Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments

Baihua Zheng, Jianliang Xu, *Student Member*, *IEEE*, and Dik L. Lee

Abstract—Mobile location-dependent information services (*LDIS*s) have become increasingly popular in recent years. However, data caching strategies for *LDIS*s have thus far received little attention. In this paper, we study the issues of cache invalidation and cache replacement for location-dependent data under a geometric location model. We introduce a new performance criterion, called *caching efficiency*, and propose a generic method for location-dependent cache invalidation strategies. In addition, two cache replacement policies, *PA* and *PAID*, are proposed. Unlike the conventional replacement policies, *PA* and *PAID* take into consideration the valid scope area of a data value. We conduct a series of simulation experiments to study the performance of the proposed caching schemes. The experimental results show that the proposed location-dependent invalidation scheme is very effective and the *PA* and *PAID* policies significantly outperform the conventional replacement policies.

Index Terms—Mobile computing, location-dependent information, cache replacement, cache invalidation, semantic caching, performance evaluation.

1 Introduction

The advent of high-speed wireless networks and the popularity of portable devices have fueled the development of mobile computing. Compared to traditional computing paradigms, mobile computing enables clients to have unrestricted mobility while maintaining network connection. The ability of users to move and identify their own locations opens up a new kind of information services, called *location-dependent information services* (*LDIS*s), which produce the answer to a query according to the location of the client issuing the query [5], [8], [21]. Examples of mobile *LDIS*s include nearest object searching (e.g., finding the nearest restaurant) and local information access (e.g., local traffic, news, and attractions).

Mobile clients in wireless environments suffer from scarce bandwidth, low-quality communication, frequent network disconnections, and limited local resources [10]. Data caching on mobile clients has been considered an effective solution to improve system performance [1], [2]. There are two common issues involved in client cache management: A *cache invalidation scheme* maintains data consistency between the client cache and the server [2]; a *cache replacement policy* determines which data item(s) should be deleted from the cache when the cache does not have enough free space to accommodate a new item [1]. Various techniques have been proposed in the literature to address these two issues [1], [2], [20]. However, these studies do not explore the properties of location-dependent data.

• *The authors are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: {baihua, xujl, dlee}@cs.ust.hk.*

Manuscript received 15 July 2001; revised 15 May 2002; accepted 20 May 2002.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 116595.

The *spatial* property of location-dependent data introduces new problems for data caching research. First, the cached result for a query (e.g., the nearest restaurant) may become invalid when the client moves from one location to another. The maintenance of the validity of the cached data when the client changes location is called *location-dependent cache invalidation* [23]. Second, the cache replacement policy on the client has to consider the sizes of the *valid scopes* (hereinafter called *valid scope areas*) of the cached values. The valid scope of a data value is defined as the geographical area within which the data value is valid. When the valid scope of a data value is large, the chance for the client to issue the same query within the valid scope, thus generating a cache hit, is also large. As such, the cache replacement policy should try to retain the data value with a larger valid scope area in the cache. Let's consider an example in which the user is driving on the highway and wants to find the nearest hospital. If the hospital returned has a valid scope of 500 square kilometers,[1] it means that, within such an area, any query to find the nearest hospital will return the same hospital. Compared to the case when the valid scope area is only 5 square kilometers, meaning that another hospital will emerge as the nearest hospital after a few minutes of driving, it is clear that the result in the former has a higher chance of being a valid answer when the user asks the same query again and thus should have a higher priority to be kept in the cache.

A common way to perform location-dependent cache invalidation is to attach the valid scopes to the data values

1. As discussed later, a valid scope is in general a polygon, but, for the purpose of this example, it can be assumed to be a circle or square.

returned to the client [23].[2] Hence, an important aspect of cache invalidation in *LDIS* is how to identify and represent the valid scopes. This depends on the location model employed. In a symbolic location model, the valid scope of a data value is represented by a set of logical *ID*s (e.g., the *ID* of a cell in a cellular communication system). In contrast, a geometric location model represents a valid scope by the geographical coordinates of the area, which often takes the shape of a polygon. These two location models have different overheads and levels of precision in representing a valid scope. The appropriate location model to be adopted depends on the application.

In a previous study [22], [23], we addressed the issue of location-dependent cache invalidation for a cell-based symbolic location model. However, the same problem has not been explored for a geometric location model. Further, cache replacement policies for location-dependent data were not investigated in the previous study [22], [23]. In this paper, we study both location-dependent cache invalidation and replacement under a geometric location model. We first introduce two basic location-dependent invalidation schemes, namely *Polygonal Endpoints* (*PE*) and *Approximate Circle* (*AC*), for representing valid scopes with different overheads and levels of precision. Then, we introduce a new performance criterion, called *caching efficiency*, and propose a generic method, called *Cache-Efficiency Based scheme* (*CEB*), for balancing the overhead and the precision of a representation scheme. Based on this, we propose two novel cache replacement policies, namely *PA* and *PAID*, which take into consideration the valid scope area of a data value. To the best of our knowledge, no existing study has incorporated the factor of valid scope area in cache replacement policies.

To evaluate the performance of the proposed location-dependent data caching strategies, a simulation model is carefully designed and a series of simulation experiments are conducted. The experimental results show that the proposed location-dependent invalidation method, *CEB*, is very effective in cache performance and the proposed *PA* and *PAID* replacement policies outperform the conventional replacement policies significantly.

The rest of this paper is organized as follows: Section 2 reviews related work. The location-dependent information system model is described in Section 3. Section 4 discusses the location-dependent invalidation schemes and Section 5 proposes two cache replacement policies for location-dependent data. The simulation model is described in Section 6. In Section 7, the experimental results are presented. Finally, we close the paper with a brief conclusion and discussion of possible future work in Section 8.

2 Related Work

Caching data at mobile clients is an important technique for improving system performance in a mobile computing environment [1], [2]. In the following, we review the existing studies on cache invalidation and replacement strategies for mobile clients. Most of them were designed for general data services and only a few addressed the caching issues for location-dependent data.

As categorized in [22], [23], there are two kinds of cache invalidation methods for mobile databases: *temporal-dependent invalidation* and *location-dependent invalidation*. Temporal-dependent invalidation is caused by data updates. To carry out temporal-dependent invalidation, the server keeps track of the update history (for a reasonable length of time) and sends it, in the form of an *invalidation report* (*IR*), to the clients, either by periodic/aperiodic broadcasting or upon individual requests from the clients [2], [4], [11], [12]. In the basic *IR* approach, the server broadcasts a list of *ID*s for the items that have been changed within a history window. The mobile client, if active, listens to the *IR*s and updates its cache accordingly. Most of the existing temporal-dependent invalidation schemes are variations of the basic *IR* approach. They differ from one another mainly in the organization of *IR* contents and the mechanism of uplink checking. A good survey can be found in [19].

In location-dependent services, a previously cached data value may become invalid when the client moves to a new location. Location-dependent invalidation is due to mobile clients movements. In a previous paper [23], we assumed a cell-based symbolic location model and proposed three location-dependent invalidation schemes. Their performance was investigated using an analytical model in [22]. No location-dependent invalidation schemes have been proposed for a geometric location model.

Semantic data caching has been suggested for managing location-dependent query results [6], [15], where a cached result is described with the location(s) associated with the query. Unfortunately, the possibility was not explored that a cached data value may be valid for queries issued from locations different from that associated with the original query. As demonstrated in this paper, the exploration of this possibility can significantly enhance the performance of location-dependent data caching. As a matter of fact, the invalidation information in our methods can be considered a kind of semantic description which could improve cache hit rates.

Cache replacement policies for wireless environments were first studied in the broadcast disk project [1], [13]. In [1], the $\mathcal{PIX}$ policy took into consideration both data access probability and broadcast frequency during replacement [1]. In [13], the *Gray* scheme made replacement decisions based on both data access history and retrieval delays. Motivated by a realistic broadcast environment, an optimal cache replacement policy, called *Min-SAUD*, was investigated in [20]. The *Min-SAUD* policy incorporated various factors that affect cache performance, i.e., access probability, retrieval delay, item size, update frequency, and cache validation delay.

