

Semantic Caching for Multiresolution Spatial Query Processing in Mobile Environments

Sai Sun, Xiaofang Zhou, and Heng Tao Shen

School of Information Technology and Electrical Engineering
The University of Queensland, Australia
{sunsai, zxf, shenht}@itee.uq.edu.au

Abstract. Spatial data are particularly useful in mobile environments. However, due to the low bandwidth of most wireless networks, developing large spatial database applications becomes a challenging process. In this paper, we provide the first attempt to combine two important techniques, multiresolution spatial data structure and semantic caching, towards efficient spatial query processing in mobile environments. Based on the study of the characteristics of multiresolution spatial data (MSD) and multiresolution spatial query, we propose a new semantic caching model called Multiresolution Semantic Caching (MSC) for caching MSD in mobile environments. MSC enriches the traditional three-category query processing in semantic cache to five categories, thus improving the performance in three ways: 1) a reduction in the amount and complexity of the remainder queries; 2) the redundant transmission of spatial data already residing in a cache is avoided; 3) a provision for satisfactory answers before 100% query results have been transmitted to the client side. Our extensive experiments on a very large and complex real spatial database show that MSC outperforms the traditional semantic caching models significantly.

1 Introduction

Spatial data have been increasingly used in mobile environments such as electronic navigation, dynamic map generalization, and location-dependent applications. This raises many new research challenges and opportunities. Compared to traditional distributed systems, mobile computing environments have distinctive characteristics which impact on the manipulation of spatial databases [2]. First, in mobile computing environments, there are usually restrictions such as a limited bandwidth, unstable wireless link and short-life battery power. All of these seriously conflict with the tremendous volume of spatial data. Second, the capabilities of mobile units vary greatly, each of which could be a powerful portable computer or a personal digital assistants (PDAs) with very small screens. Thus, users of different mobile units may require different qualities of query answers. Third, in mobile environments, spatial queries are generally not isolated to each other but lie in several typical patterns.

Semantic caching is an important method to improve the performance of mobile computing. This method maintains the semantic descriptions as well as

results of previous queries on the client side. Semantic caching is based on an assumption that queries are relevant to each other semantically. Interestingly, this matches the property of spatial queries in mobile environments, which will be discussed later in section 5.1. By caching pertinent and frequently queried data in mobile units, semantic caching can reduce network traffic, shorten response time and provide better scalability [10].

Generally speaking, there are two basic approaches to reduce the data size processed in spatial queries: (1) reducing the size of candidate sets; and (2) reducing the complexity of geometry objects. The filter-refinement strategy has been well-established for the former purpose, while the multiresolution method is designed for the latter purpose. Multiresolution databases break spatial objects into parts, and then fetch only those parts that may contribute to a particular resolution [13]. Hence, the multiresolution method can lead to a smaller data transmission which is crucial in mobile environments. Furthermore, it permits users to flexibly designate their required query qualities or parameters.

Therefore, combining multiresolution and semantic caching could greatly improve the performance of spatial queries in mobile environments. However, the structure and semantic description of multiresolution spatial data (MSD) is much more complex than standard data, which raises the issues in the caching mechanism, including caching granularity, query trimming strategy, cache replacement policy, etc. To the best of our knowledge, there has been no significant research in this direction to date.

In this paper, we deploy MSD in semantic cache by introducing Multiresolution Semantic Caching (MSC) to improve the performance of spatial queries in mobile environments. We analyze the characteristics of MSD and MSD queries and their impact on the semantic caching mechanism, then propose MSC inspired from the analysis. In traditional semantic caching, query processing divides the relationship between the query and the cache into three categories: 1) exact match - the query is fully answered by the cache; 2) partial match - the query is partially answered by the cache; and 3) no match - the query is not answerable from the cache. And different operations are designed toward different categories. Incorporating the characteristics of MSD, MSC enriches the relationship by two more categories: Assumable Exact Match and Approximate Match. The motivation is to provide the user acceptable query results as soon as possible. MSC improves the performance in three ways: 1) by reducing the amount and complexity of the remainder queries, and further reducing the retrieval time in the database server; 2) by avoiding the redundant transmission of spatial data that have already existed in the client cache; and 3) by providing satisfactory answers before 100% of the results have been transmitted to the client. Our extensive experiment results confirm the effectiveness of our methods for spatial query processing in mobile environments.

