CALBA: Capacity-Aware Location-Based Advertising in Temporary Social Networks

Wenjian Xu
Department of Computer Science
City University of Hong Kong
Hong Kong
wenjianxu2@cityu.edu.hk

Chi-Yin Chow
Department of Computer Science
City University of Hong Kong
Hong Kong
chiychow@cityu.edu.hk

Jia-Dong Zhang
Department of Computer Science
City University of Hong Kong
Hong Kong
jzhang26@student.cityu.edu.hk

ABSTRACT

A temporary social network (TSN) is confined to a specific place (e.g., hotel and shopping mall) or activity (e.g., concert and exhibition) in which the TSN service provider allows nearby third party vendors (e.g., restaurants and stores) to advertise their goods or services to its registered users. However, simply broadcasting all the vendors' advertisements to all the users in the TSN may cause the service provider to lose its fans. In this paper, we present Capacity-Aware Location-Based Advertising (CALBA), which is a framework designed for TSNs to select vendors as advertising sources for mobile users. In CALBA we measure the relevance of a vendor to a user by considering their geographical proximity and the user's preferences. Our goal is to maximize the overall relevance of selected vendors for a user with the constraint that the total advertising frequency of the selected vendors should not exceed the user's specified capacity. First, we model the snapshot selection problem as 0-1 knapsack and solve it using an approximation method. Then, CALBA keeps track of the selection result for moving users by employing a safe region technique that can reduce its computational cost. We also propose three pruning rules and a unique access order to effectively prune vendors which could not affect a safe region, in order to improve the efficiency of the client-side computation. We evaluate the performance of CALBA based on a real location-based social network data set crawled from Foursquare. Experimental results show that CALBA outperforms a naïve approach which periodically invokes the snapshot vendor selection.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Design, Performance

SIGSPATIAL'13, November 05 - 08 2013, Orlando, FL, USA

http://dx.doi.org/10.1145/2525314.2525356.

Keywords

Temporary social networks, location-based advertising, location-based services, mobile computing

1. INTRODUCTION

The concept of temporary social networks (TSNs) has been used at hotels, concerts, theme parks, and sports arenas, where people form a social group for a short time period with a common interest or purpose. People confined to a specific place or activity are allowed to join its TSN using their social network accounts (e.g., Foursquare and Facebook). Then they could text the entire group, share pictures or locations, and set up sub-groups, etc. For example, LobbyFriend [13] is a TSN for a hotel to connect its guests and staff, and notify them with events in or around the hotel. When a guest checks out of the hotel, all the traces of her actions in the TSN are erased.

One of the most important functions in TSNs is advertising. For instance, in a hotel TSN, a hotel acts a a service provider to allow nearby third party vendors (e.g., restaurants and tour companies) to advertise their services or goods to its guests. Thus, the hotel can benefit from gaining commissions from vendors as well as building closer relationships with its guests. Advertising on TSNs has advantages over other platforms because (1) TSN users are more willing to contact vendors because of geographical proximity, (2) it is easier for vendors to find potential customers based on users' social network profiles, and (3) TSNs provide opportunities for users to set up new friendships for some special deals (e.g., group purchase or coupon).

However, simply broadcasting all vendors' advertisements to all the users in a TSN may cause the service provider to lose its fans. Even if a vendor paying more money to the service provider can send its advertisements to the TSN users more frequently, users may treat the advertisements as spam. Such simple advertising approaches not only have negative effects on vendors [11], but also waste the network bandwidth and drive users away from the TSN. In reality, users prefer more relevant advertisements with respect to their preferences [20]. Therefore, a successful advertising service on TSNs should fulfill two essential requirements: ($\mathcal{R}1$) capacity requirement and ($\mathcal{R}2$) relevance requirement. The former one requires that the frequency of advertisements sent to users cannot exceed their specified capacity [11] and the latter guarantees the relevance of vendors to individual users.

To this end, we present Capacity-Aware Location-Based

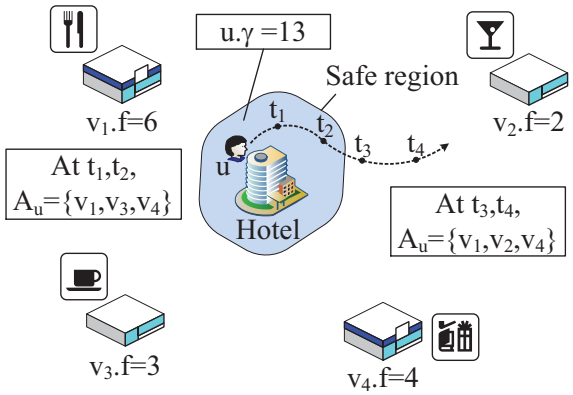


Figure 1: Capacity-aware location-based advertising

Advertising (CALBA), which is a framework designed for TSNs to select vendors as advertising sources for mobile users. In this paper, we take a TSN in a hotel as an example. In the framework, when a user u registers at a hotel, the hotel's sponsored vendors become the *candidate advertising sources* of u, which is denoted as $\mathcal{V}_u$. In the example of Figure 1, $\mathcal{V}_u = \{v_1, v_2, v_3, v_4\}$. CALBA aims at selecting the most relevant subset of vendors $\mathcal{A}_u \subseteq \mathcal{V}_u$ as the *actual advertising sources* for u under the capacity constraint.

First we discuss the capacity requirement ($\mathcal{R}1$). For each vendor $v \in \mathcal{V}_u$, it has an *advertising frequency* $v.f$ obtained according to how much it pays to the service provider (i.e., the hotel). In Figure 1, $v_1.f = 6$, which means the vendor v_1 generates 6 different advertisements for its services or products per hour. We assume that once a candidate vendor v is selected as the actual advertising source for u, the service provider will deliver any advertisement generated by v to u. Since users may not want to read too many advertisements within a certain time period, we enable each user u to set a *tolerance capacity* $u.\gamma$. The selected actual advertising sources $\mathcal{A}_u \subseteq \mathcal{V}_u$ should satisfy that the total advertising frequency of the selected vendors in $\mathcal{A}_u$ cannot exceed $u.\gamma$, i.e., $\sum_{v \in \mathcal{A}_u} v.f \leq u.\gamma$. For example, in Figure 1, since the total advertising frequency of all the vendors (i.e., 15) is larger than u's tolerance capacity (i.e., 13), we cannot select all of them as the actual advertising sources.

For the relevance requirement ($\mathcal{R}2$), we combine non-spatial and spatial factors to measure the relevance of a vendor v to a user u. Especially, we indicate that as the spatial distance between u and v changes, their relevance would change accordingly. Based on these two requirements, we define the *Snapshot Vendor Selection* problem as follows:

Definition 1 (Snapshot Vendor Selection). Given a user u's location and her specified tolerance capacity $u.\gamma$, CALBA selects the actual advertising sources $\mathcal{A}_u$ from the candidate sources $\mathcal{V}_u$, such that ($\mathcal{O}1$) the sum of advertising frequencies of vendors in $\mathcal{A}_u$ does not exceed $u.\gamma$, i.e., $\sum_{v \in \mathcal{A}_u} v.f \leq u.\gamma$, and ($\mathcal{O}2$) the total relevance of vendors in $\mathcal{A}_u$ is maximized.