In the studies on location-dependent data caching, data-distance-based cache replacement policies, *Manhattan distance*, and *FAR*, have been proposed [6], [15]. In these two policies, the data that is farthest away from the client's current location is removed during replacement. However, in [6], [15], data distance (see Section 5.1 for a formal

2. In this paper, we assume item-based data queries and use query responses and data values interchangeably.

definition) was considered alone and not integrated with other factors such as access probability. Moreover, there are no existing studies that have considered the valid scope area in cache replacement policies.

3 The System Model

This section describes the system model adopted in this paper. We assume a cellular mobile network similar to that in [2] as the mobile computing infrastructure. It consists of two distinct sets of entities: mobile clients and fixed hosts. Some of the fixed hosts, called mobile support stations (*MSS*s), are augmented with wireless interfaces. An *MSS* can communicate with mobile clients within its radio coverage area, called a *wireless cell*. Mobile clients and the fixed data servers can communicate with each other through wireless channels via *MSS*s.

A mobile client can move freely from one location to another while retaining its wireless connection. Seamless hand-off from one cell to another is assumed. The information system provides location-dependent services to mobile clients. We refer to the geographical area covered by the system as the *service area*. A data item can show different values when it is queried by clients at different locations. Note that, in this paper, we distinguish *data item value* from *data item*, i.e., an item value for a data item is an instance of the item valid for a certain geographical region. For example, "nearest restaurant" is an item and the data values for this item vary when it is queried from different locations.

In this paper, we assume a geometric location model, i.e., a location is specified as a two-dimensional coordinate. Mobile clients can identify their locations using systems such as the Global Positioning System (*GPS*) [9]. We are now going to introduce two definitions: *valid scope* and *scope distribution*. The *valid scope* of an item value is defined as the region within which the item value is valid. The set of valid scopes for all of the item values of a data item is called the *scope distribution* of the item. In a two-dimensional space, a valid scope v can be represented by a geometric polygon $p(e_1, \cdots, e_i, \cdots, e_n)$, where e_is are endpoints of the polygon.

A mobile client can cache data values on its local disk or in any storage system that survives power-off. In this paper, data values are assumed to be of fixed sizes and read-only so that we can omit the influence of data sizes and updates on cache performance and concentrate on the impact caused by the unique properties of location-dependent data.

4 Location-Dependent Invalidation Strategies

In location-dependent services, the data value for a data item depends on geographical locations. Traditional caching strategies did not consider this unique characteristic and therefore are inefficient for location-dependent data. In this section, we discuss the location-dependent invalidation strategies.

For location-dependent invalidation, we have proposed in a previous paper [23] an idea similar to a semantic cache to attach complete/partial invalidation information to various data values. The advantages of this idea are summarized as follows:

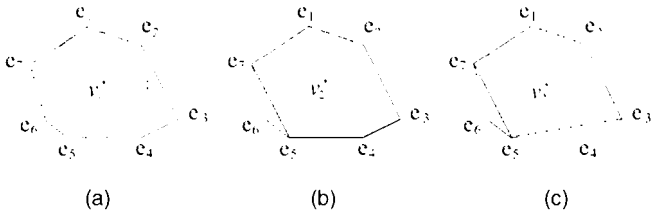


Fig. 1. An example of possible candidate valid scopes ($v = \mathrm{p}(e_1, e_2, \cdots, e_7)$). (a) $v'_1 =$ inscribed circle of v. (b) $v'_2 = \mathrm{p}(e_1, e_2, e_3, e_4, e_5, e_7)$. (c) $v'_3 = \mathrm{p}(e_1, e_2, e_3, e_5, e_7)$.

- The attached invalidation information provides a way for the client to check the validity of cached data with respect to a certain location without connecting the fixed data server. There are two situations where validity checking is necessary: 1) the same query may be issued later when the client has moved to a new location; 2) a mobile client may keep moving after it submits a query and the client may have moved to a new location when the response is returned if there is a long data access delay.
- As we will illustrate in Section 5, the invalidation information can be utilized by cache replacement policies to enhance performance.

The major issue of this idea is the organization of invalidation information for different data values. In this paper, we assumed that, when a data value is delivered from the server to the client, its complete valid scope is attached so that the client can check the data validity against its location of response time. However, different methods might be employed to represent a valid scope in the client cache. In the following, we first describe two basic schemes, Polygonal Endpoints (*PE*) and Approximate Circle (*AC*), followed by a generic method based on the caching efficiency criterion proposed in this paper.

4.1 The Polygonal Endpoints (PE) Scheme

The *PE* scheme is a straightforward way to record the valid scope of a data value. It records all the endpoints of the polygon representing the valid scope. However, when the number of the endpoints is large, the endpoints will consume a large portion of the client's limited cache space for storage, effectively reducing the amount of space for caching the data itself. This may worsen the overall performance.

4.2 The Approximate Circle (AC) Scheme

The *PE* scheme contains *complete* knowledge of the valid scope of a data value. Its performance suffers when a polygon has a large number of endpoints. An alternative is to use an inscribed circle to approximate the polygon instead of recording the whole polygon (e.g., the shadowed area as illustrated in Fig. 1a). In other words, a valid scope can be approximated by the center of the inscribed circle and the radius value. As can be seen, the overhead of this scheme can be minimized. For example, suppose eight bytes are used to record a point in a two-dimensional space;

a polygon having seven points needs 56 bytes, while a circle only needs 12 bytes: eight for the center and four for the radius. However, the inscribed circle is only a conservative approximation of a valid scope. When the shape of the polygon is thin and long, the imprecision introduced by the *AC* method is significant. This will lead to a lower cache hit ratio since the cache will incorrectly treat valid data as invalid if the query location is outside the inscribed circle but within the polygon.

4.3 The Caching-Efficiency-Based (CEB) Method

As we saw in the previous subsections, both the *PE* and the *AC* schemes may perform poorly due to either a high overhead or imprecision of the invalidation information. In this subsection, we propose a generic method for balancing the overhead and the precision of valid scopes.

We first introduce a new performance criterion, *caching efficiency*. Suppose that the valid scope of a data value is v, and v'_i is a subregion contained in v (see Fig. 1). Let D be the data size, $A(v'_i)$ the area of v'_i, and $O(v'_i)$ the overhead needed to record the scope v'_i. The caching efficiency of the data value with respect to a scope v'_i is defined as follows:

$$E(v'_i) = \frac{A(v'_i)/A(v)}{(D + O(v'_i))/D} = \frac{A(v'_i)D}{A(v)(D + O(v'_i))}. \quad (1)$$

If we assume that the cache size is infinite and the probabilities of a client issuing queries at different locations are uniform, $A(v'_i)/A(v)$ is a data value's cache hit ratio when the client issues the query within the valid scope v and v'_i is the approximated scope information stored in the client cache. In contrast, $(D + O(v'_i))/D$ is the cost ratio for achieving such a hit ratio. The rationale behind this definition is as follows: When none of the invalidation information is cached, $E(v'_i)$ is 0 since the cached data is completely useless; $E(v'_i)$ increases with more invalidation information attached. However, if too much overhead is therefore introduced, $E(v'_i)$ would decrease again. Thus, a generic method for balancing the overhead and the precision of invalidation information is proposed:

For a data item value with a valid scope of v, given a candidate valid scope set $V' = \{v'_1, v'_2, \cdots, v'_k\}$, $v'_i \subseteq v$, $1 \le i \le k$, we choose the scope v'_i that maximizes caching efficiency $E(v'_i)$ as the valid scope to be attached to the data.

Fig. 1 illustrates an example where the valid scope of the data value is $v = p(e_1, e_2, \cdots, e_7)$ and v'_1, v'_2, and v'_3 are three different subregions of v, $A(v'_1)/A(v) = 0.788$, $A(v'_2)/A(v) = 0.970$, and $A(v'_3)/A(v) = 0.910$. Assuming that the data size D is 128 bytes, eight bytes are needed to represent an endpoint, and four bytes for the radius of an inscribed circle, we have $O(v) = 56$, $O(v'_1) = 12$, $O(v'_2) = 48$, and $O(v'_3) = 40$. Thus, we obtain $E(v) = 0.696$, $E(v'_1) = 0.721$, $E(v'_2) = 0.706$, and $E(v'_3) = 0.694$. As a result, we choose v'_1 as the valid scope to be attached to the data.

