Road Network-Aware Spatial Alarms

Kisung Lee, Ling Liu, Balaji Palanisamy, and Emre Yigitoglu

Abstract—Road network-aware spatial alarms extend the concept of time-based alarms to spatial dimension and remind us when we travel on spatially constrained road networks and enter some predefined locations of interest in the future. This paper argues that road network-aware spatial alarms need to be processed by taking into account spatial constraints on road networks and mobility patterns of mobile subscribers. We show that the Euclidian distance-based spatial alarm processing techniques tend to incur high client energy consumption due to unnecessarily frequent client wakeups. We design and develop a road network-aware spatial alarm processing system, called ROADALARM, with three unique features. First, we introduce the concept of road network-based spatial alarms using road network distance measures. Instead of using a rectangular region, a road network-aware spatial alarm is a star-like subgraph with an alarm target as the center of the star and border points as the scope of the alarm region. Second, we describe a baseline approach for spatial alarm processing by exploiting two types of filters. We use subscription filter and Euclidean lower bound filter to reduce the amount of shortest path computations required in both computing alarm hibernation time and performing alarm checks at the server. Last but not the least, we develop a suite of optimization techniques using motion-aware filters, which enable us to further increase the hibernation time of mobile clients and reduce the frequency of wakeups and alarm checks, while ensuring high accuracy of spatial alarm processing. Our experimental results show that the road network-aware spatial alarm processing significantly outperforms existing Euclidean space-based approaches, in terms of both the number of wakeups and the hibernation time at mobile clients and the number of alarm checks at the server.

Index Terms—Spatial alarms, road networks, motion behavior, optimization, scalability.

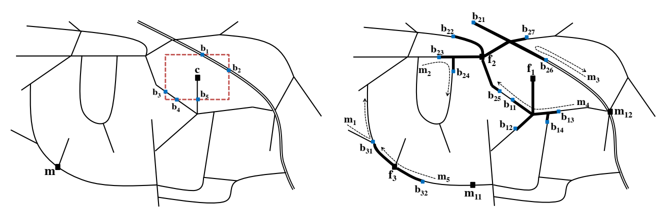
1 INTRODUCTION

MANY of us use time-based alarms daily to remind us the arrival of some predefined time points of interest in the future, such as getting up in the morning and attending an important business meeting. Spatial alarms extend the concept of time-based alarms to spatial dimension and remind us when we enter some predefined locations of interest in the future. An example of spatial alarms is "alert me when I am within 2 miles of the dry clean store in Buckhead". Spatial alarms are basic building blocks for many location-based services, such as location-based advertisements, factory danger zone alert system, and sex offender monitoring system. Since the number of smart devices including smart-phones and tablets is rising steeply , scalable processing of spatial alarms is becoming increasingly important in mobile applications and location-aware computing.

Characterization of spatial alarms. A spatial alarm is defined by four components: a focal point representing the alarm target, a spatial distance representing the alarm region, an owner (or a publisher) of the alarm and a set of alarm subscribers. Spatial alarms are categorized into three groups by their ownership: *private*, *shared* and *public*. A *private* alarm has only one subscriber who is also the publisher of the alarm. A *shared* alarm has a publisher and several subscribers approved by the publisher. In terms of a *public* alarm, its publisher does not set any restriction on subscribers and thus anyone can be a subscriber of the alarm. Public alarms are typically classified by alarm interests, such as traffic alerts and coupons from grocery stores and restaurants. Spatial alarms can also be categorized by the motion behavior of their subscribers and their monitoring targets: *moving objects with static targets*, *static objects with moving targets* and *moving objects with moving targets*. Typical examples of spatial alarms having *moving objects with static targets* are "alert me when I am within 5 miles of a Whole Foods Market in Buckhead" (private) and "notify anyone entering I85 North from Spaghetti Jct. in Atlanta" (public). "The Macy's store at Lenox Square sends advertisements to its customers who are within 10 miles of its store location" (i.e., Macy's customers are spatial alarm targets for the Macy's store and the store will be notified when its customers are within a specified spatial range from the store) is an example of spatial alarms having *static objects with moving targets*. "Alert Lucy when her car is 1 mile apart from her friends' vehicles on the way to Walt Disney World in Orlando" is an example of *moving objects with moving targets*.

Challenges of spatial alarm processing. Spatial alarms are essential for many location-based services, ranging from location-based advertisement to location-based personal reminders. Negligent management of spatial alarms can lead to excessive energy consumption of mobile devices, especially when spatial alarm processing relies on continuous tracking of mobile devices, which is known to be prohibitively expensive. We argue in this paper that intelligent techniques can be developed for scalable processing of spatial alarms by minimizing the amount of continuous monitoring of mobile users locations. Furthermore, the performance of spatial alarm processing can be affected by a number of factors, such as *frequency of wakeups* – how often mobile devices should wake up because of possible alarm hits and *frequency of alarm checks* – how many spatial alarms should be evaluated at each wakeup. Since frequent and

- *K. Lee, L. Liu and E. Yigitoglu are with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, 30332.*
- *B. Palanisamy is with University of Pittsburgh.*



(a) Rectangular Spatial Alarm (b) Road Network Spatial Alarms

Fig. 1: Spatial alarms

possibly unnecessary wakeups and alarm checks not only reduce battery life of mobile devices considerably but also increase the loads of a spatial alarm processing server, we need efficient spatial alarm processing that can reduce the number of unnecessary wakeups and alarm checks at each wakeup. Finally, the spatial alarm processing system should scale to a large number of spatial alarms and mobile users while meeting the *high accuracy* goal by minimizing the alarm miss rate.

Existing approaches on spatial alarm processing can be categorized into two groups by their criteria for controlling the frequency of wakeups: *time-based* approaches (such as periodic wakeups) and *distance-based* approaches (such as safe period [1], [2] and safe region [3], [4]). Most existing techniques define spatial alarms using Euclidean distances and thus a rectangular region is typically used to represent the spatial alarm region. Fig. 1(a) shows an example rectangular alarm, which has five intersecting points with the underlying road network. Such an alarm can be triggered even though its subscribers' current location is far away from the alarm target c based on road network distance. For example, if a mobile subscriber of the alarm is located on b_1, the alarm should be triggered because the subscriber is within the rectangular alarm region. However, even though b_1 is the nearest intersecting point to the alarm target c based on the Euclidean distance, its road network location is far away from the alarm target c based on the road network distance and thus the subscriber can save its battery energy by sleeping for a longer time. This example illustrates the problem of using Euclidean distance to define spatial alarms and highlights the potential benefit of road network-aware spatial alarm processing. To the best of our knowledge, no existing research has taken into account spatial constraints on road networks in optimizing spatial alarm evaluation.

Contributions and organization of the paper. In this paper, we present RoadAlarm – a road network-aware spatial alarm processing system. By taking into account spatial constraints on road networks and mobility patterns of mobile subscribers, RoadAlarm can reduce the frequency of wakeups and increase hibernation time of mobile clients and, at the same time, minimize the computation cost of alarm checks by filtering out those spatial alarms that are irrelevant or far away from the current location of their mobile subscribers. Concretely, we define road network-aware spatial alarms using network distances (e.g., segment length-based or travel time-based). Instead of using a rectangular region, a road network-aware spatial alarm is defined as a star-like subgraph with an alarm target as the center of the star. We define the scope of an alarm region by the set of border points of the star. In addition, we formulate our baseline approach to road network-aware spatial alarm processing by exploiting subscription filtering and Euclidean lower bound filtering. The former can filter out those spatial alarms that are clearly irrelevant by considering only subscribed spatial alarms. The latter can reduce the number of the network distance computations without loss of accuracy. Furthermore, we develop a suite of motion-aware filters as optimization techniques to further reduce the frequency of wakeups as well as the frequency of alarm checks while ensuring high accuracy by considering mobility patterns of mobile subscribers. To the best of our knowledge, RoadAlarm is the first systematic approach to exploring road network-aware and motion-aware filters to reduce the search space and computation cost of road network-aware alarm processing. Our experimental results show that RoadAlarm outperforms existing Euclidean distance-based techniques and can scale to a large and growing number of spatial alarms as well as mobile subscribers.

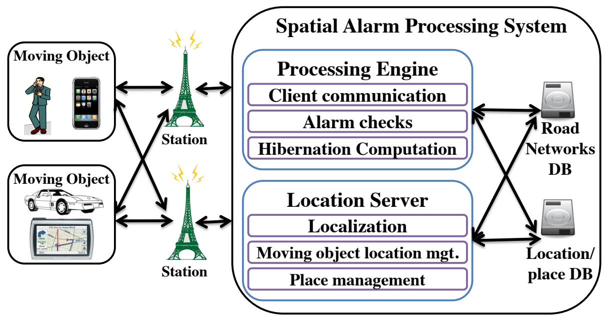


Fig. 2: System architecture

The rest of the paper is organized as follows: We give an overview in Section 2 and present the baseline approach in Section 3. We improve the performance of the baseline approach by developing a suite of optimization techniques in Section 4. We evaluate the performance of RoadAlarm in Section 5, outline the related work in Section 6, and conclude the paper in Section 7.