In this paper we model this problem as 0-1 knapsack and solve it using a fully polynomial approximation scheme [18]. However, the real challenge lies in the continuous version of this problem, i.e., the users keep moving around the hotel, and thus the relevance of vendors to users changes all the time due to the change of their spatial distance. A straightforward solution to continuously monitor the selection result is to periodically invoke the snapshot vendor selection. However, if the period is too short, it will incur unnecessary computation cost in the server as well as increase communication cost between the client and the server. To illustrate this, the dashed curve in Figure 1 shows the trajectory of a mobile user u. We assume that the correct selection result for u changes from $\{v_1, v_3, v_4\}$ to $\{v_1, v_2, v_4\}$ as u moves. Using the straightforward solution with a short invoking period, we compute the selection result at times t_1, t_2, t_3 and t_4, in which the computation at time t_2 is not necessary. On the other hand, if the period is too long, the vendor selection result may be incorrect during the long interval of two invoking points. Continue with the example. If we only invoke the selection process at times t_1 and t_4, then the selection result may be sometimes incorrect (e.g., at time t_3). In general, it is impossible to set a certain period which avoids unnecessary computations and offers correctness at the same time.

To this end, we employ the *safe region* technique which is similar to that for moving spatial queries [4, 14, 19]. However, their techniques cannot be applied to our problem due to the differences in inherent features between their queries and our vendor selection. In this paper, for a mobile user u we define and construct a safe region for the Continuous Vendor Selection problem. The user u is able to monitor whether she is inside her safe region. As long as u remains in her safe region, the result of vendor selection for u remains the same, and there is no need for CALBA to re-select vendors for u. In the example of Figure 1, if u remains in her current safe region (represented by the shaded area), her selection result (i.e., $\{v_1, v_3, v_4\}$) remains the same. u will notify CALBA with her location when she moves out from her safe region. CALBA then assigns new advertising sources for u by performing vendor selection, and returns an updated safe region to u. The advantage of exploiting the safe region technique is that it enables CALBA to perform vendor selection for mobile users when necessary, and hence it avoids unnecessary server-side computation and communication between the client and the server. Furthermore, to save the cost of checking whether u lies in her safe region at the client side, we propose three pruning rules and an access order to effectively prune vendors which do not contribute to the final safe region. The contributions of our work can be summarized as follows:

- We propose the Snapshot Vendor Selection problem and model it as 0-1 knapsack, which can be solved by a fully polynomial approximation scheme.
- We design a relevance measure function to evaluate the relevance of a vendor to a user, which combines spatial and non-spatial factors (e.g., users' category preferences).
- We define a *conservative* safe region for moving users to address the Continuous Vendor Selection problem. For optimization purposes, we propose three pruning rules and a unique access order to effectively prune vendors which do not contribute to the final safe region, thus reducing the communication cost between the client and the server as well as the computation cost of the client.

- We evaluate CALBA through extensive experiments based on a real social network data set. Experimental results show that the performance of our safe region based approach is an order of magnitude better than a naïve approach which periodically invokes the snapshot vendor selection.

The rest of this paper is organized as follows. We first review related work in Section 2. Then, we present our CALBA framework in Section 3. The vendor relevance measure function is described in Section 4. Section 5 models the snapshot vendor selection problem as 0-1 knapsack and solves it approximately. In Section 6, we define the safe region for the Continuous Vendor Selection and propose optimization techniques. Section 7 analyzes experimental results. Finally, Section 8 concludes this paper.

2. RELATED WORK

Continuous monitoring of spatial queries. Continuous monitoring of spatial queries (e.g., range queries [4, 21], nearest neighbor queries [14, 21] and top-k spatial keyword queries [8, 19]) has been extensively studied in recent literatures, most of which employ the technique of safe region to reduce the client-server communication cost significantly. It is guaranteed that the query result remains unchanged as long as the user is within the safe region. A new result and an updated safe region are requested from the server if the user leaves the safe region.

In the inspiring work [4] Cheema et al. define the safe region for a moving circular range query, which is shaped as the intersection of a set of *circles*. Furthermore, they propose powerful pruning rules and a unique access order to identify the guard objects which shape the final safe region. At first glance, we cannot apply their techniques due to inherent differences between circular range queries and the vendor selection problem (Definition 1). Innovatively, we utilize the characteristics of the approximation scheme for snapshot vendor selection, such that for a moving user, the safe region can be defined as the intersection of a set of *annuli*. Furthermore, we propose some pruning rules to effectively prune non-influential vendors which do not contribute to the final safe region.

The safe region of a k nearest neighbor query can be captured by an order-k Voronoi cell [21] or a V*-diagram [14]. A more related type of spatial queries to our problem is the top-k spatial keyword query, which returns a list of k objects ranked according to a ranking function that combines their distances to the query location and their textual relevance to the query keywords [8]. Similarly, in our vendor selection problem, to measure the relevance of a vendor to a user, we consider the category preferences of the user as well as their spatial distance (see Section 4). For continuous monitoring of top-k spatial keyword queries, Wu et al. [19] propose to use a multiplicatively weighted Voronoi cell [15] as the safe region of an object. However, we cannot extend it to address the continuous vendor selection in our scenario.

Location-based advertising. Related research work on location-based advertising (or mobile advertising) has focused on four major directions. (a) *Privacy issues.* Researchers pay more attention on the protection of the users' privacy since location-based advertising may become extremely intrusive in a personal space [3, 7]. (b) *Customer attitudes investigation.* Some researchers investigate consumer attitudes toward mobile advertising and the relationship between attitudes and behaviors [6,17]. (c) *Business model.* It would be essential in the successful deployment of location-based services to fully utilize multiple business models of mobile advertising [9, 16]. (d) *Personalized/contextualized advertising.* Advertisers create tailor-made advertisements for different users according to their locations, their needs of the moment and the devices they are using [5, 20]. Our work belongs to the fourth category since we consider the location proximity and preferences of users to select advertising sources (i.e., vendors). To our best knowledge, this paper is the first work aiming at reducing the client-server communication cost by using safe region techniques in personalized mobile advertising.

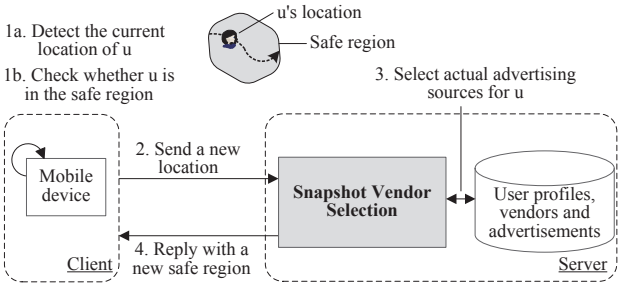


Figure 2: Framework architecture

3. SYSTEM OVERVIEW

In this section we illustrate the framework architecture of CALBA, as depicted in Figure 2. We employ the client-server architecture, i.e., the location of a client is updated to a server through a wireless network. The server stores: (i) vendors and their advertisements, (ii) users' candidate/actual advertising sources, and (iii) user profiles for computing the relevance of vendors (details will be illustrated in Section 4). The mobile user u repeatedly obtains her current location using her GPS-equipped mobile device (Step 1a), and checks whether the location is in her current safe region (Step 1b). As long as u stays inside the safe region, the vendor selection result is guaranteed to remain unchanged. When u exits the safe region, she notifies CALBA to perform vendor selection with her updated location (Step 2). By performing snapshot vendor selection (Definition 1), CALBA assigns updated advertising sources for u (Step 3). Finally, CALBA computes the updated safe region and returns it to u (Step 4). The process starts with Step 1a and 1b again. Once a candidate vendor v is selected as an actual advertising source for u, the service provider will deliver any advertisement generated by v to u.