In the proposed *CEB* method, a practical issue is the generation of the candidate valid scope set V'. There are various ways to do this. As a case study, in this paper, we consider the following method that can be handled by a general program.[3] Basically, we consider contained circles and subpolygons as candidate valid scopes. For circles, due to the lack of available geometric algorithms, we only consider the first-degree circle, i.e., the inscribed circle of a polygon, which can be obtained using the *medial axis* approach [14]. For subpolygons, we generate a series of candidate polygons in a greedy manner. Suppose the current candidate polygon is v'_i. We consider all polygons resulting from the deletion of one enpoint from v'_i and choose as the next candidate, v'_{i+1}, the polygon which is bounded by v and has the maximal area. The pseudo algorithm is described in Algorithm 1, where the generation of candidate valid scopes and the selection of the best valid scope are integrated.

Algorithm 1 Selection of the Best Valid Scope for the *CEB* Method

Input: valid scope $v = p(e_1, \cdots, e_n)$ of a data value;
Output: the attached valid scope v', $v' \subseteq v$;
Procedure:

```
1:  v'_1 := the inscribed circle of p(e_1, ..., e_n);
2:  v' := v'_1; E_max := E(v'_1);
3:  v'_i = p(e_1, ..., e_n);
4:  i := 2;
5:  while n - i >= 1
6:      //{containing at least three end-points for a polygon}
7:      if E(v'_i) > E_max then
8:          v' := v'_i; E_max := E(v'_i);
9:      end if
10:     if n - i > 1
11:         v'_{i+1} := the polygon that is deleted one endpoint
            from v'_i while being bounded by v and has the
            maximal area;
12:     end if
13:     i := i + 1;
14: end while
15: output v'.
```

5 Location-Dependent Cache Replacement Policies

Since a mobile client has only limited cache space, cache replacement is another important issue to be handled in client cache management. This section proposes two new cache replacement policies, *PA* and *PAID*, and discusses their implementation issues.

5.1 The Proposed *PA* and *PAID* Policies

In traditional cache replacement policies, access probability is considered the most important factor that affects cache performance. A probability-based policy is to replace the data with the least access probability. Policies such as *LRU*, *LFU*, and *LRU-K* are various implementations of the probability-based policy. However, in location-dependent services, besides access probability, there are two other factors, namely *data distance* and *valid scope area*, which should be considered in cache replacement. In the following

3. In a real-life application, it is possible to use other ad hoc methods to generate candidates while using the proposed *CEB* method to guide the selection of the best valid scope.

two paragraphs, we analyze respectively their impacts on cache performance.

Data Distance: Data distance refers to the distance between the current location of a mobile client and the valid scope of a data value. In a location-dependent data service, the server responds to a query with the suitable value of the data item according to the client's current location. As such, when the valid scope of a data value is far away from the client's current location, this data will have a lower chance to become usable again since it will take some time before the client enters the valid scope area again and the data is useless before the user reaches the valid scope area. In this respect, we should favor ejecting the "farthest" data when replacement takes place. However, this reasoning is invalid in the following two cases: First, if the client continues to move away from a location, this location would have a smaller chance of being revisited, even though the client's current location is very close to it. Thus, a directional data distance would make more sense. Second, with random movement patterns, the time it takes the client to traverse a distance is not always directly proportional to the distance. In summary, data distance, either directional or undirectional, may or may not affect cache performance, depending on the mobile client's movement and query patterns.

Valid Scope Area: Valid scope area refers to the geometric area of the valid scope of a data value. For location-dependent data, valid scope areas can somehow reflect the access probabilities for different data values. That is, the larger the valid scope area of the data, the higher the probability that the client requests this data. This is because, generally, the client has a higher chance of being in large regions than small regions. Thus, we argue that a good cache replacement policy should also take this factor into consideration. Fortunately, the proposed location-dependent cache invalidation schemes allow the client to calculate the valid scope areas of the cached data values.

Based on the above analysis, a promising cache replacement policy should choose as its victim data with a low access probability, a small valid scope area, and a long distance if data distance is also an influential factor. Therefore, in the following, we propose two cost-based cache replacement policies, *PA* and *PAID*, which integrate the factors that are supposed to affect cache performance.

- **Probability Area (PA):** As the name suggests, for this policy, the cost of a data value is defined as the product of the access probability of the data item and the area of the attached valid scope. That is, the cost function for data value j of item i is as follows:

$$c_{i,j} = P_i \cdot A(v'_{i,j}), \tag{2}$$

  where P_i is the access probability of item i and $A(v'_{i,j})$ is the area of the attached valid scope $v'_{i,j}$ for data value j. The *PA* policy chooses the data with the least cost as its victim when cache replacement is performed.
- **Probability Area Inverse Distance (PAID):** Compared with *PA*, this scheme further integrates the data distance factor. For the *PAID* policy, the cost function for data value j of item i is defined as follows:

$$c_{i,j} = \frac{P_i \cdot A(v'_{i,j})}{D(v'_{i,j})}, \tag{3}$$

  where P_i and $A(v'_{i,j})$ are defined in the same way as above and $D(v'_{i,j})$ is the distance between the current location and the valid scope $v'_{i,j}$. Similar to *PA*, *PAID* ejects the data with the least cost during each replacement. Depending on different methods of calculating $D(v'_{i,j})$, we have two variations of *PAID*, i.e., *PAID-U* and *PAID-D*. In *PAID-U*, the data distance is undirectional and is calculated regardless of the current direction of movement of the client. In *PAID-D*, the calculation of the data distance considers the client's current direction of movement: If the client is currently moving away from the valid scope, the distance is multiplied by a very large number δ (i.e., the longest distance in the system); otherwise, it is calculated normally, as in *PAID-U*. This way, *PAID-D* favors keeping the data in the direction of movement of the mobile client.

5.2 Implementation Issues

This subsection discusses the implementation issues for the proposed *PA* and *PAID* policies. In these two policies, there are three factors (i.e., access probability, valid scope area, and data distance) involved in the computation of the cost values for the cached data. Among these three factors, the valid scope area of a data value can be simply obtained based on the attached valid scope. In order to estimate the access probability for each data item, the well-known exponential aging method is employed. Two parameters are maintained for each data item i: a running probability (P_i) and the time of the last access to the item (t_i^l). P_i is initialized to 0. When a new query is issued for data item i, P_i is updated using the following formula:

$$P_i = \alpha/(t_c - t_i^l) + (1-\alpha)P_i, \tag{4}$$

where t_c is the current system time and α is a constant factor to weight the importance of the most recent access in the probability estimate.

Note that the access probability is maintained for each data item rather than for each data value. If the database size is small, the client can maintain the *prob* parameters (i.e., P_i and t_i^l for each item i) for all items in its local cache. However, if the database size is large, the *prob* information will occupy a significant amount of cache space. To alleviate this problem, we set an upper bound to the amount of cache used for storing the *prob* information (5 percent of the total cache size in our simulation) and use the *LFU* policy to manage the limited space reserved for the *prob* information.