The remainder of this paper is organized as follows. In Section 2, we discuss previous work on semantic caching and multiresolution. In Section 3, some preliminary work is presented, followed by MSC query processing in Section 4. An

extensive performance study is reported in Section 5. Finally, we conclude our paper in Section 6.

2 Related Work

One research area that is directly related to this paper is that of semantic caching, especially semantic caching in mobile environments. The other part of relevant work concerns multiresolution spatial data. In this section, we review previous works from these two fields.

2.1 Semantic Caching

The semantic caching technique was first proposed in [4] to contrast with page caching and tuple caching. The client maintains the results of previous queries as well as their semantic descriptions in its cache, which allows the tuples available locally not to be fetched from the database again. A value function based on semantic locality was also proposed in this paper, named as ‘Manhattan distance’. Later, this technique was widely studied in centralized systems, OLAP, LDAP, WWW, and heterogeneous systems [10]. Semantic caching in mobile computing environments has also received extensive attentions. [3] proposed a semantic caching scheme for handling both projection and selection in a mobile environment. [9] discussed how to use semantic caching to manage location dependent data in mobile computing and extended Manhattan Distance to a new replacement policy FAR (Furthest Away Replacement). In [14], DBMS was proposed in mobile units to manage caches. Recently, processing spatial queries has begun to occur in this field, such as [15], [6]. Because mobile clients are often roaming, [15] identified the problem of the validity of previous queries and proposed the corresponding algorithms to handle neighbour and window queries. In [6], a proactive caching model, which caches the result objects as well as their R-tree index nodes, was proposed to improve the utility of local caches. However, both spatial object and location information discussed in the above papers is concerned with **Point** data. It is also the reason why [6] thought that a new query could be answered only by the cached data of the same query type. Our work differs from these previous research in that we focus on **Polygon** data, for example we focus on how to cache MSD to dynamically generalize digital maps in mobile environments.

2.2 Multiresolution

The main goal of multiresolution (also called multiscale) systems is to be able to retrieve objects with different representations to satisfy different requests from users. As different details of a particular feature are distinguishable at different scales, a geometry object should be treated as a collection of spatial elements. In [1] multiple representations of spatial objects are pre-generated and stored in the database. Queries on different scale access different representations. This

technology is also known as ‘multirepresentation’. Its main disadvantage is the much higher storage overhead than normal. In addition, by using pre-computed representations with a large amount of redundant data, this scheme is not capable of supporting a desirable form of data caching. Later work by [8] focused on the concept of indexing objects by not only their spatial location, but also a scale value. With the scale in the index, I/O cost can be significantly reduced during retrieval. However, the approach is too simplistic and only supports graphical and content zoom. An improvement was achieved in [13], which assigns each vertex of an object to a set of map scales and vertices that share similar or equivalent scales are grouped into ‘vertex layers’. The vertices in each layer form a representation of the object at a particular scale. This idea is similar to our multiresolution data structure – ‘Bit-map’. However, they did not point out how to assign scales to vertices, which is a lengthy process if done manually. Later, they extended their work in [16], which proposed that the control on spatial generalization should be multi-dimensional with spatial resolution as one dimension and various types of generalisation style metrics as the other dimensions. However, they did not provide detailed algorithms or a performance study.

3 Preliminary Work

This section firstly introduces the multiresolution spatial data structure used in the remainder of the paper, then presents the constraint formula of the semantic region of MSC.