4. VENDOR RELEVANCE MEASURE

CALBA requires the *relevance measure* function to return a score to indicate the relevance of a vendor v to a user u, i.e., *relevance(u, v)*. We combine the following non-spatial and spatial factors to implement this function.

4.1 Category Preferences

Users usually have preferences for certain vendor categories. For instance, shopaholics would be interested in discount information from nearby shopping malls. In the following, we first describe a method to construct user profiles

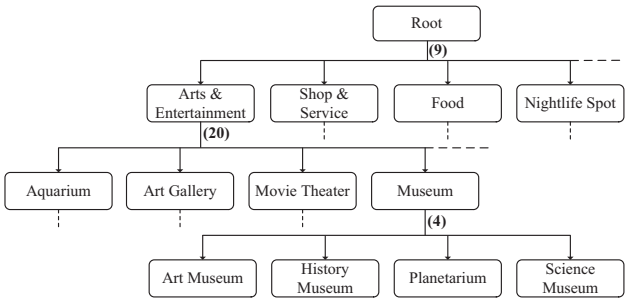


Figure 3: Fragment of categories from Foursquare

from users' check-in history, and then derive the relevance between users and vendors in terms of category preferences.

Categorization-driven user profile construction. In CALBA, vendors belong to difference categories. Since categories have different granularities, they could be represented by a *category hierarchy* (i.e., a tree) [2]. For example, Figure 3 depicts the fragment of category hierarchy obtained from Foursquare. Specifically, set $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$ denotes all categories of vendors, where each entry represents a node in the category hierarchy. Leaf nodes are categories with zero outdegree, i.e., most specific categories. The root represents the most general category. Note that each vendor may be associated with more than one category, since categorization into only one single category would incur loss of precision. Formally, function $g : \mathcal{V} \to 2^{|\mathcal{C}|}$ assigns a set of categories to every vendor $v \in \mathcal{V}$.

Ziegler et al. [22] propose to utilize the categories of rated products to construct a vector as the user profile for further recommendation. We adopt their method to assemble a *category vector* for each user u as her profile. The category vector characterizes u's preferences for categories as a whole, and it is used to calculate the relevance of vendors to users in terms of category preferences, as we shall see. The category vector $\overrightarrow{W_u} = (w_1, w_2, \ldots, w_{|\mathcal{C}|})^T$ for u is composed of $|\mathcal{C}|$ entries, where w_e gives the weight for category $c_e \in \mathcal{C}$. For each user u, CALBA extracts a set *History*(u) from u's main social networks (e.g., Foursquare), which contains vendors (i.e., locations) checked in by u. To construct the category vector $\overrightarrow{W_u}$ for u, we perform the following two steps for each vendor $v \in$ *History*(u):

Step 1. For v, we distribute a score s evenly among v's associated categories (i.e., $g(v)$). Specifically, for each category $c_e \in g(v)$ categorizing vendor v, we assign score $sc(c_e) = \frac{s}{|g(v)|}$.

Step 2. For the score allocated to each category, i.e., $sc(c_e)$, we distribute it to the category itself and its *super-categories* in the category hierarchy (e.g., Figure 3). Concretely, let $(p_0, p_1, \ldots, p_q)$ denotes the path from top element p_0 (i.e., root) to descendant $p_q = c_e$ within the category hierarchy. Then, we distribute score $sc(c_e)$ to node c_e and its q super-categories. Let $sco(p_a)$ denotes the final assignment of score to category p_a along the path from p_q (i.e., c_e) to p_0, then we have:

$$\sum_{a=0}^{q} sco(p_a) = sc(c_e) \tag{1}$$

However, the score is not evenly distributed along the path. Intuitively, the score assigned to super-categories drops with increasing distance from node c_e in the hierarchy. For category p_a which is the super-category of p_{a+1}, its score $sco(p_a)$ with respect to $sco(p_{a+1})$ depends on the number of siblings of p_{a+1} (denoted as $sib(p_{a+1})$):

$$sco(p_a) = \frac{sco(p_{a+1})}{sib(p_{a+1}) + 1} \tag{2}$$

Finally, each computed score $sco(p_a)$ adds up for category p_a in the category vector. After performing above-mentioned two steps for each $v \in$ *History*(u), we could build a category vector $\overrightarrow{W_u}$ for u.

Example. Suppose the category hierarchy is depicted in Figure 3, where the number labeled is the number of children of corresponding node. For instance, the root has nine children. For a user u, a visited vendor $v_1 \in$ *History*(u) is associated with two categories, one of them pointing to leaf node Art Museum in the category hierarchy (Figure 3). Suppose that $s = 200$. Then the score assigned to category Art Museum is $\frac{200}{2} = 100$. Ancestors of leaf node Art Museum are Museum, Arts & Entertainment, and the top element Root. Score 100 should be distributed among these four categories according to Equation 1 and 2. Computation result yields score 79.121 for Art Museum, 19.780 for Museum, 0.989 for Arts & Entertainment, 0.110 for Root. These values then add up to the category vector of u.

Category preferences measure. With a category vector as the user profile, we want to measure the relevance of a vendor v to a user u in terms of category preferences. To achieve that, we also build a category vector $\overrightarrow{W_v} = (w'_1, w'_2, \ldots, w'_{|\mathcal{C}|})^T$ for each vendor v based on above-mentioned two steps. Then, we could utilize the Pearson correlation to define the relevance of v to u in terms of category preferences:

$$preference(u, v) = \frac{\sum_{e=1}^{|\mathcal{C}|}(w_e - \overline{w}) \cdot (w'_e - \overline{w'})}{\sqrt{\sum_{e=1}^{|\mathcal{C}|}(w_e - \overline{w})^2 \cdot \sum_{e=1}^{|\mathcal{C}|}(w'_e - \overline{w'})^2}} \tag{3}$$

where $\overline{w}$ and $\overline{w'}$ give the mean values for vectors $\overrightarrow{W_u}$ and $\overrightarrow{W_v}$, respectively. The range of $preference(u, v)$ is $[-1, +1]$.

4.2 Distance

We argue that the geographical proximities between users and vendors have a significant effect on the relevance measure. In our scenario, the relevance of a vendor v to a user u can be measured by their Euclidean distance, i.e., $||u\ v||$.

4.3 Integrated Relevance Measure

To integrate the above non-spatial and spatial factors, we employ a method similar to [19] to derive the final relevance of a vendor v to a user u:

$$relevance(u, v) = \frac{1 + preference(u, v)}{||u\ v||} \tag{4}$$

where we use $1 + preference(u, v)$ as the numerator to make the relevance score positive. A key advantage of this function is that the different measurement units of the preferences and the spatial distance do not affect the result, since the effect of the units is canceled in the ratio of two factors.

5. SNAPSHOT VENDOR SELECTION

In this section, we model the Snapshot Vendor Selection problem as the 0-1 knapsack, and then solve it using a fully polynomial approximation scheme.

5.1 0-1 Knapsack for Vendor Selection

First of all, the 0-1 knapsack problem is defined as follows:

DEFINITION 2 (0-1 Knapsack [18]). Given a set $S = \{a_1, \ldots, a_n\}$ of objects, with specified sizes and profits, as well as a knapsack capacity B, find a subset of objects whose total size is bounded by B, and their total profit is maximized.