For the *PAID* policy, we also need to compute the data distances between the current location and different valid scopes. Since a valid scope is normally a region (either a polygon or a circle in this paper) rather than a single point, we introduce a *reference point* for each valid scope and take the distance between the current location and the reference point as the data distance. For a polygonal scope, the reference point is defined as the endpoint that is closest to the current

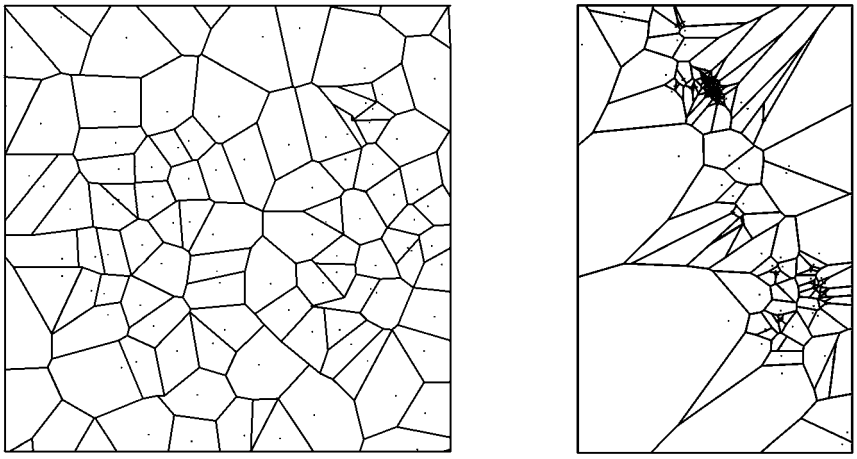


Fig. 2. Two scope distributions for performance evaluation. (a) Scope distribution 1: (*ScopeNum* = 110). (b) Scope distribution 2: (*ScopeNum* = 185).

location. For a circular scope, the reference point is defined as the point where the circumference and the line connecting the current location and the center of the circle meet.

6 Simulation Model

This section describes the simulation model used to evaluate the performance of the proposed location-dependent caching strategies. The discrete-time simulation package *CSIM* [16] is used to implement the model.

6.1 System Execution Model

Although a cellular mobile network consists of many cells, the network can coordinate between the clients and the cells to provide clients with a seamless service when they move across different cells. As such, the network can be considered a single, large service area within which the clients can move freely and obtain location-dependent information services.

In our simulation, the service area is represented by a rectangle with a fixed size of *Size*. We assume a "wrapped-around" model for the service area in which the left side is regarded as adjacent to the right side and the top side adjacent to the bottom side. In other words, when a client leaves one border of the service area, it enters the service area from the opposite border at the same velocity.

The database contains *ItemNum* items. Every item may display *ScopeNum* different values for different client locations within the service area. Each data value has a size of *DataSize*. In the simulation, the scope distributions of the data items are generated based on *Voronoi Diagrams (VDs)*[4] [18]. In general, a *VD* records the nearest object within a region. For instance, given a set P containing some fixed points on a plane, a *VD* identifies, for each $p \in P$, a region around p such that any point within the region has the shortest distance to p than to any other point in P [3]. In our simulation, there are two different scope distributions obtained from two different datasets (see Fig. 2). The first data set contains 110 points randomly distributed in a square Euclidean space. The second data set contains the locations of 185 hospitals in the Southern California area, which is extracted from the point data set at [7]. The model assumes that two floating-point numbers are used to represent a two-dimensional coordinate and one floating-point number to represent the radius. The size of a floating-point number is *FloatSize*.

The wireless network is modeled by an uplink channel and a downlink channel. The uplink channel is used by clients to submit queries and the downlink channel is used by the server to return query responses to target clients. The communication between the server and a client makes use of a point-to-point connection. It is assumed that the available bandwidth is *UplinkBand* for the uplink channel and *DownlinkBand* for the downlink channel. Table 1 summarizes the configuration parameters of the system model.

6.2 Client Execution Model

The mobile client is modeled with two independent processes: *query process* and *move process*. The query process continuously generates location-dependent queries for different data items. After the current query is completed, the client waits for an exponentially distributed time period with a mean of $QueryInterval$ before the next query is issued. The client access pattern over different items follows a *Zipf* distribution with skewness parameter θ [26]. When θ is set to 0, the access pattern is uniform; with increasing θ, the access pattern becomes more skewed. To answer a query, the client first checks its local cache. If the data value for the requested item with respect to the current location is available, the query is satisfied locally. Otherwise, the client submits the query and its current location uplink to the server and retrieves the data through the downlink channel.

The move process controls the movement pattern of the client using the parameter *MovingInterval*. After the client keeps moving at a constant velocity for a time period of *MovingInterval*, it changes the velocity in a random way: The next moving direction (represented by the angle relative to the x axis) is selected randomly between 0 and 360 and the next speed is selected randomly between *MinSpeed* and *MaxSpeed*. When the value of *MovingInterval*

4. The Voronoi diagram was employed to index locations for nearest object searching [24], [25].

TABLE 1
Configuration Parameters of the Server Execution Model

Parameter	Description
$Size$	size of the rectangle service area
$ItemNum$	number of data items in the database
$ScopeNum$	number of different values at various locations for each item
$DataSize$	size of a data value
$UplinkBand$	bandwidth of the uplink channel
$DownlinkBand$	bandwidth of the downlink channel
$FloatSize$	size of a floating-point number

TABLE 2
Configuration Parameters of the Client Execution Model

Parameter	Description
$QueryInterval$	average time interval between two consecutive queries
$MovingInterval$	time duration that the client keeps moving at a constant velocity
$MinSpeed$	minimum moving speed of the client
$MaxSpeed$	maximum moving speed of the client
$CacheSizeRatio$	ratio of the cache size to the database size
$ParaSize$	space needed for storing each parameter for cached data
θ	skewness parameter for the *Zipf* access distribution
α	weight factor for running access rate estimate

is small, the client's movement is rather random; when the value of *MovingInterval* is large, the movement of the client behaves more like a predefined trip which consists of long straight-line segments.

The client is assumed to have a cache of fixed size, which is a *CacheSizeRatio* ratio of the database size. In order to be fair to different caching schemes, the cache contains both the space needed for storing item parameters (e.g., the *prob* parameters for *PA* and *PAID*) and the space available for storing data. Each cached parameter occupies $ParaSize$ bytes. Table 2 summarizes the configuration parameters of the client model.

6.3 Server Execution Model

The server is modeled by a single process that services the requests from clients. The requests are buffered at the server if necessary and an infinite queue buffer is assumed. The *FCFS* service principle is assumed in the model. To answer a location-dependent query, the server uses a *planar point location algorithm (pploc)*[5] [17] to locate the correct data value with respect to the specified location. Since the main concern of this paper is the cost of the wireless link, which is more expensive than the wired-link and disk IO costs, the overheads of request processing and service scheduling at the server are assumed to be negligible in the model.

5. pploc uses the traditional *sweep line* algorithm to return the edge that is just below the point, then, by a mapping function, the corresponding region to the edge is returned. The algorithm runs in $O(ln(n))$ time, supposing n is the total edge number.

7 Performance Evaluation

In this section, the proposed location-dependent data caching strategies are evaluated using the simulation model described in the previous section. Table 3 shows the default parameter settings of the simulation model. In the experiments, two scope distributions with 110 and 185 valid scopes are used (see Fig. 2a and Fig. 2b). For our evaluation purposes, we assume that all data items follow the same scope distribution in a single set of experiments. Since the average valid scope areas differ for these two scope distributions, different moving speeds are assumed, i.e., the pair of (*MinSpeed*, *MaxSpeed*) is set to (1, 2) and (5, 10) for the first and the second scope distributions, respectively.

In the performance evaluation, *cache hit ratio* is employed as the primary performance metric. This is because most of the other performances can be derived from the cache hit ratio. Specifically, the higher the cache hit ratio, the higher the local data availability, the less the uplink and downlink costs, and the less the battery consumption. The results are obtained when the system has reached the stable state, i.e., the client has issued at least 20,000 queries, so that the warm-up effect of the client cache is eliminated.

TABLE 3
Default Parameter Settings for the Simulation Model

Parameter	Setting	Parameter	Setting
$Size$	4000*4000, 44000*27000	$QueryInterval$	50.0 s
		$MovingInterval$	100.0 s
$ItemNum$	500	$MinSpeed$	1, 5 s^{-1}
$ScopeNum$	110, 185	$MaxSpeed$	2, 10 s^{-1}
$DataSize$	128 bytes	$CacheSizeRatio$	10%
$FloatSize$	4 bytes	$ParaSize$	4 bytes
θ	0.5	$UplinkBand$	19.2 Kbps
α	0.25	$DownlinkBand$	144 Kbps

For cache replacement policies, the *LRU* policy, the pure probability-based *P* policy, and the data-distance-based *FAR* policy [15] are also included for comparison. For the *LRU* policy, the last-access timestamps are maintained for each cached data value. For the *P* policy, we evaluate two variations: *P_I* and *P_V*. Access probabilities in *P_I* and *P_V* are maintained in terms of data items and data values, respectively. For the *P_I*, *P_V*, and *FAR* policies, we employ the methods presented in Section 5 to estimate access probabilities and data distances. Both the *P* and the *FAR* policies break ties, if there are any, arbitrarily.