3.1 Multiresolution Spatial Databases

A spatial object O can be considered as a set of points: $\{p_1, p_2, \dots, p_n\}$, where if O is a point, $n = 1$; if O is a line, $n = 2$; if O is a simple polygon (without hole), $n \geq 3$ and each point pair (p_a, p_{a+1}) forms a line on the object boundary. In traditional spatial databases, the spatial elements (points) of the object are opaque to applications and have to be accessed together. That is why only one fixed resolution can be provided by such systems. A simple method to overcome this flaw is to store and process spatial objects based on points. However, point-based data structures create undesirable overheads due to the large amount of tuples and expensive calculation required to reconstruct representations. Hence, the multiresolution spatial data structure (MSDS) is proposed to produce constituent representations effectively. A proper MSDS in mobile environments should take the following four aspects into account.

Firstly, the MSDS should make it feasible to build representation-derivation into the query process of multiresolution systems to avoid time-consuming post retrieval. Thus, when processing a query, the multiresolution database system can choose a proper resolution level and retrieve only necessary data to satisfy this resolution.

Secondly, different representations derived from the same MSD should be consistent with each other. More precisely, spatial relationships should remain

the same or at least similar between different representations. Here, spatial relationships could be topological, directional or metrical.

Thirdly, different representations are convertible to each other. That is, low resolution representations can be derived from high resolution representations and high resolution representations can be produced by low resolution representations plus additional information. This characteristic is especially important for semantic caching as it can largely improve the utilization of cached data. For example, suppose that all polygons overlapping window w at resolution l_1 are cached and the most recent query is ‘to find all polygons overlapping window w at resolution l_2 ’ ($l_2 > l_1$), if representations are convertible, only the additional data between l_1 and l_2 need to be retrieved and transmitted to the mobile client. Otherwise, the data cached would not be useful in answering this query.

Finally, as the bandwidth is so scarce in mobile environments, MSD should be well compressed before transmission in mobile servers efficiently and decompressed in mobile clients.

3.2 MSDS – *Bit_map*

In the remainder of this paper, we use *Bit_map* as our MSDS. The basic idea of *Bit_map* is to group points based on scales, then abstract the data of every group to a tuple. This method avoids the opacity of organizing data based on polygon and the high overhead of organizing data based on points.

There are three important functions in *Bit_map* scheme.

One is *Scale Value Function*, which is used to group points. This function assigns a scale value to each point, denoted as $\Delta(p)$. Points holding the same value belong to the same resolution level. Thus, with scale value function, a spatial object is broken into disjointed parts: $O = \{O_n, O_{n+1}, \dots, O_m\}$, where $n = \min(\Delta(p))$ and $m = \max(\Delta(p))$ and $O_i = \{p | \Delta(p) = i\}$ ($n \leq i \leq m$). As n is the lowest scale of representations users may be interested in, it is named as *Base Level*.

The second function is *Abstract Compression Function*, which is used to abstract information from grouped points and compress it to a tuple. Using this function, we get $\{Abst(O_n), Abst(O_{n+1}), \dots, Abst(O_m)\}$. Here, $Abst(O_n)$ is different from other $Abst()$. As it includes the data equal to or less than *Base Level*, we denote it as $blData$. Whereas $Abst(O_i)$ ($n+1 \leq i \leq m$) only includes the additional information to refine O from scale $i-1$ to scale i , denoted as $aData_i$. Then, spatial object O is organized as $(ObjectID, BLData, AData, delta)$ in databases, where $delta$ denotes the scale level and each O includes $m - n + 1$ tuples as $(O_{id}, blData, \emptyset, n)$, $(O_{id}, \emptyset, aData_{n+1}, n+1)$, ..., $(O_{id}, \emptyset, aData_m, m)$.

The last function is *Reconstruction Function*. After retrieving data at specific resolution, $Recon()$ will reconstruct them to a specific representation of objects. In more details, when $l = n$, Reconstruction Function is $Recon(blData)$; When $l > n$, it is $Recon(blData, aData_{n+1}, \dots, aData_l)$.

Both Scale Value Function and Abstract Compression Function utilize the iterative decomposition property of Z-ordering. Interested readers could refer

to [11], [12] for more detailed algorithms. *Bit_map* defeats the overhead problem and expensive reconstruction in point-based data structures and avoids the accessing and processing of unnecessary data.