2 Overview

In this section, we describe the system architecture of RoadAlarm and define road network-aware spatial alarms, alarm miss and hibernation time. A spatial alarm system typically consists of a spatial alarm processing engine and a location server where the locations of moving objects (mobile clients) and the locations of static objects (such as gas stations, restaurants, and so on) are managed. The spatial alarm processing engine communicates with the location server to obtain the current road network locations of mobile subscribers as well as the road network locations of alarm targets for all alarms maintained in its database. The location server uses localization techniques (such as GPS, WiFi or any hybrid localization technology) to keep track of the current positions of moving objects. Fig. 2 presents a sketch of the ROADALARM system architecture.

We assume that moving objects can be any devices (e.g., smart-phones, tablets, navigation systems) with any localization technology such as GPS and WiFi localization. ROADALARM adopts the client-server architecture for spatial alarm processing. Concretely, mobile objects may install (publish) their spatial alarms at the location server as private, shared or pubic alarms. In addition to their own private alarms, mobile objects can subscribe to any public alarms and a subset of shared alarms authorized by other alarm owners. Mobile objects need to install the thin client of ROADALARM as a mobile application on their devices. Each mobile subscriber will obtain an initial hibernation time at the commit of her alarm subscription. Upon the expiration of its old hibernation time, the mobile client will automatically contact the alarm server on behalf of the mobile subscriber to obtain its new hibernation time. We assume that the mobile clients are able to communicate with the server through wireless data channel. During the hibernation time, the ROADALARM application is hibernated at the mobile client, and the alarm server consumers zero alarm processing cost for this mobile client.

Road network model. A road network is represented by a directed graph $G = (\mathcal{V}, \mathcal{E})$, composed of the road junction nodes $\mathcal{V} = \{n_0, n_1, \ldots, n_N\}$ and directed edges $\mathcal{E} = \{n_i n_j | n_i, n_j \in \mathcal{V}\}$. We refer to an edge $n_i n_j$ as a road segment connecting the two road junction nodes n_i and n_j with direction from n_i to n_j. When a road segment is bidirectional, we use edge $n_i n_j$ and edge $n_j n_i$ to denote the two directions of the same road segment. For each road segment, road-related information can be maintained, such as segment length (e.g., 1.2 miles), speed limit (e.g., 55 mph), current traffic data (e.g., average speed is 35 mph), and direction (e.g., one-way road). The length and speed limit of a road segment $n_i n_j$ are denoted by $seglength(n_i n_j)$ in miles and $speedlimit(n_i n_j)$ in miles per hour respectively. Other road-related information such as direction and current traffic data, if available, can be easily incorporated to provide more accurate travel time.

Let n_1 and n_2 denote two road junction nodes and $n_1 n_2 \notin \mathcal{E}$. We define a path from n_1 to n_2 as a sequence of road segment edges, one connected to another, denoted as $n_1 n_{i_1}$, $n_{i_1} n_{i_2}, \ldots, n_{i_{k-1}} n_{i_k}$, $n_{i_k} n_2$ ($k > 0$). The length of a path h between n_1 and n_2, denoted by $pathlength(h)$, is computed as $seglength(n_1 n_{i_1}) + seglength(n_{i_k} n_2) + \sum_{\alpha=1}^{k-1} seglength(n_{i_\alpha} n_{i_{\alpha+1}})$. Given two road junctions n_1 and n_2, since there can be more than one path from n_1 to n_2, we use $PathSet(n_1, n_2)$ to denote the set of all paths from n_1 to n_2. We define a segment length-based shortest path from n_1 to n_2, denoted by $sl\_shortestpath(n_1, n_2)$, as $\{h_{sl} | pathlength(h_{sl}) = \min_{h \in PathSet(n_1,n_2)} pathlength(h)\}$. The travel time of a road segment $n_i n_j$ is defined as $\frac{seglength(n_i n_j)}{speedlimit(n_i n_j)}$ and thus the travel time of a path h, denoted by $traveltime(h)$, is calculated as $\frac{seglength(n_1 n_{i_1})}{speedlimit(n_1 n_{i_1})} + \frac{seglength(n_{i_k} n_2)}{speedlimit(n_{i_k} n_2)} + \sum_{\alpha=1}^{k-1} \frac{seglength(n_{i_\alpha} n_{i_{\alpha+1}})}{speedlimit(n_{i_\alpha} n_{i_{\alpha+1}})}$. The travel time-based shortest path from n_1 to n_2, denoted by $tt\_shortestpath(n_1, n_2)$, is defined as $\{h_{tt} | traveltime(h_{tt}) = \min_{h \in PathSet(n_1,n_2)} traveltime(h)\}$.

A *road network location*, denoted by $L = (n_i n_j, p)$, is a tuple of two elements: a road segment $n_i n_j$ and the *progress* p along the segment from n_i to n_j. The road network distance between two road network locations $L_1 = (n_{i_1} n_{i_2}, p_1)$ and $L_2 = (n_{j_1} n_{j_2}, p_2)$ is the length of the shortest path between L_1 and L_2 in terms of either segment length or travel time. The **segment length-based road network distance**, denoted by $sldistance(L_1, L_2)$, and **travel time-based road network distance**, denoted by $ttdistance(L_1, L_2)$, are formally defined respectively as follows:

$$\begin{aligned} sldistance(L_1, L_2) = {} & seglength(n_{i_1} n_{i_2}) - p_1 + p_2 \\ & + pathlength(sl\_shortestpath(n_{i_2}, n_{j_1})) \end{aligned}$$

$$ttdistance(L_1, L_2) = \frac{seglength(n_{i_1} n_{i_2}) - p_1}{speedlimit(n_{i_1} n_{i_2})} + \frac{p_2}{speedlimit(n_{j_1} n_{j_2})} + traveltime(tt\_shortestpath(n_{i_2}, n_{j_1}))$$

Even though the segment length-based distance is the most commonly used distance measure on road networks, it may not provide sufficient and accurate distance information in terms of actual travel time from the current location to the destination, considering that highway road segments are usually much longer but also with much higher speed limits and thus may have relatively shorter travel time compared to some local road segments. To ensure high accuracy and high performance of spatial alarm processing, we use the travel time-based distance as default road network distance measure in ROADALARM.

Road Network-Aware Spatial Alarms. In ROADALARM, we define a road network-aware spatial alarm as a star-shaped subgraph centered at the alarm focal point, denoted as $SA_{RN}(p_f, r, S)$ where p_f is the alarm target or the alarm focal point (a road network location), r is the alarm monitoring region, represented by a spatial range (segment length or travel time) from p_f, and S is a set of subscribers. Consider Fig. 1(b) that shows three star-shaped alarms with focal points f_1, f_2 and f_3. The road network-aware spatial alarm with focal point f_1 has a range of 5 miles based on the segment length. The road network-aware spatial alarm with focal point f_2 has a range of 10 minutes based on the travel time. We call those points on the road network which bound a star-shaped spatial alarm *border points*. For example, b_{12} is one of the four border points of the alarm with focal point f_1.

Alarm Miss and Hibernation Time. We define an *alarm miss* as a case when a spatial alarm is not triggered as it should be even though a mobile subscriber of the alarm enters or passes through the alarm region. Consider Fig. 1(b): moving objects m_1, m_2, m_3 and m_5 should each receive a spatial alarm alert and m_4 should get two alerts from the spatial alarms with alarm targets f_1 and f_2 sequentially. If any one of those alarms is not triggered during the course of travel for the five subscribers, the alarm miss has happened. Opposite to alarm miss, an *alarm hit* refers to the case when a spatial alarm is triggered when one of its mobile subscribers enters the alarm region.

As mentioned earlier, in ROADALARM we compute a hibernation time for each mobile subscriber, during which the mobile subscriber hibernates the ROADALARM thin client on her device. We define a *hibernation time* for each moving object, which is a time interval during which the moving object does not need to wake up and the alarm server does not need to perform alarm checks for this mobile subscriber. A hibernation time of a moving object is specified by a time interval consisting of its start time and end time. If the current time is between them, the moving object's current status is *hibernation*; otherwise, it is *alive*. Upon expiration of the current hibernation time, the mobile client wakes up, and communicates with the spatial alarm server to obtain a new hibernation time. The alarm server examines the current location of the mobile subscriber and the set of alarms subscribed by this subscriber to determine the new hibernation time. If the new hibernation time is smaller than a system-defined threshold δ, a spatial alarm is triggered and the mobile subscriber is notified. Otherwise, a new hibernation time will be sent to the mobile subscriber.

We would like to note that the timeliness of alarm triggering is also important, especially when spatial alarms are defined with some quality of service (QoS) guarantee. For example, it is possible that based on the current hibernation time for the moving object m_5 in Fig. 1(b), m_5 may receive the spatial alarm alert when it approaches the focal point f_3 or just before leaving the spatial alarm through the border point b_{31}. Thus, in ROADALARM we use a stronger definition of alarm miss: if a moving object's current status is *hibernation* when it enters alarm region of a spatial alarm by crossing a border point of the alarm, we treat it as an alarm miss.

The *alarm success rate* is the percentage of spatial alarm alerts which are not missed, and is defined as follows:

$$\textit{alarm success rate} = 1 - \frac{\textit{Total number of alarm misses}}{\textit{Total number of actual alarm hits}}$$

For example, if there are 9 alarm hits and 1 alarm miss (actual hit but not triggered), the success rate is 90%.