7.1 Evaluation of Location-Dependent Invalidation Schemes

This subsection examines the performance of different location-dependent invalidation schemes, namely *PE*, *AC*, and *CEB*. The *LRU* cache replacement policy is employed. As we will see in Section 7.3, the relative performance of the invalidation schemes does not change when combined with other replacement policies. Figs. 3 and 4 show the cache hit ratio performance for scope distributions 1 and 2, respectively, under various query and moving intervals.

We observed in the simulation that, when no location-dependent invalidation scheme is employed, the cache hit ratio is almost zero since the probability that the client issues the same query at the same place and that the query result is cached is extremely low. From Figs. 3 and 4, we can see that all proposed invalidation schemes improve the cache hit ratio significantly. When we compare the *PE*, *AC*, and *CEB* schemes, the proposed *CEB* method has the best performance overall. On average, *CEB* is 3 percent better than *PE* and 70 percent better than *AC*. As observed in the experiments, *CEB* caches more data than *PE* and contains

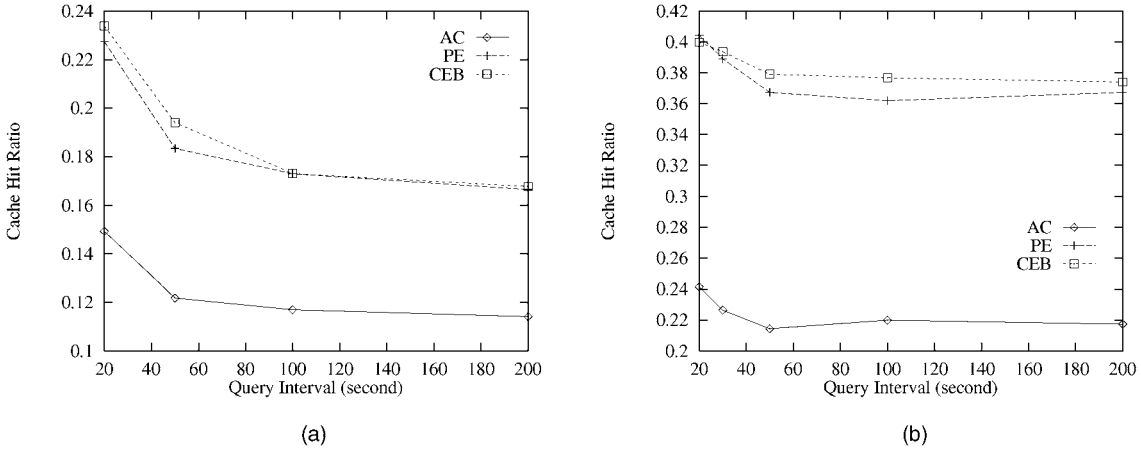


Fig. 3. Cache hit ratio of invalidation schemes vs. query interval. (a) Scope distribution 1. (b) Scope distribution 2.

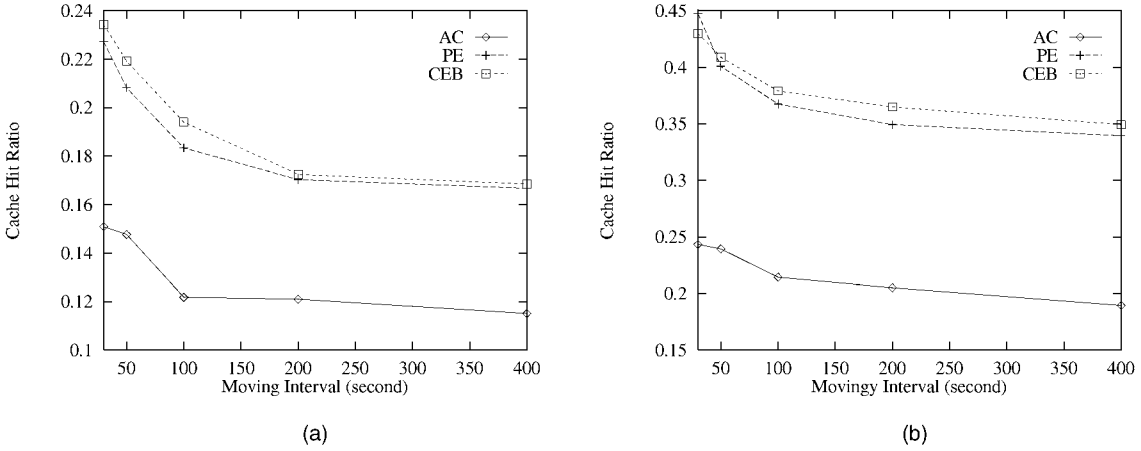


Fig. 4. Cache hit ratio of invalidation schemes vs. moving interval. (a) Scope distribution 1. (b) Scope distribution 2.

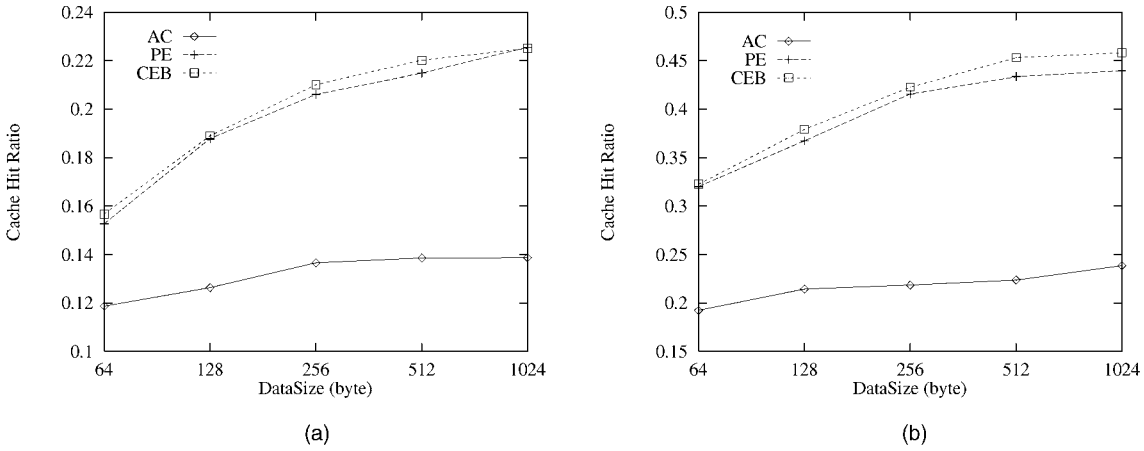


Fig. 5. Performance comparison of invalidation schemes vs. data size. (a) Scope distribution 1. (b) Scope distribution 1.

more invalidation information than *AC*. Therefore, as expected, *CEB* is a balance between the overhead and the precision of the invalidation information attached to the data. The results show that caching efficiency can be used as the criterion for the tradeoff with a good performance.

Fig. 5 shows the result when the data size is varied from 64 bytes to 1,024 bytes. When the data size is increased, the overhead of the invalidation information becomes relatively smaller. As such, all of the schemes get a better performance. It was observed in the simulation that *CEB* is able to adapt the invalidation overhead to different data sizes. The overhead is increased with increasing data size D. For example, for the first scope distribution, the average overhead introduced by the complete valid scopes is around 48 bytes. When D is 64 bytes, the average overhead of *CEB* is 30 bytes. It increases to 44 bytes when D is 1,024 bytes. As a result, *CEB* consistently outperforms the other two schemes for different data sizes. The improvement is about 51 percent and 2 percent over *AC* and *PE*, respectively, for the first scope distribution and 86 percent and 3 percent for the second scope distribution.

