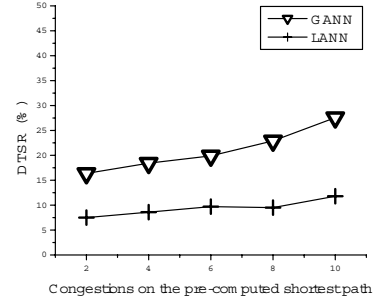


Fig. 7c. Riverside County.

**Figure 7: The percentage of driving time that are saved by the LANN and the GANN algorithm as a function of congestions on the pre-computed shortest path.**

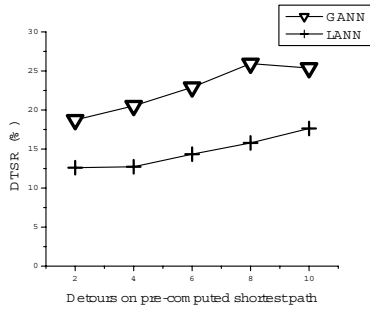


Fig. 8a. Los Angeles County.

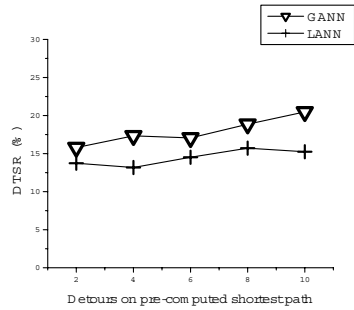


Fig. 8b. Synthetic Suburbia.

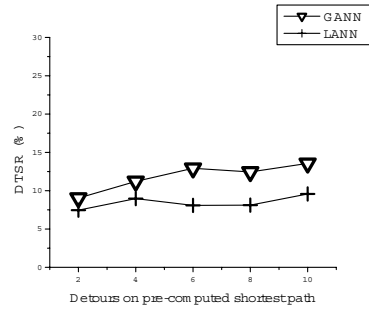


Fig. 8c. Riverside County.

**Figure 8: The percentage of driving time that are saved by the LANN and the GANN algorithm as a function of detour on the pre-computed shortest path.**

compared with following the pre-computed  $S_P$ . As the transmission range extends, a mobile host can reach more peers and more related traffic events can be retrieved. As expected, the effect is most pronounced in Los Angeles County, because of its high vehicle density. At a transmission range of 200 m, the LANN algorithm can save approximately 10% to 15% driving time compared with the pre-computed shortest path solution.

#### 4.3.2 Effect of Congestions Frequency

Since congestions have a significant impact on driving time, we studied the effect of varying the congestions frequency on the pre-computed shortest path  $S_P$  by changing the number from 2 to 10

and the results are shown in Figure 7. We observe that when the congestions frequency increases, our algorithms have a better performance in areas with a higher vehicle density. This is because more traffic information can be detected by peer vehicles. In addition, the advantage of the travel time network becomes more pronounced when congestions on  $S_P$  increase.

#### 4.3.3 Effect of Detours Frequency

Next we varied the detours frequency from 2 to 10 on  $S_P$ . Figure 8 illustrates the simulation results of the three parameter sets. As we can observe, both our algorithms have a better performance than the pre-computed shortest path solution based on driving time.

However since a mobile host can avoid a detour by taking nearby routes, all the result DTSR rates are lower than in the previous experiments.

We conclude from all the performed experiments that our algorithms have better performance among all the experimental cases. We note that the mobile host density has a considerable influence on the driving time savings rate.

## 5. CONCLUSIONS AND FUTURE WORK

Geographic information systems are getting increasingly sophisticated and finding nearest neighbor objects represents a significant class of queries. Existing algorithms work on realistic, but static, spatial networks. The next generation of applications will require real-time information to be integrated to produce search results that reflect the most current network conditions. To this end we have presented the concept of a travel time network that is dynamically and continuously updated. Additionally, we have introduced two nearest neighbor query algorithms that operate on such travel time networks. We have shown through simulation results that our techniques outperform the static approaches and reduce the travel time when dynamic events occur.

## 6. REFERENCES

- [1] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R\*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD Conference*, pages 322–331, 1990.
- [2] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4<sup>th</sup> ACM/IEEE MobiCom*, pages 85–97, 1998.
- [3] Albert M.K. Cheng and Koushik Rajan. A Digital Map/GPS Based Routing and Addressing Scheme for Wireless Ad-hoc Networks. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV'03)*, 2003.
- [4] Zhiming Ding and Ralf Hartmut Güting. Modeling temporally variable transportation networks. In *Database Systems for Advances Applications, 9th International Conference, DASFAA 2004, Jeju Island, Korea, Proceedings*, pages 154–168, 2004.
- [5] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 419–429, 1994.
- [6] Antomn Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, Massachusetts, June 18-21, 1984.
- [7] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.
- [8] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Integrated query processing strategies for spatial path queries. In *ICDE*, pages 477–486, 1997.
- [9] Mohammad R. Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada*, pages 840–851, 2004.
- [10] Duke Lee, Roberto Attias, Anuj Puri, Raja Sengupta, Stavros Tripakis, and Pravin Varaiya. A wireless token ring protocol for intelligent transportation systems. In *Proceedings of the 4<sup>th</sup> International IEEE Conference on Intelligent Transportation Systems*, 2001.
- [11] Jun Luo and Jean-Pierre Hubaux. A Survey of Inter-Vehicle Communication. Technical Report IC/2004/24, School of Computer and Communication Sciences, EPFL, 2004.
- [12] Jörg Ott and Dirk Kutscher. Drive-thru internet: Ieee 802.11b for "automobile" users. In *INFOCOM*, 2004.
- [13] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query Processing in Spatial Network Databases. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 790–801, 2003.
- [14] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest Neighbor Queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, San Jose, CA, May 22-25, 1995.
- [15] Thomas Seidl and Hans-Peter Kriegel. Optimal Multi-step k-Nearest Neighbor Search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 154–165, 1998.
- [16] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *VLDB*, pages 507–518, 1987.
- [17] Cyrus Shahabi, Mohammad R. Kolahdouzan, and Mehdi Sharifzadeh. A Road Network Embedding Technique for k-Nearest Neighbor Search in Moving Object Databases. In *Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems*, pages 94–100, McLean, Virginia, November 2002.
- [18] Shashi Shekhar and Jin Soung Yoo. Processing in-route nearest neighbor queries: a comparison of alternative approaches. In *Proceedings of the Eleventh ACM International Symposium on Advances in Geographic Information Systems, New Orleans, Louisiana, USA*, pages 9–16, 2003.
- [19] Jing Tian, Lu Han, and Kurt Rothermel. Spatially Aware Packet Routing for Mobile Ad Hoc Inter-Vehicle Radio Networks. In *Proceedings of the IEEE Intelligent Transportation System Conference (ITSC'03)*, 2003.