For our Snapshot Vendor Selection problem (Definition 1), we map the relevance of vendors into profits of objects in knapsack problem. Similarly, we map the advertising frequencies of vendors into sizes of objects, and map the user-specified tolerance capacity into the knapsack capacity. Therefore, the Snapshot Vendor Selection problem can be modelled exactly as 0-1 knapsack.

5.2 Fully Polynomial Approximation Scheme

By modelling the Snapshot Vendor Selection problem as 0-1 knapsack, we can solve it exactly using *dynamic programming* [18]. However, dynamic programming requires that (i) relevances of vendors, (ii) advertising frequencies of vendors, and (iii) users' tolerance capacity should be integers. We assume that (ii) and (iii) are limited to integers when specified. To scale the relevance of a vendor v to a user u to integers, we set:

$$\mathit{relevance}(u, v) = \frac{1 + \mathit{preference}(u, v)}{||u\ v||} \cdot S \tag{5}$$

where S is the unified scaling factor for relevances of vendors. The complexity of solving Snapshot Vendor Selection by dynamic programming is $O(|\mathcal{V}_u|^2 P)$ [18], where $|\mathcal{V}_u|$ is the number of vendors in u's candidate advertising sources, and P is the largest relevance score of vendors in $\mathcal{V}_u$, i.e., $P = \max_{v \in \mathcal{V}_u} \mathit{relevance}(u, v)$. It is a *pseudo-polynomial* time algorithm and it may be expensive due to the value of P.

To this end, we propose to utilize the *fully polynomial approximation scheme* (FPTAS) of 0-1 knapsack [18] to solve the Snapshot Vendor Selection problem approximately, as described in Algorithm 1. The key idea is that, given the error parameter ε set by the system, we ignore a certain number of least significant bits of relevances of vendors (Lines 1 and 2 in Algorithm 1), such that the modified relevances can be regarded as numbers bounded by a polynomial in $|\mathcal{V}_u|$ and $1/\varepsilon$. According to [18], the FPTAS finds a solution whose total relevance is at least $(1 - \varepsilon) \cdot \mathsf{OPT}$ in time $O(|\mathcal{V}_u|^2 \lfloor \frac{|\mathcal{V}_u|}{\varepsilon} \rfloor)$, where OPT is the total relevance of optimal solution obtained using exact dynamic programming.

Algorithm 1 FPTAS [18] for Snapshot Vendor Selection

1: Given $\varepsilon > 0$, let $K \leftarrow \frac{\varepsilon P}{|\mathcal{V}_u|}$
2: For each vendor $v \in \mathcal{V}_u$, define $\mathit{relevance}'(u, v) = \lfloor \frac{\mathit{relevance}(u,v)}{K} \rfloor$
3: With these modified values as relevances of vendors, using dynamic programming to find the actual advertising sources $\mathcal{A}_u \subseteq \mathcal{V}_u$

In the next section, we will focus on the Continuous Vendor Selection, i.e., the user u keeps moving around the hotel. Specifically, we will exploit the characteristics of FPTAS to construct the safe region for u, such that the computation cost of the server as well as the communication cost between the server and the client can be reduced.

6. CONTINUOUS VENDOR SELECTION

In this section, we employ the technique of *safe regions* to tackle the Continuous Vendor Selection. Given u's current location, u's specified tolerance capacity $u.\gamma$, u's candidate sources $\mathcal{V}_u$ along with their advertising frequencies, we aim at computing a spatial region as the *safe region* for u, such that as long as u remains in this region, the result of vendor selection for u does not change. Thus, there is no need for CALBA to perform re-selection. Instead of trying to derive the *exact* safe region which may incur large computation overhead, we utilize the characteristics of the FPTAS for our Snapshot Vendor Selection (see Section 5.2) to derive the *conservative* safe region. This conservative safe region is a sub-region of the exact safe region, and it guarantees the correctness of the vendor selection result within the approximation ratio (see Section 5.2). However, if u moves out from the region, the selection result may or may not change. For simplicity, the conservative safe region is called the safe region in the remainder of the paper.

In the following, we first give an overview of the safe region technique in our scenario, and then present some optimization techniques, including pruning rules and the access order, to reduce the client-server communication cost and the computation cost of the client.

6.1 Solution Overview

In this section, we formally define the safe region for the Continuous Vendor Selection. We describe the mechanism for client-side checking and discuss how to design optimizations to save the client-server communication overhead and the client-side computational cost.

6.1.1 Safe Region for Continuous Vendor Selection

In our FPTAS for Snapshot Vendor Selection, we ignore some of the least significant digits of the relevance value of each vendor for the sake of efficiency. Specifically, according to Line 2 in Algorithm 1 and Equation 5, we have:

$$\mathit{relevance}'(u, v) = \lfloor \frac{(1 + \mathit{preference}(u, v)) \cdot S}{||u\ v|| \cdot K} \rfloor \tag{6}$$

With this equation, the intuition of our safe region technique is that, for each vendor $v \in \mathcal{V}_u$, when the distance between u and v (i.e., $||u\ v||$) varies in a certain range, the *modified value* of relevance of v to u (i.e., $\mathit{relevance}'(u, v)$) remains unchanged. Thus, the selection result does not change.

We want to derive such a distance range for each vendor v_i. Assume that $\mathit{relevance}'(u, v_i)$ equals an integer α_i, then we have:

$$\alpha_i \leq \frac{(1 + \mathit{preference}(u, v_i)) \cdot S}{||u\ v_i|| \cdot K} < \alpha_i + 1 \tag{7}$$

When we treat $||u\ v_i||$ as a variable, we can derive that $||u\ v_i|| \in (r_i, R_i]$, where

$$r_i = \frac{(1 + \mathit{preference}(u, v_i)) \cdot S}{K \cdot (\alpha_i + 1)} \tag{8}$$

$$R_i = \frac{(1 + \mathit{preference}(u, v_i)) \cdot S}{K \cdot \alpha_i} \tag{9}$$

Let O_i be an annulus of inner radius r_i and outer radius R_i with center at the location of the vendor v_i. When u

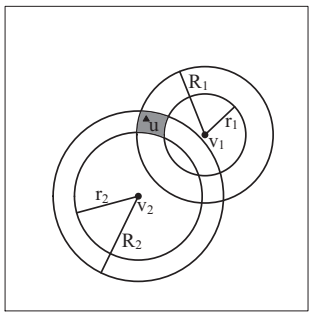


Figure 4: A user and her safe region

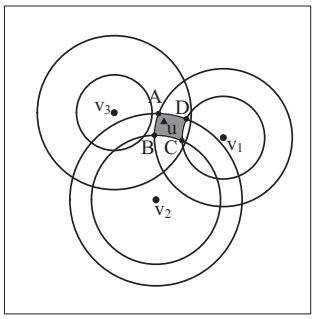


Figure 5: v_3 does not affect the safe region

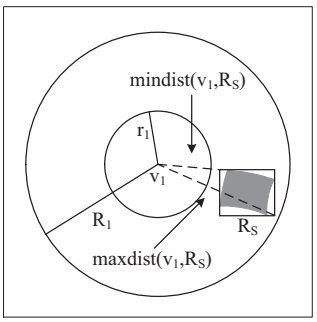


Figure 6: Pruning using approximation

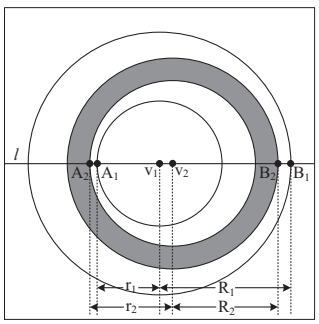


Figure 7: Pruning using complete coverage

moves within this annulus, we have: $||u\ v_i|| \in (r_i, R_i]$, so the relevance of v_i to u does not change. We term such annulus O_i as a *valid annulus* for vendor v_i with respect to u's current location. Based on this concept, we define the safe region for the Continuous Vendor Selection as follows:

DEFINITION 3 (Safe Region for Continuous Vendor Selection). Given a user u's current location, u's safe region is defined as the *intersection*[1] of valid annuli of all vendors in $\mathcal{V}_u$.