7.2 Evaluation of Cache Replacement Policies

In this subsection, the performance of the proposed cache replacement policies, namely *PA* and *PAID* (including *PAID-U* and *PAID-D*), is evaluated. We compare them to the existing policies *LRU*, *P_I*, *P_V*, and *FAR*. Since the *CEB* method shows the best cache performance in the previous subsection, it is employed here to represent the invalidation information. Two types of access patterns over data items, *uniform access* ($\theta = 0$) and *skewed access*($\theta = 0.5$), are simulated.

7.2.1 Effect of Changing Query Interval

Query interval is the time interval between two consecutive client queries. In this set of experiments, we vary the mean query interval from 20 seconds to 200 seconds. Figs. 6 and 7 show the cache performance for uniform access and skewed access, respectively.

As illustrated, when the query interval is increased, almost every scheme gets a worse performance. This is because, for a longer query interval, the client would make more movements between two successive queries, thus the client has a lower probability of residing in one of the valid scopes of the previously queried data items when a new query is issued.

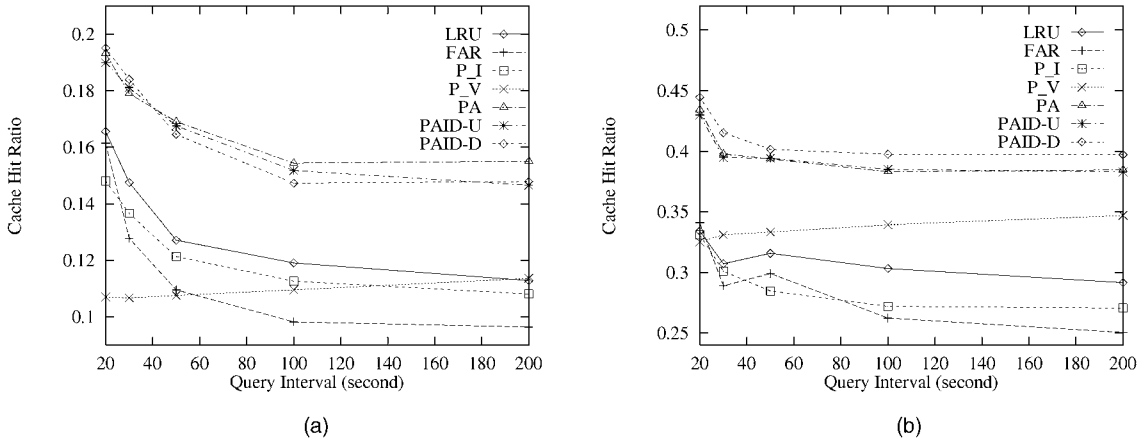


Fig. 6. Cache hit ratio vs. query interval (uniform access). (a) Scope distribution 1. (b) Scope distribution 2.

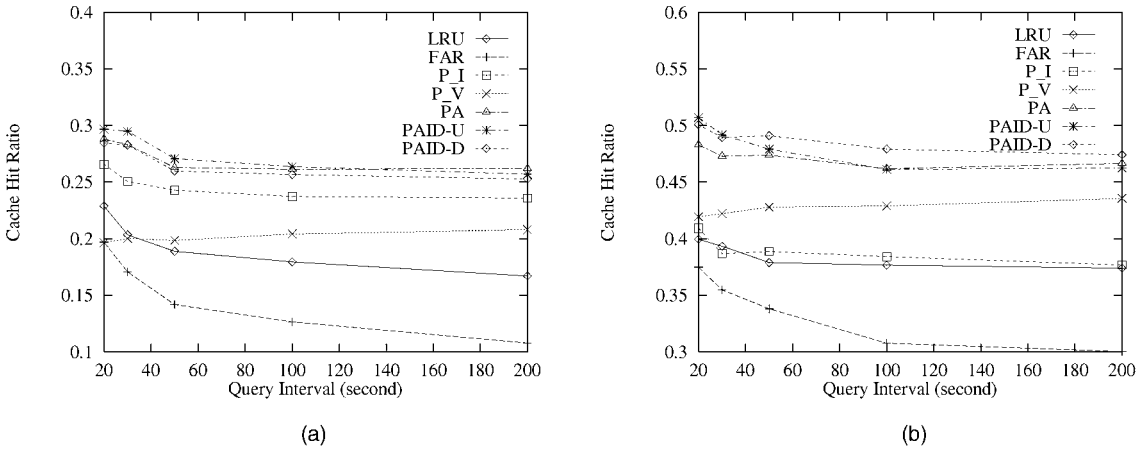


Fig. 7. Cache hit ratio vs. query interval (skewed access). (a) Scope distribution 1. (b) Scope distribution 2.

When different cache replacement policies are compared, the proposed policies substantially outperform the existing policies. For example, the *PA* policy improves over *P_I*, *P_V*, *LRU*, and *FAR* by 10.2 percent, 34.8 percent, 41.2 percent, and 89.3 percent, respectively, for the first scope distribution, and by 21.3 percent, 10.6 percent, 22.7 percent, and 41.6 percent, respectively, for the second scope distribution. In particular, the factor of valid scope area is crucial to the cache performance. Having taken into consideration the valid scope area, *PA*, in all cases, is much better than *P_I* and *P_V*. For the factor of data distance, the impact depends on the client's query pattern and the scope distribution. For uniform data access (Fig. 6), *PAID-D* obtains a slightly better performance than *PA* for the second scope distribution, with an average improvement of 3.2 percent; *PAID-U* cannot improve the performance over *PA* for both scope distributions. For skewed data access (Fig. 7), for both scope distributions, *PAID-U* has the best performance when the query interval is short; *PAID-D* performs best when the query interval is longer than 60 seconds for the second scope distribution.

7.2.2 Effect of Changing Moving Interval

This subsection examines the performance of the replacement policies when the moving interval is varied. The longer the moving interval, the less frequently the client changes velocity and, hence, the less random the client's movement. The performance results for uniform access and skewed access are shown in Figs. 8 and 9, respectively.

We can see that, when the moving interval is varied from 20 seconds to 200 seconds, the hit ratio performance decreases drastically. The reason for this is as follows: For a relatively longer moving interval, a larger average distance difference is observed for two successive queries, which implies that the client has a higher possibility of leaving certain valid regions. Consequently, the cached data are less likely to be reused for subsequent queries, which leads to a worse performance. After the moving interval reaches 200 seconds, such an influence is eliminated and, hence, the performance becomes flat.

As shown in Figs. 8 and 9, the proposed *PA* and *PAID*, in all cases, significantly improve the performance over the existing policies. In this set of experiments, the *PA* policy improves over *P_I*, *P_V*, *LRU*, and *FAR* by 10.51 percent, 34.9 percent, 40.5 percent, and 75.3 percent, respectively, for the first scope distribution, and by 18.8 percent, 10.5 percent,

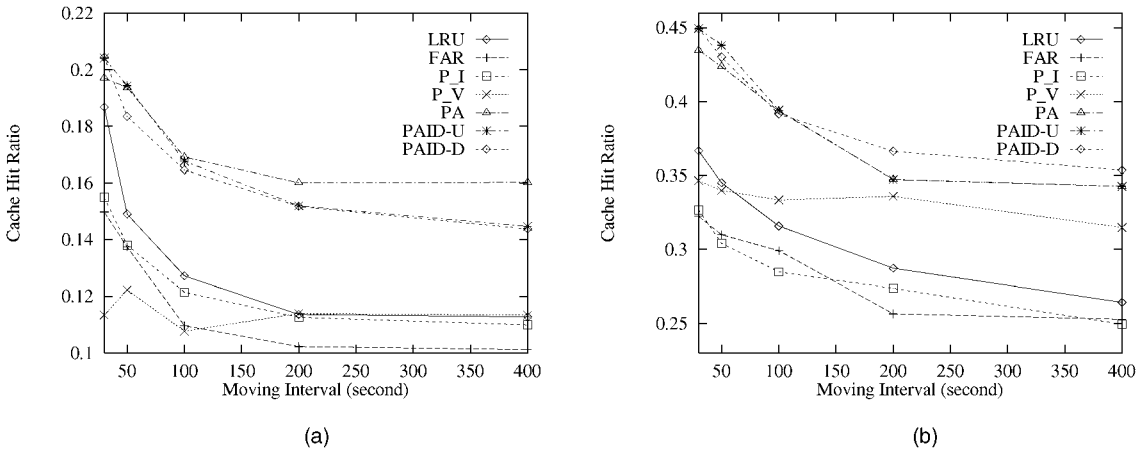


Fig. 8. Cache hit ratio vs. moving interval (uniform access). (a) Scope distribution 1. (b) Scope distribution 2.