3 SPATIAL ALARM PROCESSING

In this section we present the design consideration of the ROADALARM baseline algorithm for efficient processing of spatial alarms. We first describe the Euclidean distance-based approach and the conventional network expansion-based approach to process road network-aware spatial alarms and analyze the problems with these two approaches. Then we introduce the baseline algorithm for ROADALARM by incorporating subscription filter and Euclidean lower bound filter.

3.1 Euclidean Distance-based Approach

The Euclidean distance-based approach is often considered as the most intuitive baseline approach to implementing spatial alarm processing. Concretely, for every mobile object m, upon the expiration of its hibernation time, m wakes up and contacts the spatial alarm server to obtain its new hibernation time. The alarm server first retrieves the index of all spatial alarms and obtains the set of border points for each active spatial alarm. Then the alarm server computes the Euclidean distance between the current location of m and each of the border points for all spatial alarms and selects the border point that is the nearest to the current location of m, denoted by $b_{nearest}$, and calculates the new hibernation time for m based on the Euclidean distance and a velocity metric that is representative, such as the global maximum speed (V_{max}) or the expected speed of m ($V_{expected}$). For example, if there are 35 mph, 55 mph and 65 mph road segments on the road network, the global maximum speed is 65 mph. Although using the global maximum speed is too pessimistic to calculate the hibernation time, it ensures high alarm success rate. The end time of the new hibernation time for m based on this Euclidean distance-based method using the global maximum speed is defined as $\textit{current time} + \frac{eucdistance(m, b_{nearest})}{V_{max}}$ where $eucdistance(m, b_{nearest})$ is the Euclidean distance between m and $b_{nearest}$. The object m will be in the hibernation status during the above hibernation time.

In general, the Euclidean distance-based method using the global maximum speed is the most conservative technique since not all mobile objects are traveling at the maximum speed. Thus an alternative metric we adopted in the first prototype of ROADALARM is the expected speed calculated using the current location of m, the previous location, the previous expected speed and the previous maximum speed [2]. The end time of the new hibernation time for m based on this Euclidean distance-based method using the expected speed is defined as $\textit{current time} + \frac{eucdistance(m, b_{nearest})}{V_{expected}(m)}$ where $V_{expected}(m)$ is the expected speed of m.

Although the Euclidean distance-based approach is simple to implement, it suffers from a number of fatal weaknesses. First, the hibernation time is computed using the Euclidean distance rather than road network distance, thus the hibernation time is unnecessarily short. Consequently, mobile objects need to wake up frequently, making ROADALARM consuming higher battery energy than necessary. Second, for each mobile object m, the nearest spatial alarm may not be subscribed by m, thus the hibernation time computed using the Euclidean distance to the nearest spatial alarm is misleading. This is especially true when all the spatial alarms subscribed by m is far away from the current location of m.

3.2 Network Expansion-based Approach

Another intuitive baseline approach to evaluating road network-aware spatial alarms is to use Dijkstra's network expansion algorithm [5]. We present two methods using different road network distances: one is using the segment length-based distance (*NE-S*) and the other is using the travel time-based distance (*NE-T*).

When a moving object m wakes up, *NE-S* and *NE-T* first retrieve a set of spatial alarms (A_m) subscribed by m. For each spatial alarm $a_i \in A_m$, the set of border points of a_i are obtained. *NE-S* and *NE-T* take the current location of m and each border point of a_i as input to calculate the segment length-based shortest path and the travel time-based shortest path respectively, using Dijkstra's network expansion algorithm. After computing the shortest path from m's current location to every border point of all alarms in A_m, *NE-S* selects the shortest path with the smallest segment length-based distance, denoted by p_{sl}, to compute the new hibernation time for m. Similarly, *NE-T* chooses the shortest path having the smallest travel time-based distance, denoted by p_{tt}, to compute the new hibernation time for m. Thus, we can compute the end time of the hibernation time for m based on *NE-S* and *NE-T* as follows:

$$HT_{NE-S}(m) = \textit{ current time} + \textit{traveltime}(p_{sl})$$
$$HT_{NE-T}(m) = \textit{ current time} + \textit{traveltime}(p_{tt})$$

Recall that $\textit{traveltime}(p)$ computes the travel time of a path p as described in Section 2.

The network expansion-based approach is simple and straightforward to implement. However, the computation cost of this approach is extremely high since it examines all border points of every spatial alarm subscribed by a mobile object m at each wakeup. The shortest path computation cost to calculate the hibernation time of m increases rapidly as the number of alarms subscribed by the mobile object m increases and most of the subscribed alarms are far away from the current location of m. This is because the computation cost of Dijkstra's network expansion algorithm primarily depends on the size of underlying road network, the distance between the source location and the destination location, and the number of shortest path computations to be performed. If the destination is far away from the source, it is highly costly to compute the shortest path using the Dijkstra's network expansion algorithm because it exhaustively expands too many unnecessary nodes and edges.

3.3 RoadAlarm Baseline Approach

Bearing in mind the problems with the Euclidean distance-based approach and network expansion-based approach, we design the baseline approach of RoadAlarm by introducing two simple and yet effective filters. We use the subscription filter to scope the computation of the hibernation time for each mobile object to only those alarms that are subscribed by the mobile object. In addition, we use *Euclidean lower bound (ELB)* as another type of filter to minimize the number of shortest path computations required to compute the hibernation time upon each wakeup by filtering out some irrelevant border points of subscribed spatial alarms. The concept of *Euclidean lower bound* refers to the fact that the segment length-based shortest path distance between two network locations L_1 and L_2 is at least equal to or longer than the Euclidean distance between L_1 and L_2. By combining the subscription filter and ELB filter, the RoadAlarm baseline approach (BA) can minimize the number of shortest path computations required for computing hibernation time for each mobile subscriber while maintaining the accuracy of alarm evaluation. We present two methods using the segment length-based and the travel time-based road network distance, denoted by *BA-S* and *BA-T* respectively.

Concretely, instead of computing shortest paths from the current location L_m of the mobile subscriber m to every border point of all alarms subscribed by m, *BA-S* computes the new hibernation time of m in five steps: *First*, for every alarm subscribed by m, denoted by $a_i \in A_m$, we find the border point that has the shortest distance from L_m. Instead of computing shortest paths from L_m to every border point of alarm a_i, we compute the Euclidean distance between L_m and every border point of a_i and sort the set of border points based on their Euclidean distances from L_m in an ascending order using the Incremental Euclidean Restriction (IER) algorithm [6], [7], [8]. *Second*, let b_{nn} denote the border point that has the smallest Euclidean distance from L_m. We compute the shortest path from L_m to b_{nn} using the segment length-based distance. *Third*, we use a binary search algorithm to examine the sorted list of border points and remove those border points whose Euclidean distance from L_m is bigger than *sldistance*(L_m,b_{nn}). *Fourth*, for each remaining border point b_j, *BA-S* computes the shortest path from L_m to b_j. If *sldistance*(L_m,b_j) $<$ *sldistance*(L_m,b_{nn}) holds, we assign b_j to be b_{nn}. Thus, for a given mobile object and an alarm $a_i \in A_m$, the nearest border point b_{nn} of a_i will be used as the reference border point of a_i to compute the hibernation time for m. *Finally*, *BA-S* examines every alarm $a_i \in A_m$ and its nearest border point b_{nn} and chooses the border point whose segment length-based distance from L_m is the smallest. Let b_{min} denote this nearest border point and p_{min} denote the shortest path from L_m to b_{min}. Thus we compute the end time of the new hibernation time for m as $\textit{current time} + \textit{traveltime}(p_{min})$.

Now we illustrate the working of the baseline approach using the example in Fig. 1(b). We have three spatial alarms a_1, a_2, a_3 with focal points f_1, f_2, f_3 respectively and two moving objects m_{11} and m_{12}. Let us assume that m_{11} subscribes to a_1 and a_3 and m_{12} subscribes to a_1 and a_2. Let $L_{m_{11}}$ and $L_{m_{12}}$ denote the current location of m_{11} and m_{12} respectively. When m_{11} and m_{12} wake up upon the expiration of their hibernation time, without the subscription filter and ELB filter, we will need to compute the shortest paths from $L_{m_{11}}$ and $L_{m_{12}}$ to all 13 border points and then choose the nearest border point which has the shortest network distance (either segment length-based or travel time-based) from $L_{m_{11}}$ and $L_{m_{12}}$ respectively. With the subscription filter, we can filter out alarm a_2 for m_{11} and alarm a_3 for m_{12} when computing the new hibernation time. By the ELB filter, to find the new hibernation time for m_{12}, we only need to perform one shortest path computation from $L_{m_{12}}$ to b_{13}. This is because by Euclidean distance, b_{13} is the nearest border point of a_1 from $L_{m_{12}}$ and b_{26} is the nearest border point of a_2 from $L_{m_{12}}$. Given that *eucdistance*($L_{m_{12}}$,b_{13}) $<$ *eucdistance*($L_{m_{12}}$,b_{26}), b_{13} is the nearest border point for m_{12}. Now we compute the network distance (either segment length-based or travel time-based) from $L_{m_{12}}$ to b_{13}, denoted by *sldistance*($L_{m_{12}}$,b_{13}). By comparing *sldistance*($L_{m_{12}}$,b_{13}) with the Euclidean distance from $L_{m_{12}}$ to all other border points of a_1 and a_2, we find that the following condition *eucdistance*($L_{m_{12}}$,b_k) $>$ *sldistance*($L_{m_{12}}$,b_{13}) holds ($k = 11, 12, 14, 21, 22, 23, 24, 25, 26, 27$). Thus the ELB filter effectively removes the unnecessary shortest path computations when computing the new hibernation time for m_{12}.