Figure 4 shows that there are two vendors (i.e., v_1 and v_2) in $\mathcal{V}_u$, and the part of intersection of their valid annuli containing u's current location (i.e., the shaded area) is the safe region, since the relevances of v_1 and v_2 to u do not change as long as u remains in this area.

6.1.2 Client-side Checking

Although the shape of a safe region could be very complicated, a straightforward solution just requires the server to send (the location of) all the vendors in $\mathcal{V}_u$ along with the inner and outer radii of their valid annuli to the client u. On the client side, the cost of checking whether u lies inside her safe region takes only σ distance computations and $2 \cdot \sigma$ distance comparisons, where σ is the number of vendors received from the server (e.g., $\sigma = |\mathcal{V}_u|$ in the straightforward case). Specifically, the client computes its distance from each of σ vendors. If the corresponding distance is in between the inner radius and outer radius of every vendor's valid annulus, then u lies within her safe region.

6.1.3 The Opportunity for Optimization

Since the client may have limited computational power (e.g., mobile devices), we require that the client-side checking should not be computationally expensive. In the straightforward case, the server sends all the vendors to the client for checking, i.e., $\sigma = |\mathcal{V}_u|$. However, we observe that the valid annulus of some vendors in $\mathcal{V}_u$ may not affect the shape of a safe region, so those vendors can be ignored during the safe region computation process. In the example of Figure 5, we have three vendors in $\mathcal{V}_u$. If we access v_3 after v_1 and v_2, we find that the valid annulus of v_3 completely covers the current safe region[2] (i.e., the shaded area formed by v_1 and v_2). Therefore, v_3 does not change the shape of the current safe region and thus will not affect the final safe region. In this paper, the vendors that contribute to the shape of the final safe region are called *influential* vendors (e.g., v_1 and v_2 in Figure 5). Specifically, each influential vendor has its arcs that contribute to the final safe region. In the example of Figure 5, v_1's arcs $\overset{\frown}{AB}$, $\overset{\frown}{CD}$ and v_2's arcs $\overset{\frown}{AD}$, $\overset{\frown}{BC}$ constitute the final safe region.

As discussed above, in the server side when we access the vendors in $\mathcal{V}_u$, we can identify those vendors which are not influential using pruning rules (will be covered in Section 6.2) and avoid sending them to the client. Since we reduce the number of vendors (i.e., σ) received by the client, we can save the client-server communication cost and the computation cost of the client.

6.2 Pruning Rules

In this section, we present three effective pruning rules to prune vendors which do not contribute to the final safe region. For brevity, from now on, r_i and R_i refer to the inner and outer radii of the valid annulus of a vendor v_i, respectively.

Using approximation of the safe region. First we present a pruning rule based on the approximation of the safe region by a rectangle. Let R_S be the minimum bounding rectangle of the current safe region S, as shown in Figure 6. Let $mindist(v_i, R_S)$ and $maxdist(v_i, R_S)$ be the minimum and maximum Euclidean distance between a vendor v_i and the rectangle R_S, respectively.

PRUNING RULE 1. Given a vendor v_i to be accessed, if $r_i \leq mindist(v_i, R_S) \leq maxdist(v_i, R_S) \leq R_i$, then v_i cannot affect the final safe region.

The proof is trivial and we omit it. In the example of Figure 6, where $r_1 \leq mindist(v_1, R_S) \leq maxdist(v_1, R_S) \leq R_1$, we can immediately verify that the valid annulus of vendor v_1 covers the current safe region S, so we can prune v_1 safely.

Using complete coverage between two annuli. As shown in Figure 7, the valid annulus of vendor v_1 completely covers that of vendor v_2, and we conclude that v_1 cannot become an influential vendor. Before we present the formal pruning rule, we provide an auxiliary lemma.

Consider a circle C with center at O and radius r, as shown in Figure 8. Any line through O intersects circle C at A and B. Consider any point D in this line, and without loss of generality, we assume D is closer to A than to B (see Figure 8). We have the following lemma.

[1]The intersection may contain multiple parts of regions (e.g., Figure 4). The safe region only refers to the part containing u's current location.

[2]A current safe region is a non-finalized safe region computed by the client during the safe region computation process.

LEMMA 1. Given a circle C centered at O as well as points A, B and D as described above, the Euclidean distance between D and any point E on the circle increases monotonically as E moves along the circle from A to B, either clockwise or counter-clockwise.

Proof. Consider the triangle $\triangle DOE$ in Figure 8. If we denote the length of $\overline{DO}$ by x, the length of $\overline{OE}$ is derived by the Cosine law as $||D\ E|| = \sqrt{r^2 + x^2 - 2rx \cdot \cos\theta}$, where θ denotes the degree of $\angle DOE$. As E moves along the circle from A to B, $\cos\theta$ decreases monotonically from 1 to -1 since θ increases from 0° to 180°. Since both r and x are constant, $||D\ E||$ increases monotonically. Note that the lemma also holds for the case when D lies outside the circle. □

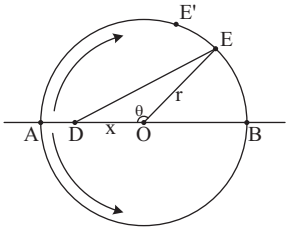


Figure 8: Lemma 1

Based on Lemma 1, we present the following pruning rule.

PRUNING RULE 2. Given a vendor v_i to be accessed, if there exists a vendor v_j belonging to current influential vendors, and we have: (a) $r_j \geq ||v_i\ v_j|| + r_i$ and (b) $R_i \geq ||v_i\ v_j|| + R_j$, then v_i cannot affect the final safe region.

Proof. Assume that in Figure 7, v_1 is the vendor to be accessed, and v_2 is a current influential vendor. When we draw a line l through center points v_1 and v_2, it intersects with the inner circle of v_2, the inner circle of v_1, the outer circle of v_2 and the outer circle of v_1 at A_2, A_1, B_2 and B_1 in turn.

According to Lemma 1, $||v_1\ A_2||$ is the minimum distance from v_1 to the inner circle of v_2. Furthermore, based on inequality (a), we have: $||v_2\ A_2|| - ||v_1\ v_2|| = ||v_1\ A_2|| \geq ||v_1\ A_1||$. Thus, the valid annulus of v_2 lies completely outside the inner circle of v_1.

Similarly, by Lemma 1, $||v_2\ B_1||$ is the minimum distance from v_2 to the outer circle of v_1. By inequality (b), we have: $||v_1\ B_1|| - ||v_1\ v_2|| = ||v_2\ B_1|| \geq ||v_2\ B_2||$. Thus, the valid annulus of v_2 lies completely inside the outer circle of v_1.