3.3 MSC Region

A semantic cache is a collection of semantic regions, each of which groups together semantically related tuples and maintains the semantic description of these tuples. As semantic regions are created dynamically based on the queries submitted at the client [4], we need to define multiresolution spatial queries before discussing the semantic description of semantic regions. (All definitions in our model are based on *Bit_map*, however these definitions can also be easily extended to other multiresolution data structures). In this paper, our work only focuses on the multiresolution window query, the most common query in multiresolution spatial databases. Processing more complex multiresolution spatial queries with the framework of semantic caching is a potential direction for future research.

Definition 1. (*Multiresolution Window Query MWQ*) Given a MSD relation $R(\text{ObjectID}, \text{BLData}, \text{AData}, \text{delta})$ and a window W , a MWQ finds all objects O having at least one point in common with W at resolution r from R ,

$$MWQ(W, r) = \{Re(O, r) | Re(W, r) \cap Re(O, r) \neq \emptyset\},$$

where $Re(O, r)$ is the representation of O at resolution r and r should not be less than the base level n .

Multiresolution window query is a window query with resolution constraints. For the sake of describing semantic region, now we extend the concept of MWQ to be more general.

Definition 2. (*Part Multiresolution Window Query PMWQ*) Given a window W and two resolution level $r1$ and $r2$, where $n \leq r1 \leq r2 \leq m$, a PMWQ finds all objects O satisfying $MWQ(W, r2)$ and only fetch their data between resolution $r1$ and resolution $r2$.

Part multiresolution window query denotes ‘select $\text{ObjectID}, \text{blData}, \text{aData}, \text{delta}$ from R where $Re(O, r2) \in MWQ(W, r2)$, $\text{delta} \geq r1$ and $\text{delta} \leq r2$ ’. Obviously, MWQ is a special kind of PMWQ when $r1 = n$. The conception of PMWQ is important in MSC as it records the result of MSC query trimming. Further, MSC region can be described as $S(W, r1, r2)$, a cached PMWQ query. And the whole cache can be described as $\sum S(W, r1, r2)$

4 MSC Query Processing

In this section, we initially analyze the characteristics of MSD and MWQ. Approaches incorporating these characteristics are then proposed to improve the performance of query processing.

4.1 Caching MSD vs. Caching Traditional Multi-dimensional Data

As mentioned in Section 2.1, there are some related works about caching standard data and point data in previous research. Then, what are the differences between caching MSD and caching those data? Specious answer is that MSD is multi-dimensional data. But point data are also 2D and queries are frequently multi-dimensional in OLAP systems as each of which covers more than one table and one attribute [5].

Then, what are the essential differences between caching MSD and caching traditional multi-dimensional data? The obvious differences of MSD from traditional data include large size, complex structures and operations etc., which have been extensively studied. Now we will analyze the characteristics of MSD which can be utilized to improve query performance.

One characteristic is that spatial objects have *extensions*. A spatial object may intersect two or more disjointed windows at the same time. Hence, the results of two disjointed window queries may overlap and caching these results may cause redundancy. The characteristic also causes another situation. Figure 1 (a) shows a window query and its result; figure 1 (b) is the results of two other small window queries. We can see that although the windows of the two small queries do not wholly cover the window of the first large query, their data can still fully answer it. That is because the window query finds all polygons having at least **one point** in common with the window and there is no polygon fully contained in the shadow area. Thus, given a window query and a cache, there are two kinds of full answer. One is **Full Window Answer**, which is that the area of cache fully covers the window of the query. Another is **Full Data Answer**, which is that the data set of cache fully contains the result of the query. A Full Window Answer must be a Full Data Answer, but the reverse is not tenable. As the constraint formula of MSC region is $(W, r1, r2)$, query trimming can only find out Full Window Answer. Finding the Full Data Answer is a key factor in improving the performance.