We below show that the network location of the mobile object has significant impact on the effectiveness of the ELB filter. Consider the mobile object m_{11} and the two alarms a_1 and a_3 subscribed by m_{11} in Fig. 1(b). For the alarm a_3, we do not need to compute the shortest path from $L_{m_{11}}$ to the border point b_{31} since the Euclidean distance between $L_{m_{11}}$ and b_{31} is longer than the segment length-based distance from $L_{m_{11}}$ to b_{32}. However, for the alarm a_1, the list of border points sorted in ascending order of their Euclidean distance from L_{m_1} is $\{b_{12}, b_{11}, b_{14}, b_{13}\}$. Clearly, the segment length-based network distance from $L_{m_{11}}$ to its nearest border point b_{12}, denoted by *sldistance*($L_{m_{11}}$, b_{12}), is longer than the Euclidean distance between $L_{m_{11}}$ and each of the last three border points in the list and thus none of the three border points are filtered out for alarm a_1.

BA-T finds the nearest border point of a moving object m using the travel time-based road network distance. For *BA-T*, we cannot directly use the ELB filtering as done in *BA-S* since the Euclidean lower bound property does not hold for the travel time-based distance. For example, when the Euclidean distance and the segment length-based distance between a border point and the current location of a mobile object m are 5 miles and 10 miles respectively, there could be another border point in which the Euclidean distance and the segment length-based distance are 12 miles and 15 miles respectively, but it has shorter travel time-based distance since there is a freeway connecting the border point and the current location of m. Therefore, we extend the ELB filtering for the travel time-based distance. Instead of using only segment lengths, *BA-T* defines the *travel time-based Euclidean lower bound* as the travel time multiplied by the global maximum speed limit on the entire road network. For example, if the travel time from the current location of m to an alarm border point is 1 hour and the global maximum speed limit is 70 mph, the travel time-based ELB in *BA-T* is 70 miles (1h x 70mph). *BA-T* excludes border points whose Euclidean distance is longer than 70 miles since the moving object m cannot get to those border points within 1 hour even if it travels at the global maximum speed. Since *BA-T* is using the global maximum speed limit to calculate the travel time-based ELB, the search space of *BA-T* is usually larger than that of *BA-S*, i.e., *BA-T* considers more border points of alarms subscribed by m to find the nearest one. The remaining steps of *BA-T* are the same as those in *BA-S*. In the first prototype of RoadAlarm we use the global maximum speed limit for travel time-based ELB in order to ensure the high accuracy of alarm evaluation. It would be interesting to use some less conservative speed or motion metrics to see if we can further reduce the search space needed for computing the hibernation time for mobile objects at the cost of a small and affordable accuracy loss.

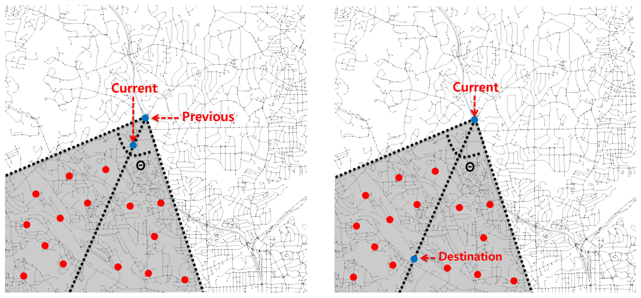
4 Motion-aware Optimizations

Compared to the Euclidean distance-based approach and the conventional network expansion-based approach, the RoadAlarm baseline approach (BA) improves the efficiency of road network-aware alarm processing along two dimensions. First, it uses the subscription filter to narrow down the set of spatial alarms to be considered for computing the hibernation time upon wakeup of each mobile subscriber. Second, it utilizes the ELB filter to cut down the number of border points to be examined for shortest path computation while achieving high alarm success rate.

However, the ELB filter is not always effective. In some cases, the number of border points after applying the ELB filter remains to be high. Recall the case of m_{11} in Fig. 1(b) in which the ELB filter can filter out one border point (b_{31}) for alarm a_3. For alarm a_1, the Euclidean distance from $L_{m_{11}}$ to b_{12} is the shortest and thus the road network-based distance from $L_{m_{11}}$ to b_{12} is first calculated. Because this road network distance is longer than the Euclidean distances from $L_{m_{11}}$ to all the other border points (b_{11}, b_{13}, b_{14}), the ELB filter filters out none of border points for alarm a_1.

In this section we introduce a suite of motion-aware filters to further reduce the search space and the computation time of the RoadAlarm baseline approach (BA), especially for those moving objects which subscribe many spatial alarms and their alarms are scattered in a large geographical area. The main idea of the motion-aware filters comes from the observation that mobile objects traveling on road networks typically exhibit some degree of steady motion. First, a moving object traveling on a spatially constrained road network can move only by following the predefined road segments connected to the current road segment it resides. Thus, its movement cannot be changed drastically. For example, if a moving object is marching on a road segment, its current moving direction cannot be changed until it reaches a road junction. Furthermore, even if it reaches a road junction, it has high probability to follow the road segment in the same or similar direction at the junction node. We refer to such motion behavior as steady motion.

In this section, we present five types of steady motion-based filters. Our first three optimization techniques use steady motion degree Θ to capture the constrained motion characteristics of moving objects traveling on a road network. For each mobile object, its steady motion degree Θ models the direction of its movements along the road network. If a sharp turn occurs at a junction node, a new Θ value will be computed for the mobile object based on the characteristics of underlying road networks and past movement history of the mobile object. We can also view this Θ as a confidence indicator. When a mobile object moves on the road network by following its current direction, a large Θ value indicates possible sharp turns and sudden travel direction changes whereas a small Θ value indicates high probability of steady motion along the current direction. When a mobile object is traveling on the road network with a clear destination in mind, this Θ angle can be determined based on the current location of the mobile object and the destination location.



(a) Current direction-based (b) Destination-based

Fig. 3: Vector-based motion-aware filters

the remaining unfiltered border points, our approach calculates the new hibernation time by executing the ROADALARM baseline approach.

The current direction-based motion-aware filter is good when the hibernation time is relatively short and the time window in which the mobile object moves steadily is relatively low as well, since the current direction vector changes each time when the mobile object wakes up due to the expiration of its current hibernation time. Furthermore, the current direction-based motion-aware filter may be suitable for some mobile clients who do not want to disclose their destination information due to privacy reasons. However, if the destination is given (or can be inferred by using Calendar), then it is more effective to use a destination-based motion filter.

4.1 Current Direction-based Motion-aware Filter

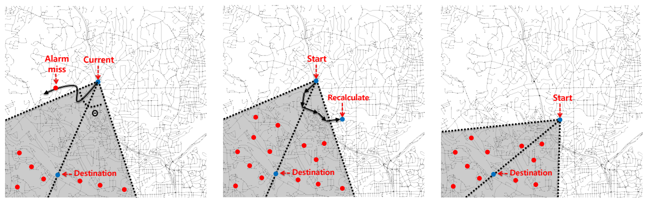
The first motion-aware filter is based on the current direction of moving objects and their steady motion degree Θ. This filter selects only those border points which reside in the Θ region anchored at the current location of the mobile object. The Θ degree is determined based on the current travel direction of the mobile object. One popular way to define the current direction of a moving object is to use the *current direction vector* in which we use the last reported location as the initial point and the current location as the terminal point of the vector. Let (p_1, p_2) and (c_1, c_2) denote the previous location and the current location of a moving object m respectively. The current direction vector of m is defined as $v = <c_1 - p_1, c_2 - p_2>$. Based on this current direction vector, when a mobile object m wakes up, this filter limits the search space using the steady motion degree Θ and selects only those border points of m that reside within this reduced search space, as shown in Fig. 3(a). For example, let (xb_1, xb_2) denote a border point. To check if the border point is within the Θ reduced search space, this filter first defines another vector $w = <xb_1 - p_1, xb_2 - p_2>$ and then calculates the degree of the border point from the current direction vector v using the following equation:

$$sm\_degree(v, w) = \arccos(\frac{v \cdot w}{|v||w|})$$

If $sm\_degree(v, w) > \Theta$, this border point is removed since it is outside the constrained search space. For

4.2 Destination-based Motion-aware Filter

The destination-based motion-aware filter utilizes both the current location and the destination information of moving objects. Destination information can be directly given by the mobile clients, such as those using car navigational systems or can be extracted from mobile clients' calendar applications. In this filter, the degree Θ indicates how confident the moving object will march toward its destination. The destination-based motion-aware filter chooses only border points which reside in the Θ region defined based on the current location of moving objects to their destination. We define a *destination vector* to represent the direction toward the destination, in which the current location and the destination location are used as the initial and terminal point of the vector respectively. Let (c_1, c_2) and (d_1, d_2) denote the current location and the destination of a moving object m respectively. The destination vector of m is defined as $v = <d_1 - c_1, d_2 - c_2>$. When m wakes up, the destination-based motion-aware filter restricts the search space using the destination vector v and Θ and then selects only the border points within this Θ restricted search space, as shown in Fig. 3(b). For example, let (yb_1, yb_2) denote a border point. To check if the border point is within the reduced search space, this filter first computes another vector $w = <yb_1 - c_1, yb_2 - c_2>$ and then calculates the degree of this border point in terms of this new vector and the destination vector v using the equation $sm\_degree(v, w)$. We remove those border points whose



(a) Alarm miss (b) Cache Invalidation (c) New search space

Fig. 4: Caching-based motion-aware filters

$sm\_degree(v, w)$ values are higher than the specific Θ defined by m or calculated by the system in the absence of user-defined Θ. Our approach calculates the nearest border point by examining all the unfiltered border points and then computes the new hibernation time for m by invoking our RoadAlarm baseline algorithm.