Combining above two facts, the valid annulus of v_1 completely contains that of v_2. Also, since v_2 is an influential vendor, its valid annulus covers current safe region. Therefore, the valid annulus of v_1 completely covers the current safe region, thus it cannot affect the final safe region. □

Using arcs of the safe region. For the first pruning rule, it is not tight due to the rectangle approximation; for the second pruning rule, it is not that effective since it requires two vendors to be close to each other. We present a tighter and more effective pruning rule below, based on the arcs shaping the current safe region.

Let $\overset{\frown}{AB}$ be an arc of radius r with subtending angle $\theta < 180°$ where O is the center (as shown in Figure 9 and Figure 10). First we want to define two spatial areas, namely, $\Theta(\overset{\frown}{AB}, r')$ and $\Omega(\overset{\frown}{AB}, r')$. For any point $p \in \Theta(\overset{\frown}{AB}, r')$, $\overset{\frown}{AB}$ lies completely inside the circle of radius r' with center at p. For any point $p \in \Omega(\overset{\frown}{AB}, r')$, $\overset{\frown}{AB}$ lies completely outside the circle of radius r' with center at p. The following two lemmas formally define these two areas.

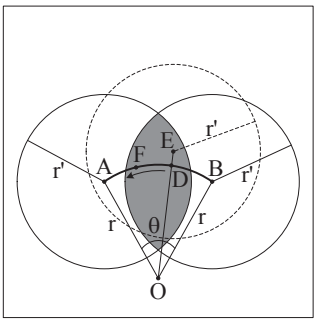


Figure 9: Lemma 2

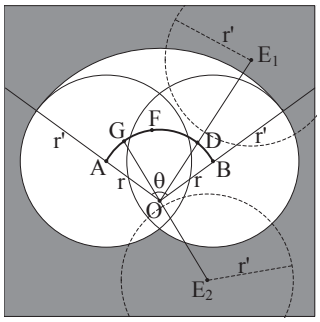


Figure 10: Lemma 3

LEMMA 2. As shown in Figure 9, let C_A and C_B be two circles of radius r' centered at A and B, respectively. Then the intersection of C_A and C_B (the shaded area in Figure 9) is exactly $\Theta(\overset{\frown}{AB}, r')$, i.e., for any point $E \in \Theta(\overset{\frown}{AB}, r')$, the circle of radius r' with center at E (the dotted circle in Figure 9) contains every point on the arc $\overset{\frown}{AB}$.

Proof. To prove this lemma, we need to show that the distance of E to any point F lying on the arc $\overset{\frown}{AB}$ is smaller than r'. We assume $r' \leq r$ in Figure 9. The line through O and E intersects $\overset{\frown}{AB}$ at D. By Lemma 1, for any point F on the arc $\overset{\frown}{DA}$ we have: $||E\ F|| \leq ||E\ A||$. Since point E lies inside the circle C_A, we have: $||E\ A|| \leq r'$. Thus, we have $||E\ F|| \leq r'$ for any point F. Similarly, we have the same conclusion when F lies on the arc $\overset{\frown}{DB}$. Note that the lemma also holds for the case when $r' > r$. □

LEMMA 3. As shown in Figure 10, the arc $\overset{\frown}{AB}$ lies completely outside the circle of radius r' centered at E (the dotted circles), if E satisfies either of the two conditions: 1) E lies within the angle range θ and $||E\ O|| > r + r'$ (e.g., E_1 in Figure 10); 2) E is outside the angle range θ and $||E\ A|| > r'$, $||E\ B|| > r'$ (e.g., E_2 in Figure 10). In other words, the shaded area in Figure 10 is exactly $\Omega(\overset{\frown}{AB}, r')$.

Proof. To prove this lemma, we need to show that the distance of E to any point F lying on the arc $\overset{\frown}{AB}$ is larger than r'. We assume $r' \geq r$ in Figure 10. We first consider a point E_1 that is inside the angle range θ (Figure 10). The line through O and E_1 intersects $\overset{\frown}{AB}$ at D. By Lemma 1, $||E_1\ D||$ is the minimum distance from E_1 to the arc $\overset{\frown}{AB}$. Since $||E_1\ O|| > r + r'$, it follows that $||E_1\ D|| > r'$. Thus, $||E_1\ F|| > r'$ for any point F lying on the arc $\overset{\frown}{AB}$.

Similarly, for the point E_2 lying outside the angle range θ (Figure 10), by Lemma 1 we can derive that the minimum distance from E_2 to arc $\overset{\frown}{AB}$ is $||E_2\ A||$ or $||E_2\ B||$. Since $||E_2\ A|| > r'$ and $||E_2\ B|| > r'$, for any point F lying on the arc $\overset{\frown}{AB}$, we have $||E_2\ F|| > r'$. Note that this lemma also holds for the case when $r' < r$. □

As a remark, we borrow ideas of Lemma 1, 2 and 3 from [4]. Based on Lemma 2 and 3, we have the third pruning rule:

PRUNING RULE 3. Let S be a current safe region such that every arc that shapes it has subtending angle smaller than 180°. Given a vendor v_i to be accessed, if $v_i \in \Theta(\xi, R_i) \cap \Omega(\xi, r_i)$ for every arc ξ of the safe region S, then v_i cannot affect the final safe region.

Proof. For every arc ξ of the safe region S, since $v_i \in \Theta(\xi, R_i)$, by Lemma 2, ξ lies completely within the outer circle of v_i; since $v_i \in \Omega(\xi, r_i)$, by Lemma 3, ξ lies completely outside the inner circle of v_i. Thus, the valid annulus of v_i contains every arc of the safe region S. Therefore, it completely covers the whole current safe region and cannot affect the shape of final safe region. □

Note that although the first two pruning rules are less effective, they are important since they require much less computation cost than the third one. So we will employ rule 1, 2 and 3 as an order of pruning. Our experimental results show that the average number of unpruned vendors (i.e., σ) remains under 20 even when $\mathcal{V}_u = 100$ (Figure 16 in Section 7).

6.3 Access Order

In order to compute the final safe region for u, we need to access all vendors in $\mathcal{V}_u$ to determine which vendors can be pruned by using rules in Section 6.2. Actually, the order in which the vendors are accessed is important. In the example of Figure 5, three vendors (i.e., v_1, v_2 and v_3) are to be accessed in order to determine the safe region. If we access v_3 first, none of three vendors will be pruned. However, if we access v_3 after v_1 and v_2, then v_3 can be pruned by using Pruning Rule 1 or 3.

As an optimization to enhance the power of pruning, we determine to access the vendors in ascending order of the difference between their inner and outer radii, i.e., a vendor v_i with a smaller value of $(R_i - r_i)$ will be accessed first. The main reason is that, by applying this order, a vendor accessed later has a higher chance to completely cover the current safe region, thus can be pruned. Our experimental results (Figure 17 in Section 7) demonstrate that our proposed access order significantly reduces the number of unpruned vendors, thus reducing the communication cost between the client and the server, as well as the computation cost of the client (see Section 6.1).