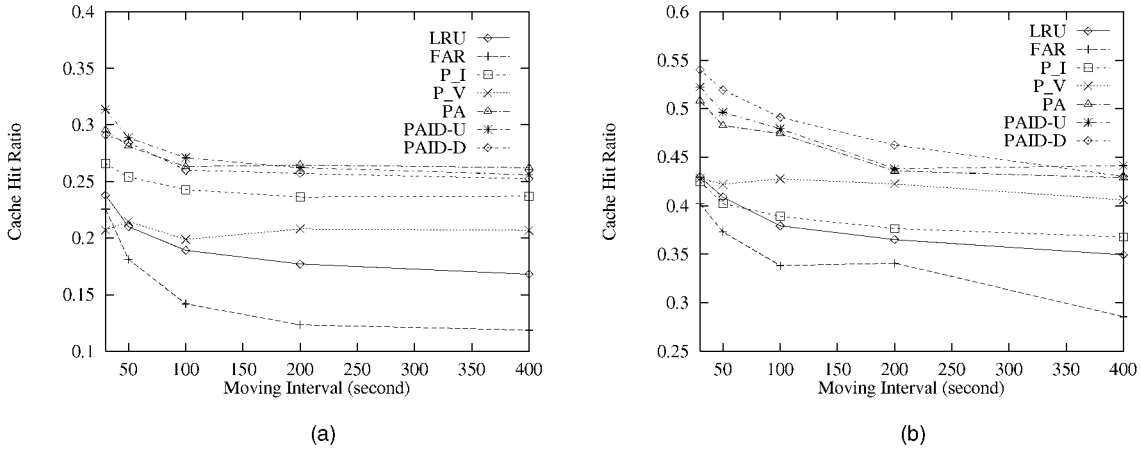


Fig. 9. Cache hit ratio vs. moving interval (skewed access). (a) Scope distribution 1. (b) Scope distribution 2.

20.7 percent, and 34.7 percent, respectively, for the second scope distribution. Again, consideration of the valid scope area in *PA* makes its performance much better than that of the *P_I* and *P_V* policies. On the other hand, the effect of the data distance factor depends on the scope distribution and the moving interval. For uniform data access (Fig. 8), for both scope distributions, *PAID-U* improves the performance slightly over *PA* when the moving interval is shorter than 50 seconds; *PAID-D* improves the performance only for the second scope distribution and has the best performance when the moving interval is longer than 100 seconds. For skewed data access (Fig. 9), the *PAID* policies have a better performance than *PA* only for short moving intervals: *PAID-U* gets the best performance for the first scope distribution, while *PAID-D* is the best for the second scope distribution.

7.2.3 Effect of Cache Size

In this set of experiments, we intend to investigate the robustness of the proposed replacement schemes under various cache sizes. Fig. 10 shows the results when *CacheSizeRatio* is varied from 5 percent to 20 percent. As expected, the performance of the replacement schemes improves with increasing cache size. The proposed *PA*, *PAID-U*, and *PAID-D* policies have a similar performance and consistently outperform the existing policies.

7.2.4 Summary

In summary, the proposed cache replacement policies, *PA* and *PAID*, substantially outperform the existing policies, namely, *LRU*, *P_I*, *P_V*, and *FAR*. In all settings, the proposed policies take advantage of the factor of the valid scope area and achieve a much better performance. In contrast, the factor of data distance is sensitive to scope distributions, client query patterns, and movement models.

7.3 Effect of Combining Different Invalidation and Replacement Schemes

To evaluate the efficiency of the proposed location-dependent invalidation schemes, the previous sets of experiments in Section 7.1 employed the *LRU* policy to manage the cache space. Here, we would like to investigate if their relative performance changes when combined with the *PA* and *PAID* policies, which show a superior performance over the other replacement policies in Section 7.2. Since, from the previous section, *PAID-U* provides better performance than *PAID-D* for the first scope distribution and *PAID-D* is more suitable for the

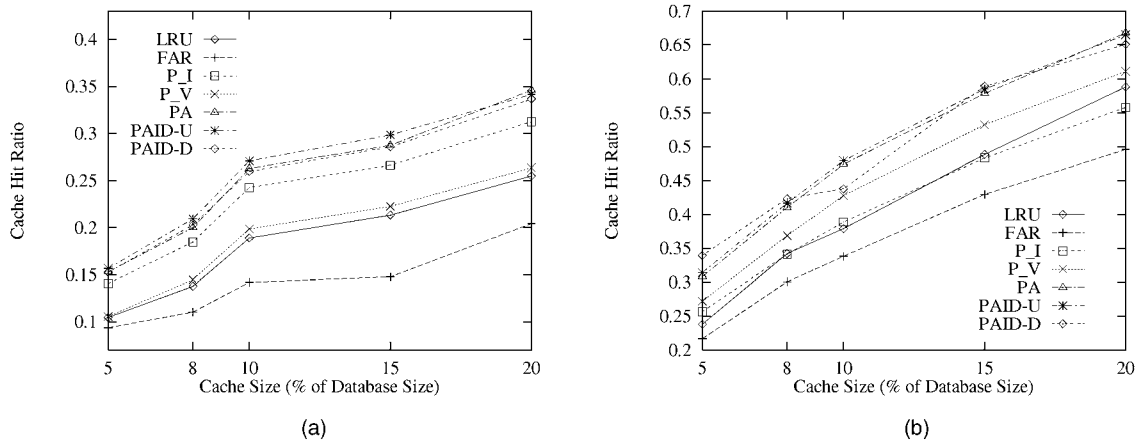


Fig. 10. Performance comparison of replacement schemes vs. cache size (skewed access). (a) Scope distribution 1. (b) Scope distribution 2.

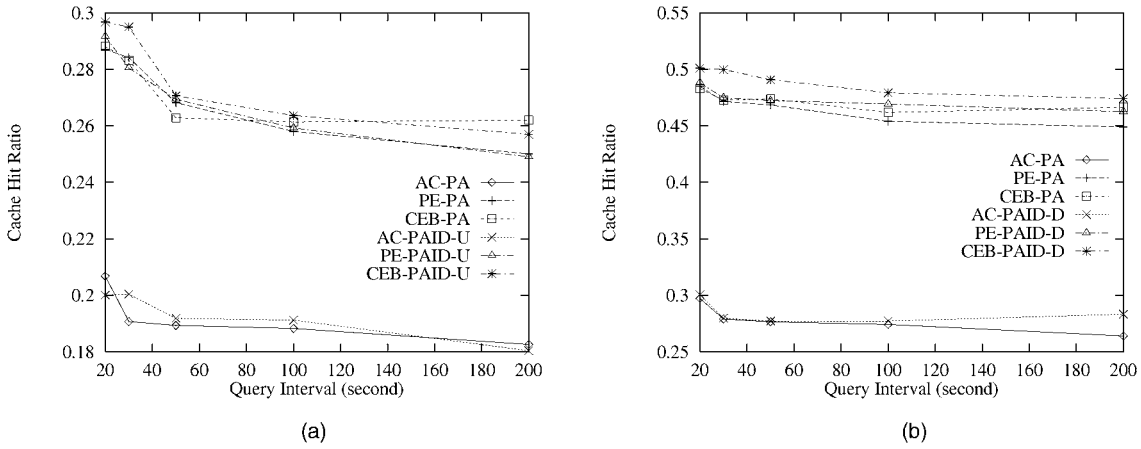


Fig. 11. Cache hit ratio of invalidation schemes combined with *PA* and PAID vs. query interval. (a) Scope distribution 1. (b) Scope distribution 2.

second scope distribution, results of the *PAID-U* and *PAID-D* schemes are presented in Fig. 11a and Fig. 11b, respectively. The parameters are set the same as in Fig. 3. Comparing Fig. 11 to Fig. 3, we can see that the relative performance of *CEB*, *PE*, and *AC* remains the same, i.e., *CEB* is the best, followed by *PE*, and *AC* is the worst. This indicates that the invalidation schemes are not sensitive to the underlying replacement policy. Similar results are obtained for other parameter settings.