4.3 Caching-based Motion-aware Filter

Both the current direction-based motion-aware filter and the destination-based motion-aware filter can reduce the computation cost of finding the nearest border point for each mobile object upon its wakeup. This computation reduction is achieved by reducing the number of candidate border points and thus the search space through a combination of the steady motion degree Θ with current direction or destination information. However, those two filters also suffer from a couple of inherent problems. First, if a mobile object m takes a short detour due to traffic and moves slightly out of the scoped spatial region defined by Θ and the current direction or destination, there could be an alarm miss. Consider Fig. 4(a): a moving object m has moved slightly outside the scoped spatial region and there exists an spatial alarm which resides just outside the scoped region and is very close to the current location of m. Unfortunately, this alarm will be missed if we use the current direction-based motion-aware filter or destination-based motion-aware filter. Another weak point is that those two filters recalculate the search space and thus the set of candidate border points at every wakeup of each moving object. This causes not only unnecessarily frequent and possibly duplicate computation of the candidate border points but also adds some unnecessary susceptibility to small detour-like movements of mobile objects. Concretely, if a moving object changes its direction slightly, for example, due to traffic directed detour, the selection of the candidate border points found at the current wakeup could be very different from the selection at the previous wakeup. Therefore, the two filters may miss some spatial alarms which have high probability to be a hit due to this unnecessarily sensitive susceptibility.

To address this problem, we propose another motion-aware filter, called *caching-based motion-aware filter*, based on the observation that moving objects will move toward their destination constantly and persistently even though they may change their direction opposite to (or deviate quite bit from) the destination for a short period of time. Initially, this filter selects the candidate border points for each moving object based on its current location, its destination location and its Θ using the destination-based motion-aware filter. This filter then stores the selected candidate border points with the calculated destination vector for each moving object. When a moving object m wakes up the next time, instead of recomputing the Θ region and the set of candidate border points as done in the destination-based motion-aware filter, this caching-based motion-aware filter retrieves the stored candidate border points of m and then finds the nearest border point to the current location of m by examining the stored border points using our RoadAlarm baseline approach. Finally, this approach calculates the hibernation time using the nearest border point in the same way as is done in the baseline approach.

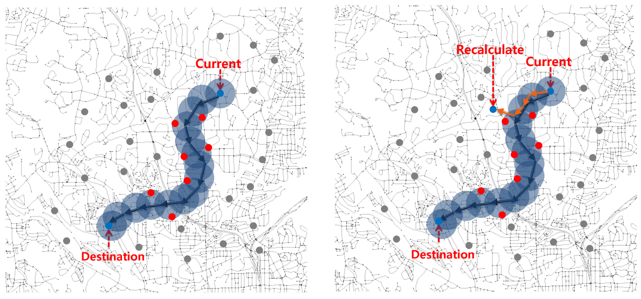
Even though the caching-based filter is proposed to handle the susceptibility to small changes, continuous small changes can make a big change as shown in Fig. 4(b). To address this problem, this filter has a mechanism to check whether the stored border points are obsolete and thus to calculate new candidate border points. When a moving object wakes up, this filter calculates the degree of the moving object's current location from the stored destination vector. If the degree is larger than Θ (i.e., the object went out of the scoped search space), then this filter recalculates the search space based on the object's current location as shown in Fig. 4(c).

4.4 Shortest Path-based Motion-aware Filter

Even though the caching-based filter avoids unnecessarily frequent filtering of border points, it still needs to examine too many border points in order to find the nearest one, especially when Θ is large and many alarms are subscribed by moving objects. Consider Fig. 3(b): the border points on the bottom far left or far right corner are unlikely to be hit by the moving object since it is far away from the object's destination. Motivated by this observation, we propose the shortest path-based motion-aware filter based on a natural assumption that moving objects will follow the shortest path to the destination. Initially, this filter calculates the shortest path (p_{min}) from the current location to the destination for each moving object and then selects some border points within a boundary distance d from the shortest path, as shown in Fig. 5(a). The distance d indicates the level of steadiness: if a moving object follows the calculated shortest path, a small value of d is sufficient. For example, let b denote a border point of the moving object. To check if b is within the boundary distance d from the shortest path p_{min}, this filter calculates the perpendicular distance from the border point to all road segments of p_{min} and then finds the minimum value as follows:

$$min_{b,p_{min}} = min_{n_{p_i}n_{p_{i+1}} \in p_{min}} pdistance(b, n_{p_i}n_{p_{i+1}})$$

where $n_{p_i}n_{p_{i+1}}$ is a constituent road segment of the path p_{min} and $pdistance(b, n_{p_i}n_{p_{i+1}})$ is the perpendicular distance from border point b to road segment $n_{p_i}n_{p_{i+1}}$. If $min_{b,p_{min}}$ is less than d, the border point is selected as a



(a) Selection of border points (b) Shortest path recalculation

Fig. 5: Shortest path-based motion-aware filter

candidate border point of the moving object. The shortest path-based motion-aware filter then stores the selected candidate border points with the calculated shortest path for each moving object. When a moving object m wakes up, this filter retrieves the stored candidate border points of m and then finds the nearest border point, among the retrieved border points, using the ROADALARM baseline algorithm. Finally, this approach calculates the hibernation time using the nearest border point in the same way as is done in the baseline approach.

Like the caching-based filter, the shortest path-based filter also has a mechanism to handle moving objects which go out of our reduced search space based on the shortest path as show in Fig. 5(b). When a moving object wakes up, this filter calculates the distance from the stored shortest path of the object and if the distance is larger than d, recalculates the search space based on the object's current shortest path to the destination.

4.5 Selective Expansion-based Motion-aware Filter

The shortest path-based motion-aware filter selects border points which have high probability to be hit based on the shortest path from the current location of moving objects to their destination. Even though it reduces the computation time to calculate the hibernation time by considering fewer (but more relevant) border points compared to the ROADALARM baseline approach and the other steady motion-based approaches, it still needs at least two shortest path computations: one for calculating the shortest path from the current location to the destination to select candidate border points and the other for choosing the nearest border point among the selected border points. These computations will increase the server loads when the destination is far away from the current location of a moving object and there is no nearby spatial alarm from the current location. To reduce the computation cost, we propose a selective expansion-based motion-aware filter in which an exact shortest path computation is not needed. The selective expansion-based filter expands only road segments which have high probability to be passed by a moving object. To select target road segments to be expanded, we utilize the destination of moving objects and apply the concept of Simulated Annealing (*SA*) to the expansion. *SA* probabilistically finds a good approximation to the global optimal solution in a large solution space by giving high randomness in early stages and almost no randomness in ending stages. Using this basic concept of *SA*, the selective expansion-based filter expands most of road segments in early steps even though some of them have opposite direction to the destination. In following steps, this filter incrementally strengthens the condition of the expansion and thus only road segments whose direction points toward the destination are expanded. This expansion is terminated if it satisfies one of three cases: 1) the expansion arrives at the destination, 2) the expansion meets any spatial alarm of the moving object and 3) there is no more road segment which satisfies the condition of the expansion. Since this filter expands only relevant road segments which have high probability to be hit from the current location to the destination and it terminates the expansion process even though there is no found border point (case 3), it considerably reduces the computation cost to calculate the hibernation time compared to other processing methods which require shortest path computations. In addition to the reduced computation cost, since it expands most of road segments in early steps, the selective expansion-based filter can cover common cases in which moving objects move in the opposite direction from the destination to take faster roads such as freeways.

The algorithm of the selective expansion-based filter is formally defined as follows. Like other processing methods we propose, this filter starts when a moving object m wakes up. Let L_m and d_m denote a current location and a current destination of m respectively. We define $T(i)$ which denotes a time-varying parameter at step i and $E(n_p, i)$ which denotes an energy of an expansion node on a road junction n_p at step i. A smaller energy of a road junction means that the road junction has higher probability to be visited by m compared to other road junctions having a larger energy. A road segment $n_p n_q$ connected to n_p is expanded if a new energy $E(n_q, i+1)$, defined as follows, is less than $T(i)$.

$$E(n_q, i+1) = E(n_p, i) \times dv(n_p n_q, d_m)$$

where $dv(n_p n_q, d_m)$ represents a deviability of $n_p n_q$ from the destination d_m. Road segments whose direction points toward the destination have a small dv value and, on the other hand, road segments in which their direction is opposite to the destination have a large dv value. Therefore, road segments in which their direction points toward the destination will have higher probability to be expanded since their dv value is small. The deviability is defined as follows:

$$dv(n_p n_q, d_m) = \frac{degree(\overrightarrow{n_p n_q}, \overrightarrow{n_p d_m})}{180^\circ} \times \frac{1}{w(speedlimit(n_p n_q))}$$

where $w(speedlimit(n_p n_q))$ is a weight based on the speed limit of $n_p n_q$. By giving more weights to faster roads such as freeways, it makes such faster roads have higher probability to be expanded than slower roads. If $degree(\overrightarrow{n_p n_q}, \overrightarrow{n_p d_m})$ is 0° (i.e. $n_p n_q$ exactly points toward the destination), we use 1° instead of 0° to continue the selective expansion.