6.4 Algorithm to prune non-influential vendors

In this section we present the complete algorithm to prune non-influential vendors based on the proposed pruning rules (Section 6.2) and the unique access order (Section 6.3). As shown in Algorithm 2, the input is the user u and its candidate vendors (i.e., $\mathcal{V}_u$). For each vendor, we can compute its inner and outer radii according to Equation 8 and 9 in Section 6.1. We use data structure S to store the current safe region. Specifically, S maintains a list of entries, where each entry stores (1) an unpruned vendor's location, (2) its inner radius, and end vertices of its inner circle's arcs that contribute to the current safe region, and (3) its outer radius, and end vertices of its outer circle's arcs that contribute to the current safe region. For example, in Figure 11, v_1 and v_2 are two vendors shaping the current safe region (the shaded area). Then S contains two entries:

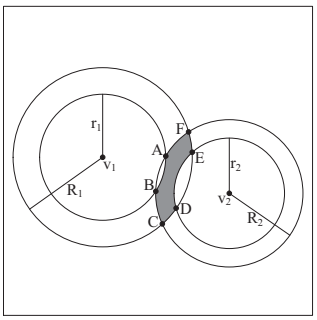


Figure 11: Current safe region

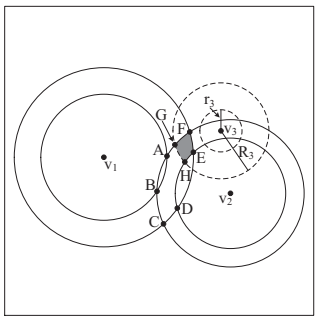


Figure 12: Trimming the safe region

$\{v_1, (r_1, A, B), (R_1, C, D, E, F)\}$ and $\{v_2, (r_2, D, E), (R_2, A, F, B, C)\}$. It indicates that v_1 contributes to the safe region by arc $\overset{\frown}{AB}$ of its inner circle with radius r_1 and arcs $\overset{\frown}{CD}$, $\overset{\frown}{EF}$ of its outer circle with radius R_1. Similar meaning holds for v_2.

Algorithm 2 Pruning (u, $\mathcal{V}_u$)

Input: u: the user; $\mathcal{V}_u$: u's candidate vendors with their inner and outer radii
1: Initialize the safe region S with no entry
2: Sort vendors in $\mathcal{V}_u$ in ascending order of the difference between their inner and outer radii ▷ Section 6.3
3: **for** each $v_i \in \mathcal{V}_u$ **do**
4: **if** v_i is pruned using rules 1, 2 or 3 **then**
5: **continue**
6: TrimSafeRegion(v_i, S) ▷ Algorithm 3
7: Send the list of unpruned vendors along with their inner and outer radii (stored in S) to u

After initializing S with an empty set, we sort the vendors in $\mathcal{V}_u$ in ascending order of the difference between their inner and outer radii (see Section 6.3). Then, for each vendor v_i, we apply the pruning rules. If v_i cannot be pruned by any of the three rules, we use it to trim the current safe region S (will be covered in Algorithm 3). Finally, after checking all vendors, we send u the list of unpruned vendors along with their inner and outer radii.

Trimming the safe region. Algorithm 3 depicts the procedures to trim the safe region with respect to a vendor v. The objective of trimming is to maintain the arcs contributing to the shape of the current safe region, such that we can apply the pruning rules 1 and 3 correctly (see Section 6.2).

Specifically, for each vendor v_i in S, we compute the intersection points of valid annuli of v and v_i. If any such point lies on the boundary of the safe region, we insert it into a vertex set Z (Lines 2 to 5). Then, we add v and its related arcs (represented by vertices in Z) to S (Line 6). Finally, we discard the existing vertices that are no longer in the trimmed safe region (Lines 7).

In the example of Figure 12, we use v_3 to trim the current safe region formed by vendors v_1 and v_2. By computing the intersections of annuli, we derive: $Z = \{G, H\}$. Then we add an entry $\{v_3, (r_3), (R_3, G, H)\}$ into S. After that, we check v_1's and v_2's existing associated vertices in S and discard A, B, C, D since they lie outside the outer circle of v_3. However, after the remove, some "isolated" vertices

Algorithm 3 TrimSafeRegion (v, S)

Input: v: a vendor to be used to trim the safe region, along with its inner radius r and outer radius R; S: safe region

```
1: Create a vertex set Z ← ∅
2: for each vendor v_i in S do
3:     for each intersection x of valid annuli of v and v_i do
4:         if x lies on the boundary of the safe region S then
5:             Insert x into Z
6: Add v with its contributing arcs (i.e., vertices in Z) to S
7: Remove every vertex y in S if ||y v|| > R or ||y v|| < r
```

may appear in S. For instance, the entry for v_2 now becomes $\{v_2, (r_2, E), (R_2, F)\}$. For such vertices (e.g., E and F), we find vertices in Z that can be paired with them in the same circle. In our case, E is paired with H and F is paired with G, so we add vertices G and H to the entry for v_2. Finally, the data structure S for current safe region (the shaded area in Figure 12) is updated with three entries: $\{v_1, (r_1), (R_1, E, F)\}$, $\{v_2, (r_2, H, E), (R_2, G, F)\}$ and $\{v_3, (r_3), (R_3, G, H)\}$.

7. EXPERIMENT RESULTS

This section evaluates the performance of CALBA through experiments using a real data set. We first describe our experiment settings, and then present experiment results.

Experiment settings. All the experiments were run on a Ubuntu 11.10 machine with a 4.3GHz Inter Core i7-3770 processor and 16GB RAM. We use a real location-aware social network data set in New York City (NYC), USA, which was crawled from Foursquare [10] from December 15 to 25, 2012. Specifically, we randomly choose 20 hotels in NYC. For each hotel, we randomly choose 100 points of interest within the distance of 5km as the vendors for that hotel. We extracted the road map of NYC from the USA Census TIGER/Line Shapefiles [1], and in the region of each hotel (i.e., the circle of radius 5km centered at that hotel) we randomly generated 100 30-minute trajectories using the A* algorithm [12]. For each trajectory we assign a real Foursquare user such that we can construct her profile by examining all the locations she has checked in. Thus, we can calculate the category vectors of users and vendors (see Section 4.1), and derive their relevances using Equation 4. Furthermore, for each vendor we randomly assign an advertising frequency from 1 message per hour to 10 messages per hour. Unless mentioned otherwise, all users move with a constant speed of 6km/h, the tolerance capacity (γ) is 60 messages per hour, the number of vendors for each hotel ($|\mathcal{V}_u|$) is 60, the error parameter (ε) for our approximation scheme (see Section 5.2) is 0.1. To evaluate the performance of CALBA, we compare it with a naïve algorithm which periodically invokes the snapshot vendor selection.

7.1 Accuracy

In this experiment we investigate the accuracy of CALBA by comparing it with the ground truth. For the ground truth, we compute the solution using exact dynamic programming for every 1 second. Specifically, we calculate the average relative error of CALBA by $\frac{1}{N} \times \frac{|G-A|}{|G|}$, where N is the number of snapshot vendor selection invoked by the ground truth, G and A are the set of vendors selected by the ground truth and our method, respectively. Figure 13 shows that as the number of vendor increases, the relative error gets smaller. The main reason is that, according to Lines 1 and 2 in Algorithm 1 (Section 5.2), less number of least significant bits of relevances is rounded off when $|\mathcal{V}_u|$ gets larger; thus, the accuracy of the approximation algorithm would be higher. Then we compute the average error of CALBA as 2.46% (the dotted line in Figure 13). To fairly compare the computation cost between CALBA and the naïve algorithm in the next experiment, we also evaluate the accuracy of the naïve algorithm in the same way as that of CALBA. As depicted in Figure 14, when the invoking period of the naïve algorithm increases, it has higher chance to incur errors between two invoking points (recall the example in Figure 1), and it is thus expected that the relative error of the naïve algorithm gets larger. Figure 14 shows that the naïve algorithm has a similar relative error (2.42%) compared to that of CALBA (2.46%) when the invoking period is 4s. Therefore, we set 4s as the default invoking period of the naïve algorithm.