8 Conclusion

In this paper, we have explored cache invalidation and replacement issues for location-dependent data under a geometric location model. For location-dependent invalidation, we presented two basic schemes (i.e., *PE* and *AC*) and a generic method, *CEB*. The *CEB* method, based on the proposed caching efficiency criterion, attempts to balance the overhead and the precision of the invalidation information when an approximation of a valid scope has to be decided. We also proposed two cache replacement policies, *PA* and *PAID*, that consider the valid scope area (for both methods) and data distance (for *PAID* only) and combine these factors with access probability.

A series of simulation experiments have been conducted to evaluate the performance of the proposed caching strategies. The results show that the *CEB* method, in a variety of system settings, obtains a better cache performance than *PE* and *AC*. This leads us to conclude that caching efficiency is an effective selection criterion for location-dependent invalidation information. We believe that *CEB* would achieve even better performance if we could develop better methods for generating candidate valid scopes. The proposed *PA* and *PAID* policies have demonstrated a substantial performance improvement over the existing *LRU*, *P_I*, *P_V*, and *FAR* policies. In particular, consideration of the valid scope area improves performance significantly in all settings. However, the factor of data distance is sensitive to scope distributions, query patterns, and movement models.

As for future work, we are planning to study the influence of data updates on location-dependent caching strategies based on the observation that it is not beneficial to attach invalidation information to data that are updated frequently. In addition, we have so far focused on item-based location-dependent queries. In our future research, we would like to explore caching and query processing problems associated with general location-dependent queries, such as "find the nearest hotel with a room rate below $100."

Acknowledgments

The authors would like to thank Dr. Siu-Wing CHENG for his valuable discussions and suggestions about the medial axis approach to find the inscribed circle of a polygon. This work was supported by grants from the Research Grant Council of Hong Kong under grant numbers HKUST-6241/00E and HKUST6079/01E.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communications Environments," *Proc. ACM SIGMOD Conf. Management of Data,* pp. 199-210, May 1995.

[2] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," *Proc. ACM SIGMOD Conf. Management of Data,* pp. 1-12, May 1994.

[3] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications,* chapter 7. New York: Springer-Verlag, 1996.

[4] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom 2000),* pp. 200-209, Aug. 2000.

[5] K. Cheverst, N. Davies, K. Mitchell, and A. Friday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom 2000),* pp. 20-31, Aug. 2000.

[6] S. Dar, M.J. Franklin, B.T. Jonsson, D. Srivatava, and M. Tan, "Semantic Data Caching and Replacement," *Proc. 22nd Int'l Conf. Very Large Data Bases (VLDB '96),* pp. 330-341, Sept. 1996.

[7] Spatial Datasets, http://dias.cti.gr/~ytheod/research/datasets/spatial.html, 2002.

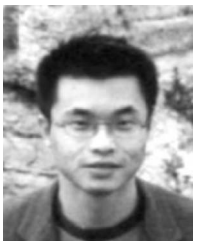
[8] M.H. Dunham and V. Kumar, "Location Dependent Data and Its Management in Mobile Databases," *Proc. Ninth Int'l Workshop Database and Expert Systems Applications,* pp. 414-419, Aug. 1998.

[9] I.A. Getting, "The Global Positioning System," *IEEE Spectrum,* vol. 30, no. 12, pp. 36-38, 43-47, Dec. 1993.

[10] T. Imielinski and B.R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Comm. ACM,* vol. 37, no. 10, pp. 18-28, 1994.

[11] J. Jing, A.K. Elmagarmid, A. Helal, and R. Alonso, "Bit-Sequences: A New Cache Invalidation Method in Mobile Environments," *ACM/Baltzer J. Mobile Networks and Applications (MONET),* vol. 2, no. 2, pp. 115-127, 1997.
[12] A. Kahol, S. Khurana, S.K.S. Gupta, and P.K. Srimani, "A Strategy to Manage Cache Consistency in a Distributed Mobile Wireless Environment," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 7, pp. 686-700, July 2001.
[13] S. Khanna and V. Liberatore, "On Broadcast Disk Paging," *SIAM J. Computing,* vol. 29, no. 5, pp. 1683-1702, 2000.
[14] J. O'Rourke, *Computational Geometry in C,* chapter 5. Univ. of Cambridge Press, 1994.
[15] Q. Ren and M.H. Dunham, "Using Semantic Caching to Manage Location Dependent Data in Mobile Computing," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom 2000),* pp. 210-221, Aug. 2000.
[16] H. Schwetman, *CSIM User's Guide (Version 18).* MCC Corp., http://www.mesquite.com, 1998.
[17] R. Seidel, "A Randomized Planar Point Location Structure," ftp://ftp.geom.umn.edu/pub/contrib/comp_geom, 2002.
[18] J.R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Proc. First Workshop Applied Computational Geometry,* pp. 124-133, May 1996.
[19] K.L. Tan, J. Cai, and B.C. Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 8, pp. 789-807, Aug. 2001.
[20] J. Xu, Q.L. Hu, W.-C. Lee, and D.L. Lee, "An Optimal Cache Replacement Policy for Wireless Data Dissemination under Cache Consistency," *Proc. 30th Int'l Conf. Parallel Processing (ICPP '01),* pp. 267-274, Sept. 2001.
[21] J. Xu and D.L. Lee, "Querying Location-Dependent Data in Wireless Cellular Environments," *Proc. WAP Forum/W3C Workshop Position Dependent Information Services,* Feb. 2000.
[22] J. Xu, X. Tang, and D.L. Lee, "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* to appear.
[23] J. Xu, X. Tang, D.L. Lee, and Q.L. Hu, "Cache Coherency in Location-Dependent Information Services for Mobile Environments," *Proc. First Int'l Conf. Mobile Data Access (MDA '99),* pp. 182-193, Dec. 1999.
[24] B. Zheng and D.L. Lee, "Processing Location-Dependent Queries in a Multi-Cell Wireless Environment," *Proc. Second ACM Int'l Workshop Data Eng. for Wireless and Mobile Access (MobiDE '01),* pp. 54-65, May 2001.
[25] B. Zheng and D.L. Lee, "Semantic Caching in Location-Dependent Query Processing," *Proc. Seventh Int'l Symp. Spatial and Temporal Databases (SSTD '01),* pp. 97-116, July 2001.
[26] G.K. Zipf, *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, 1949.

Baihua Zheng received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1999. She is currently a PhD candidate in the Department of Computer Science at the Hong Kong University of Science and Technology. Her research interests include mobile computing and spatial databases.



Jianliang Xu received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998. He is currently a PhD candidate in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include mobile computing, wireless networks, Internet technologies, and distributed systems. He is a student member of the IEEE.

Dik Lun Lee received the MS and PhD degrees in computer science from the University of Toronto in 1981 and 1985, respectively. He is a professor in the Department of Computer Science at the Hong Kong University of Science and Technology and was an associate professor in the Department of Computer and Information Science at Ohio State University, Columbus. He has served as a guest editor for several special issues on database-related topics and as a program committee member and chair for numerous international conferences. He was the founding conference chair for the International Conference on Mobile Data Management. His research interests include document retrieval and management, discovery, management and integration of information resources on the Internet, and mobile and pervasive computing. He was the chairman of the ACM Hong Kong Chapter.