T value gradually decreases as steps increase to strengthen the expansion condition and is defined as $T(i) = \frac{T_0}{k^i}$ where k is a parameter that controls the expansion rate and T_0 is an initial value for the expansion. For a large k, the T value becomes smaller rapidly as steps increase and thus more road segments are excluded from the expansion, compared to a small k. A large T_0 value makes more road segments to be expanded. We use 1 as T_0 value to ensure that all road segments are expanded regardless of the k value and their dv value at first step.

The selective expansion-based filter starts with expanding the road junction n_0, where the current location L_m of m is located, with its initial energy 1 ($E(n_0, 0)$). If L_m is located on the road segment $n_p n_q$, this filter treats L_m as a road junction n_0 and $n_0 n_p$ and $n_0 n_q$ as road segments connected to n_0. For each road segment $n_0 n_j$ connected to n_0, this filter calculates $E(n_j, 1)$ using the above formula and then expands $n_0 n_j$ if $E(n_j, 1)$ is less than $T(0)$. While expanding $n_0 n_j$, this filter stops the whole expansion process if there is a border point of m or the destination d_m on $n_0 n_j$. If $n_0 n_j$ is expanded without encountering any border point or d_m, n_j is inserted into the expansion list for the next step with its energy $E(n_j, 1)$. After checking all road segments connected to n_0, this filter moves to the next step and expands road junctions in the expansion list using the above process. This expansion process is terminated if there is no road junction for the next step or any border point or d_m is encountered during the expansion.

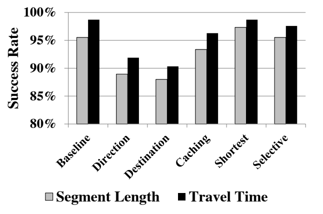
To calculate the hibernation time for m, if a border point or d_m is encountered during the expansion, this filter uses the travel time, from L_m to the encountered border point or d_m, as the hibernation time. Since this filter keeps the accumulated travel time from L_m to each expanded road junction, no additional computation is needed to calculate the hibernation time for m. If the expansion is terminated because there is no more road junction to be expanded, this filter chooses a terminal road junction (i.e., in which no connected road segment is expanded) having the smallest travel time among selected candidate terminal road junctions and then uses the travel time to the terminal road junction as the hibernation time for m. To select the candidate terminal road junctions, we introduce another confidence degree Θ_{SESM}. The selective expansion-based filter checks only terminal road junctions within Θ_{SESM} based on the vector from L_m to d_m and then chooses a terminal road junction having the smallest travel time among the candidate terminal road junctions. If Θ_{SESM} value is too large, it ensures high success rate, but its hibernation time is unnecessarily short because some terminal road junctions which are terminated at earlier steps and thus have short travel time are included in the search space. On the other hand, if Θ_{SESM} value is too small, it cannot ensure high success rate because only terminal road junctions which survived until last steps are included in the search space and thus the selected travel time is too long. Since this filter also keeps and updates the smallest travel time based on Θ_{SESM} during the expansion, no additional computation is needed to calculate the hibernation time.

One disadvantage of the above synchronous (i.e., all target road junctions are expanded at the same step) selective expansion on road networks is that, even though a border point or the destination d_m is encountered during the expansion, the point could be reached by other road segments having shorter travel time at later steps. Furthermore, nearby border points could not be reached during the expansion if there are many short road segments from L_m to the border points. Therefore, spatial alarms can be missed due to the long travel time calculated by ignoring some nearby border points or faster road segments connecting to the border points. To solve this problem, we use an asynchronous version in which a road junction having the smallest segment length ($SESM - S$) or travel time ($SESM - T$) is expanded first, regardless of its current step, using a priority queue.

In summary, ROADALARM provides five motion-aware filters to reduce the search space and the computation time of the ROADALARM baseline approach. The current direction-based and the destination-based filters are based on steady motion degree Θ to take into account the constrained motion characteristics of moving objects traveling on a road network. The caching-based filter extends the destination-based filter for improved accuracy by capturing small detour-like movements of mobile objects. To further reduce the number of border points evaluated by ROADALARM, we develop the shortest path-based filter to select only border points that are close to the shortest path to the destination. Last but not the least, we further reduce the computation cost in ROADALARM by introducing the selective expansion-based filter, which avoids exact shortest path computation by expanding only road segments that have high probability to be passed by a mobile user.

5 EXPERIMENTAL EVALUATION

In this section we evaluate the performance of our ROADALARM methods through four sets of experiments. We use *gt-mobisim* simulator [9] to generate mobility traces on real road networks downloaded from U.S. Geological Survey (USGS [10]). For the first three sets of experiments, the mobility traces are generated on a map of northwest Atlanta, which covers about 11 km (6.8 miles) by 14 km (8.7 miles), using the random trip model [11]. The road networks consist of four different road types: residential roads and freeway interchange with 30 mph speed limit (48 km/h), highway with 55 mph limit (89 km/h) and freeway with 70 mph limit (113 km/h). Ranges of spatial alarms are chosen from a Gaussian distribution with a mean of 50 m and standard deviation of 10 m. We use 50 m as the boundary distance d of the shortest path-based motion-aware filter. For the selective expansion-based motion-aware filter, we


(a) Alarm success rate

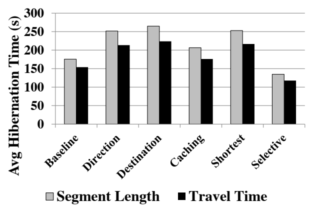

(b) Average Hibernation time

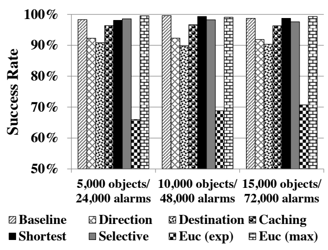
Fig. 6: Segment length-based vs Travel time-based approaches

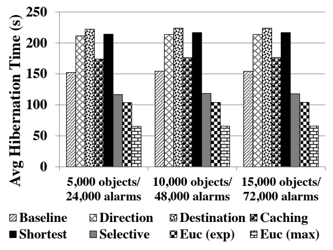
empirically use 4 as the k value and 90° as the Θ_{SESM} value to select not too short and not too long travel time. We give 1, 2 and 3 to 30 mph, 55 mph and 70 mph road segments respectively as their speed weights in order to give faster roads more chances of expansion. All experiments were conducted on a desktop having one Intel i5-2300 quad-core processor, 6GB of RAM, and one 1TB 7200 rpm hard disk.

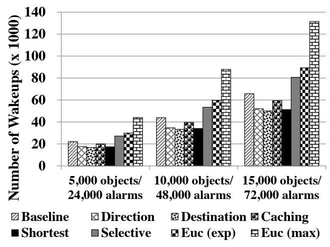
5.1 Comparison with existing methods

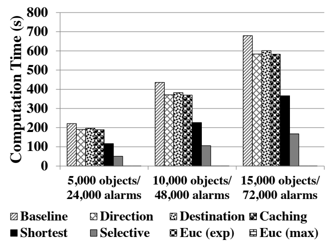
We first compare segment length-based approaches and travel time-based approaches as shown in Fig. 6. These experiments use 15,000 objects (and about 72,000 spatial alarms) and 180° as the Θ value of the current direction-based, destination-based and caching-based motion-aware filters. Each object has different number of spatial alarms, given by Zipf distribution with five alarms as the most common value (i.e., rank 1). We exclude the results of network expansion-based methods since they cannot scale to 15,000 moving objects. The alarm success rate for travel time-based approaches is higher than the corresponding segment length-based approaches as shown in Fig. 6(a). This is primarily because segment length-based approaches select the segment length-based shortest path in which spatial alarms can be missed if moving objects follow paths having shorter travel time. On the other hand, the average hibernation time of each travel time-based approach is shorter than that of its corresponding segment length-based approach since the travel time on the segment length-based shortest path is always equal to or longer than that on the travel time-based shortest path for the same source and destination location. Without loss of generality, in the rest of the experiments, we include the results of only travel time-based approaches for simplicity.