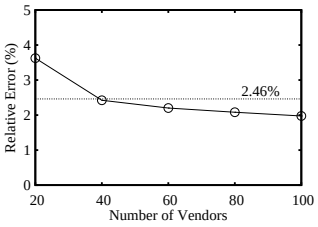


Figure 13: Accuracy of our method

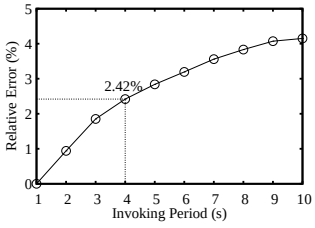


Figure 14: Accuracy of the naïve algorithm

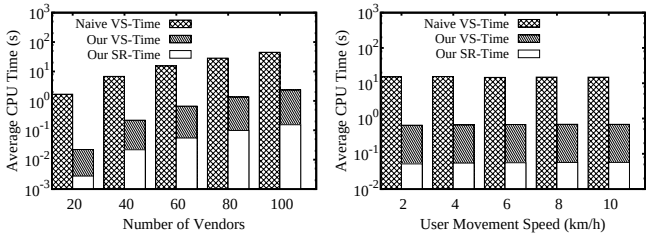
7.2 Cost Comparison

The CPU cost of our method consists of the cost for vendor selection (termed *VS-Time*) and the cost for safe region computation (termed *SR-Time*). In Figure 15, we compare the average cost of our method with the naïve algorithm with respect to various number of vendors and various user movement speed. Note that the CPU time in Figure 15 is for each 30-minute trajectory. The result shows that the performance of our method is about an order of magnitude better than the naïve algorithm, since it avoids unnecessary vendor selection by employing the safe region technique, thus saving the computation cost. Furthermore, the main cost for our method is for the vendor selection, which means the overhead of computing the safe region is very small compared to the cost of vendor selection. Figure 15a also shows that, as the number of vendors increases, the computation cost for the vendor selection becomes larger. This is expected according to the complexity of our approximation scheme for the vendor selection (see Section 5.2).

7.3 Effect of The Pruning Rules

This experiment studies the effect of our pruning rules on the performance of CALBA, as depicted in Figure 16. Our pruning rules could effectively identify those non-influential vendors and avoid sending them to the client, such that the communication cost between the client and the server as well as the computation cost of the client are reduced. Figure 16 shows that although the number of vendors increases from 20 to 100, the number of unpruned vendors sent to the client for checking remains under 20.

7.4 Effect of The Proposed Access Order



(a) Effect of the number of vendors (b) Effect of user movement speed

Figure 15: CPU Time (in log scale)

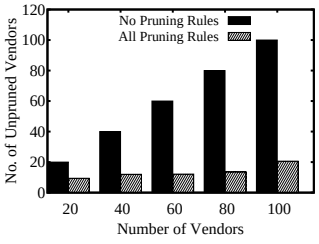


Figure 16: Effect of pruning rules

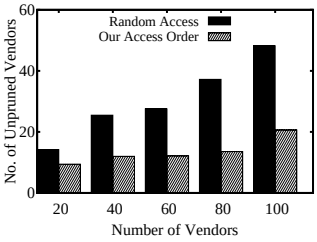


Figure 17: Effect of access order

In Figure 17, we evaluate the effectiveness of our proposed access order. The result shows that accessing vendors in ascending order of the difference between their inner and outer radii outperforms accessing vendors randomly. The main reason is that, by applying our proposed order, a vendor accessed later has a higher chance to completely cover the current safe region, thus can be pruned.

8. CONCLUSION

In this paper, we have presented a framework called CALBA for temporary social networks (TSNs) in which their service providers can effectively select appropriate third party vendors for their registered users. We have also employed the safe region technique to efficiently keep track of vendor selections for mobile users. Furthermore, we have designed three pruning rules and the unique access order to effectively prune vendors that could not affect a safe region in order to reduce the client-side computational cost. The performance of CALBA is evaluated through experiments based on a real location-based social network data set crawled from Foursquare. The experiment results show that the performance of CALBA is an order of magnitude better than the naïve approach in terms of CPU time and CALBA can effectively maintain a safe region.

9. REFERENCES

[1] 2010 Census TIGER/Line Shapefiles. `http://www.census.gov/geo/www/tiger/tgrshp2010/tgrshp2010.html`.

[2] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *ACM SIGSPATIAL GIS*, 2012.

[3] P. Barwise and C. Strong. Permission-based mobile advertising. *International Journal of Electronic Commerce*, 8(3):65–78, 2004.

[4] M. A. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wang. Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In *IEEE ICDE*, 2010.

[5] P.-T. Chen and H.-P. Hsieh. Personalized mobile advertising: Its key attributes, trends, and social impact. *Technological Forecasting and Social Change*, 79(3):543–557, 2012.

[6] H. K. Chowdhury, N. Parvin, C. Weitenberner, and M. Becker. Consumer attitude toward mobile advertising in an emerging market: An empirical study. *Marketing*, 12(2):206–216, 2010.

[7] E. B. Cleff. Privacy issues in mobile advertising. *International Review of Law, Computers & Technology*, 21(3):225–236, 2007.

[8] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *VLDB*, 2009.

[9] S. Dhar and U. Varshney. Challenges and business models for mobile location-based services and advertising. *Commun. ACM*, 54(5):121–128, 2011.

[10] Foursquare Developer API. `https://developer.foursquare.com/`.

[11] P. Haghirian, M. Madlberger, and A. Tanuskova. Increasing advertising value of mobile marketing-an empirical study of antecedents. In *Proc. of the 38th IEEE Annual Hawaii International Conference on System Sciences*, 2005.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE TSSC*, 4(2):100–107, 1968.

[13] LobbyFriend. `http://www.lobbyfriend.com`.

[14] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik. The v*-diagram: a query-dependent approach to moving knn queries. *Proceedings of the VLDB Endowment*, 1(1):1095–1106, 2008.

[15] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Wiley, 2009.

[16] J. Salo. The success factors of mobile advertising value chain. *Business Review*, 4:93–97, 2004.

[17] M. M. Tsang, S.-C. Ho, and T.-P. Liang. Consumer attitudes toward mobile advertising: An empirical study. *Journal of Interactive Marketing*, 16(1):14–24, 2002.

[18] V. V. Vazirani. *Approximation algorithms*. Springer, 2004.

[19] D. Wu, M. L. Yiu, and C. S. Jensen. Moving spatial keyword queries: Formulation, methods, and analysis. *ACM Trans. on Database Systems*, 38(1):7:1–7:47, 2013.

[20] S.-T. Yuan and Y. W. Tsao. A recommendation mechanism for contextualized mobile advertising. *Expert Systems with Applications*, 24(4):399–414, 2003.

[21] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In *ACM SIGMOD*, 2003.

[22] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *ACM CIKM*, 2004.