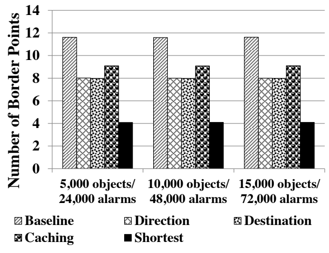
The first set of experiments compares our approaches with existing Euclidean space-based methods in Fig. 7. This set of experiments uses a moving object population with size ranging from 5,000 to 15,000 and each object has different number of spatial alarms, given by Zipf distribution with five alarms as the most common value.
Alarm success rate. The success rates for different approaches are shown in Fig. 7(a). The shortest path-based and selective expansion-based motion-aware filters have almost the same success rate as the Euclidean distance-based approach using the global maximum speed and the ROADALARM baseline approach. The caching-based filter has more than 5% better success rate than the destination-based filter. This confirms our assumption that moving objects will move toward their destination constantly even though they may change their direction opposite to the destination for a short time. The Euclidean distance-based approach using the expected speed has the lowest success rate since it fails to consider spatial constraints of moving objects.


(a) Alarm success rate


(b) Hibernation time per wakeup


(c) Number of Wakeups


(d) Total Computation Time


(e) Number of Border Points

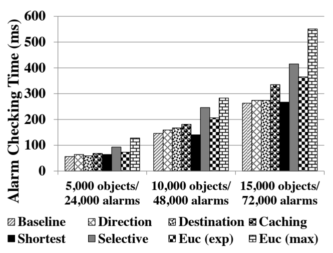
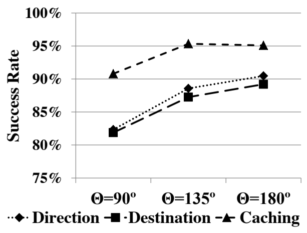
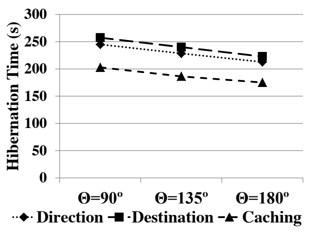

(f) Total Alarm Checking Time

Fig. 7: Comparison with Euclidean space-based approaches

Hibernation time. Fig. 7(b) shows the average hibernation time of moving objects. The longer the hibernation time is, the more energy the mobile clients can conserve. The hibernation time of the shortest path-based filter is three times longer than that of the Euclidean distance-based approach using the global maximum speed and 40% longer than that of the ROADALARM baseline approach. This result also shows that the shortest path-based filter ensures high success rate in the same way as the Euclidean distance-based approach and the ROADALARM baseline approach even though moving objects of the shortest path-based filter can conserve much more energy. The selective expansion-based filter has 45% and 25% shorter hibernation time than the shortest path-based filter and the ROADALARM baseline approach respectively since it calculates the hibernation time even though there is no found border point. It, however, still has 80% longer hibernation time than the Euclidean distance-based approach using the global maximum speed. The Euclidean distance-based approach using the global maximum speed has the shortest hibernation time



(a) Alarm success rate



(b) Hibernation time

Fig. 8: Effect of the steady motion degree Θ

since it pessimistically utilizes the Euclidean distance and the global maximum speed to calculate the hibernation time.

The number of wakeups. Fig. 7(c) shows that the number of wakeups is inversely related to the hibernation time. The smaller number of wakeups indicates the lower server loads since the server computes the hibernation time whenever a moving object wakes up.

Computation time. Fig. 7(d) shows the total computation time to calculate the hibernation time. The shortest path-based filter has 45% faster computation time than the ROADALARM baseline approach since it has smaller number of wakeups of moving objects. The selective expansion-based filter has the smallest computation time among the road network-based approaches since it does not need shortest path computations to calculate the hibernation time. Its computation time is just 24% and 45% of that of the ROADALARM baseline approach and the shortest path-based filter respectively. The Euclidean distance-based approaches need only a little computation time since the computation cost to calculate the Euclidean distance is negligible, compared to the road network distance calculation, even though the number of wakeups is more as shown in Fig. 7(c).

The number of border points. Even though there is only about 20% difference between the shortest path-based filter and the ROADALARM baseline approach in terms of the number of wakeups, there is about 45% difference in terms of the computation time. Furthermore, even though all motion-aware filters have similar number of wakeups, only the shortest path-based filter has better computation cost than the others. This is because the shortest path-based filter considers the smallest number of border points to calculate the hibernation time, as shown in Fig. 7(e).

Alarm checking time. Fig. 7(f) shows the total processing time to check whether moving objects hit any alarms. We use a hash map to store spatial alarms. The result confirms that our approach checks spatial alarms efficiently.

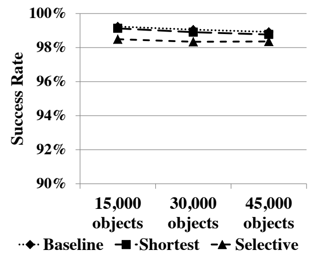
5.2 Effect of the steady motion degree

We investigate the effect of different settings of the steady motion degree Θ on success rate and hibernation time with 15,000 moving objects and about 72,000 spatial alarms. The results are shown in Fig. 8 with Θ values set to 90°, 135° and 180°. The success rate for the current direction-based, destination-based and caching-based filters increases as Θ values increase, because more border points are selected as shown in Fig. 8(a). Fig. 8(b) shows that the average hibernation time decreases with growing Θ values. This is because border points having shorter travel time are newly selected to calculate the hibernation time as the search space increases.

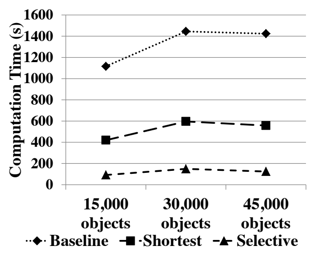
5.3 Effect of growing number of objects and alarms

Fig. 9(a) and Fig. 9(b) evaluate the scalability of our approaches by increasing the number of moving objects. Total 300,000 spatial alarms are deployed for this set of experiments and the number of moving objects increases from 15,000 to 45,000. Each object has zero to 30 spatial alarms, given by Zipf distribution with 15 alarms as the most common value, and all spatial alarms are private. We think this setting deploying 45,000 moving objects is realistic on this road network of northwest Atlanta, in which the total length of all road segments is 1384 km (865 miles), since there is a mobile user every 31 m (10 feet) on average. We include the measurement results of only the ROADALARM baseline approach, the shortest path-based filter and the selective expansion-based filter as they have high success rate compared to other methods. Fig. 9(a) confirms that our approaches ensure the high success rate with growing number of moving objects. The selective expansion-based filter has slightly lower success rate than the others since it does not try to find the nearest border point if there is no nearby border point. In terms of the total computation time, there is no increase from 30,000 to 45,000 objects since with fixed alarms, many objects have no spatial alarms as shown in Fig. 9(b). The selective expansion-based filter's computation time is only 23% and 9% of that of the shortest path-based filter and the ROADALARM baseline approach respectively while ensuring similar success rate.

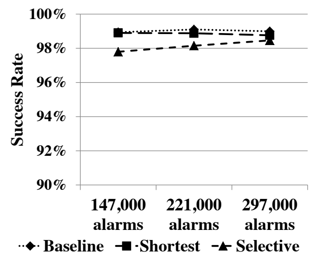
Fig. 9(c) and Fig. 9(d) show the scalability of our approaches by increasing the number of spatial alarms with 15,000 moving objects. We increase the most common value of Zipf distribution from 10 to 20 and thus the total number of alarms grows from about 147,000 to 297,000. Fig. 9(c) verifies that our approaches ensure the high success rate with increasing number of spatial alarms. Note that the success rate of the selective expansion-based filter increases as the number of spatial alarms grows. This is because, if a moving object has more spatial alarms, there is a higher probability that a border point of the object is found during the selective expansion. Fig. 9(d) shows that the computation time of the shortest path-based filter increases only slightly with the growing number of spatial alarms. This is primarily because the shortest path-based filter selects only border points having high probability to be hit and thus the increased number of spatial alarms has no huge impact on the selected border points by the shortest path-based filter. Even though the computation time of the ROADALARM baseline approach has more



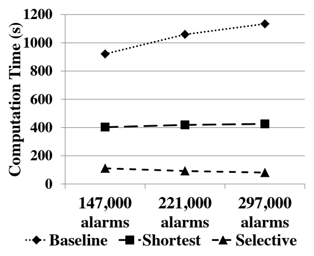
(a) With growing #objects



(b) With growing #objects

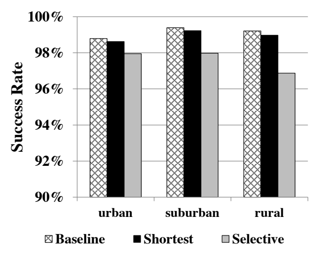


(c) With growing #alarms

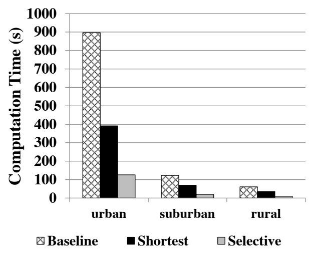


(d) With growing #alarms

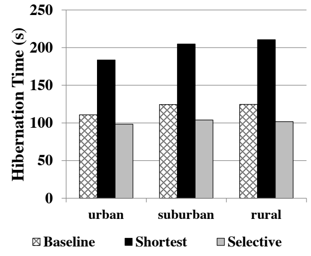
Fig. 9: Effect of growing number of objects and alarms



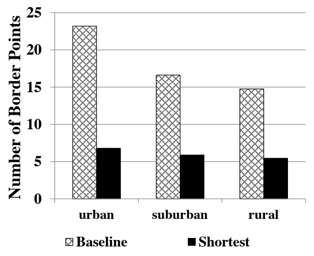
(a) Alarm success rate



(b) Total Computation Time



(c) Average Hibernation Time



(d) Number of Border Points

Fig. 10: Comparison of urban, suburban and rural performance

increase than that of the shortest path-based filter, it does not increase linearly with the growing number of spatial alarms. The computation time of the selective expansion-based filter even decreases as the number of alarms increases because the selective expansion of the filter is terminated earlier with the fewer number of expanded road segments due to the higher probability that a border point is found.

5.4 Effect of different road networks

This set of experiments measures the performance of our approaches using different types of road networks. In addition to the map of northwest Atlanta as an urban road network, we choose the map of Duluth, GA and the map of Helen, GA as a suburban and a rural road network respectively. All three road networks cover almost same size (i.e., 6.8 miles by 8.7 miles) but have totally different number of road segments and road junctions. The total numbers of road segments of the urban, suburban and rural road network are 9,187 (average length is 150.7 m), 1,600 (258.3 m) and 765 (356.3 m) respectively. The total numbers of road junctions are 6,979, 1,486 and 711 for the urban, suburban and rural road network respectively. The urban road network has 431 and 681 road segments having 70 mph and 55 mph speed limit respectively. The other road segments have 30 mph speed limit. 24 and 218 road segments of the suburban road network have 70 mph and 55 mph as their speed limit respectively. The rural road network has 27 and 66 road segments having 70 mph and 55 mph speed limit respectively.

Fig. 10 shows the experimental results for the three different road networks. This set of experiments uses 15,000 moving objects and total about 72,000 spatial alarms. Each object has different number of spatial alarms, given by Zipf distribution with five alarms as the most common value. Fig. 10(a) confirms that our approaches ensure the high success rate for different types of road network. The selective expansion-based filter has slightly lower success rate on the rural road network since moving objects are more likely to use road segments whose direction does not point toward the destination and thus not expanded, due to the limited number of road segments. On the rural road network, the computation times is about 8% of that on the urban road network as shown in Fig. 10(b). This is primarily because of the high complexity (i.e., 12 time more road segments and 10 times more road junctions than the rural road network) of the urban road network. Fig. 10(c) shows that moving objects on the suburban and rural road networks have longer hibernation time than those on the urban road network even though the number of spatial alarms for each moving object and the focal point and the range of each spatial alarm are given by same distributions for all three road networks. Since the urban road network has 12 times more segments and 16 times more segments having 70 mph speed limit compared to the rural road network, it has more probability to have a path having shorter travel time between two locations. As shown in Fig. 10(d), less border points are considered to calculate the hibernation time on the suburban and rural road networks than on the urban road network because spatial alarms on complex road networks usually have more border points.

In summary, our experimental results show that the shortest path-based motion-aware filter outperforms the rest in most cases since this approach ensures high success rate while reducing the computation cost of servers and conserving energy of mobile clients. Since the selective expansion-based filter considerably reduces the computation cost while ensuring high success rate, it is suitable for spatial alarm processing servers which should compute the hibernation time quickly for a huge number of moving objects while ensuring longer hibernation time than the Euclidean space-based approach to save the energy of moving objects. For those applications in which high success rate is required, both the ROADALARM baseline approach and the the shortest path-based motion-aware filter are good options. Especially, for some applications in which the battery power of mobile clients is not a serious problem, the ROADALARM baseline algorithm may be a better choice since it has a slightly higher success rate than the shortest path-based motion-aware filter. The current direction-based filter, the destination-based filter and the caching-based filter are appropriate for those applications in which reducing the computation cost of servers and the battery consumption of mobile clients are top priorities while maintaining the acceptable success rate (about 90%).

6 RELATED WORK

There are many existing studies on continuous spatial queries to find objects within a predefined range or k nearest objects from a query center point [12], [13], [14], [15], [16], [17]. Some of them are based on road networks [18], [19] or land surface [20]. However, spatial alarms are fundamentally different from continuous spatial queries in terms of their purposes as well as target applications. Continuous queries such as "find 3 nearest Starbucks stores while driving to Miami" require continuous query evaluation as I am driving on the US highway. On the other hand, spatial alarms have a predefined location of interest, such as "alert me when I am within 5 miles of the public library in Buckhead" and thus require alarm evaluation only when subscribers are in the vicinity of the spatial alarms. Even when the mobile subscribers are moving on the road, their spatial alarms may not need to be evaluated if those alarm targets are located far away from the current locations of their subscribers. This is the fundamental reason why spatial alarms deserve to be processed more efficiently using a different set of algorithms and optimizations.

Existing research on spatial alarms and location reminders mainly focus on the Euclidean space. [1] proposes an approach to process spatial alarms in the Euclidean space by combining spatial indexes such as R-tree and Voronoi Diagram with the maximum speed-based safe period. [3] develops a safe region-based approach for spatial alarm processing in the Euclidean space. Different shapes of safe regions are proposed and compared in [3]. [4] pointed out the high cost of safe region-based approach and proposed the Mondrian tree index that can index both spatial alarms and alarm free regions within a uniformed framework.

7 CONCLUSION

We have presented ROADALARM – an efficient and scalable approach to processing road network-aware spatial alarms. By utilizing spatial constraints on road networks and mobility patterns of mobile objects, the ROADALARM approach can provide longer hibernation time of mobile clients while ensuring high success rate. To the best of our knowledge, all existing results on spatial alarms are based on the Euclidean space. This paper is the first one that develops efficient algorithms and optimizations for scaling road network-aware spatial alarm processing in ROADALARM, with a preliminary result in [21] and a software demo in [22].

Our research on ROADALARM continues along several directions, including distributed version of ROADALARM, efficient indexing of spatial alarms, and privacy preserving management of spatial alarms.

REFERENCES

[1] B. Bamba, L. Liu, P. S. Yu, G. Zhang, and M. Doo, "Scalable processing of spatial alarms," in *HiPC'08*, 2008.
[2] A. Murugappan and L. Liu, "An energy efficient middleware architecture for processing spatial alarms on mobile clients," *Mob. Netw. Appl.*, vol. 15, pp. 543–561, August 2010.
[3] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu, "Distributed processing of spatial alarms: A safe region-based approach," in *ICDCS'09*.
[4] M. Doo, L. Liu, N. Narasimhan, and V. Vasudevan, "Efficient indexing structure for scalable processing of spatial alarms," in *GIS'10*, 2010.
[5] E. W. Dijkstra, "A note on two problems in connexion with graphs." *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[6] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, no. 2, Jun. 1999.
[7] T. Seidl and H.-P. Kriegel, "Optimal multi-step k-nearest neighbor search," in *SIGMOD'98*, 1998.
[8] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *VLDB'03*, 2003.
[9] "GT-mobisim," 2009, http://code.google.com/p/gt-mobisim/.
[10] "U.S. Geological Survey," 2012, http://www.usgs.gov/.
[11] P. Pesti, L. Liu, B. Bamba, A. Iyengar, and M. Weber, "RoadTrack: scaling location updates for mobile clients on road networks with query awareness," *Proc. VLDB Endow.*, vol. 3, 2010.
[12] Z. Song and N. Roussopoulos, "K-nearest neighbor search for moving query point," in *SSTD '01*, 2001.
[13] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch, "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," *IEEE Trans. Comput.*, vol. 51, no. 10, pp. 1124–1140, Oct. 2002.
[14] Y. Tao, D. Papadias, and Q. Shen, "Continuous nearest neighbor search," in *VLDB'02*, 2002.
[15] G. S. Iwerks, H. Samet, and K. Smith, "Continuous k-nearest neighbor queries for continuously moving points with updates," in *VLDB'03*, 2003.
[16] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *SIGMOD'05*.
[17] X. Yu, K. Q. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving objects," in *ICDE'05*, 2005.
[18] H.-J. Cho and C.-W. Chung, "An efficient and scalable approach to cnn queries in a road network," in *VLDB'05*, 2005.
[19] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis, "Continuous nearest neighbor monitoring in road networks," in *VLDB'06*.
[20] S. Xing, C. Shahabi, and B. Pan, "Continuous monitoring of nearest neighbors on land surface," *Proc. VLDB Endow.*, 2009.
[21] K. Lee, L. Liu, S. Meng, and B. Palanisamy, "Scaling spatial alarm services on road networks," in *ICWS'12*, 2012.
[22] K. Lee, E. Yigitoglu, L. Liu, B. Han, B. Palanisamy, and C. Pu, "Roadalarm: A spatial alarm system on road networks," in *ICDE '13*, 2013.

Kisung Lee is a Ph.D. candidate in the College of Computing at Georgia Tech. He is conducting research in big data management in the Cloud, mobile computing and social network analytics.

Ling Liu is a full Professor in Computer Science at Georgia Institute of Technology. She directs the research programs in Distributed Data Intensive Systems Lab (DiSL). Prof. Liu is an IEEE fellow.

Balaji Palanisamy is an assistant professor in University of Pittsburgh. He received his M.S and Ph.D. degrees in Computer Science from the college of Computing at Georgia Tech in 2009 and 2013 respectively.

Emre Yigitoglu is a Ph.D. student in the College of Computing at Georgia Tech.