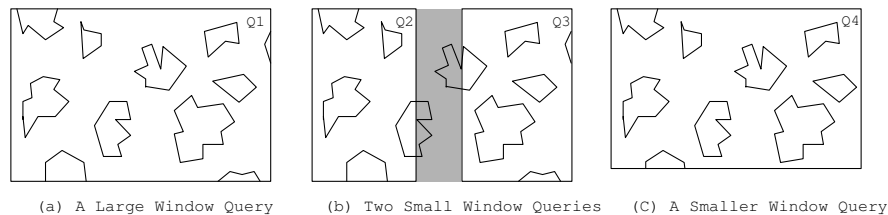


Fig. 1. The Extension of Spatial Objects

Another characteristic is that *resolution* is not a normal axial dimension where $[a, b] \cap [c, d] = \emptyset$ ($a < b < c < d$). To different resolutions, MSD at b includes MSD at a and MSD at a has a general feeling of MSD at b . In particular,

because the data distribution on resolution is nonlinear, the difference between two consecutive high resolutions is not apparent.

Finally, as all spatial data are the digitized approximations of real world objects, *spatial queries can not be well specified*. For example, although the query window in figure 1 (c) is a bit smaller than that in figure 1(a), it does not matter greatly to most users. The other example is that users who querying resolution 18 usually can still be satisfied with a map with hybrid resolution (most area at resolution 18 and few places at resolution 17). Hence, the data do not fully answer a query, but it can satisfy the user who submitted the query. We name this case as **Approximate Satisfaction**.

As most spatial queries are not isolated to each other, **Full Data Answer** and **Approximate Satisfaction** occur quite often in MSD cache. Taking them into account when processing MWQ can reduce the amount of remainder queries posted to the database and response to the users before full answers transmitted to the client.

In the following, we investigate how to process a MWQ in MSC. We introduce the method of trimming a query with a single MSC region first, followed by extension to process a query with the whole MSC cache. We finally discuss how to retrieve the remainder query, transmit the retrieval result and coalesce the probe query and the remainder query.

4.2 Trimming a PMWQ with a MSC Region

Here, we discuss trimming PMWQ (not MWQ) with a MSC region. Given a $PMWQ(W_q, r1_q, r2_q)$ and a MSC region $S(W_s, r1_s, r2_s)$, no intersection is the simplest case, which happens when $W_q \cap W_s = \emptyset$ or $r1_q > r2_s$ or $r2_q < r1_s$. Otherwise, the PMWQ intersects the region ($W_q \cap W_s \neq \emptyset$ and $[r1_q, r2_q] \cap [r1_s, r2_s] \neq \emptyset$). In the following, we discuss how to perform trimming according to resolution ranges in detail.

Case 1: $r1_q = r1_s$ and $r2_q = r2_s$

Because $PMWQ$ and S have the same resolution range, the trimming is simplified to the clipping between two windows W_q and W_s . The remainder query may be none or single or more than one sub queries. For example, in figure 2(a), the remainder query is $(W_3, r1_q, r2_q)$; in figure 2(b), the remainder query consists of two sub queries $(W_4, r1_q, r2_q)$ and $(W_5, r1_q, r2_q)$; in figure 2(c), the window of S contains the window of $PMWQ$, no remainder query needs to be submitted to the server. However the region S will be clipped into five sub-regions.

Case 2: $r1_q \leq r1_s$ and $r2_q \geq r2_s$

In this case, the resolution range of $PMWQ$ contains the resolution range of S . Thus, the $PMSQ$ will be split into two parts $W_q \cap W_s$ and $W_q \cap \neg W_s$ first. Then, $(W_q \cap W_s, r1_q, r2_q)$ will be trimmed into three parts with disjointed resolution ranges. They are $[r1_q, r1_s)$, $[r1_s, r2_s]$ and $(r2_s, r2_q]$. Notice if the query

is split by resolution first, then by window, it will cause more sub remainder queries. Figure 3(a) is a $PMWQ$ and a S fallen into case 2. After trimming, S will be simply split into 4 sub-regions. To $PMWQ$, piece 6 is the probe query, piece 5, 7, 8 together compose the remainder query.

Case 3: $r1_q \geq r1_s$ and $r2_q \leq r2_s$

In this case, the resolution range of $PMWQ$ is contained by the resolution range of S . In contrary to case 2, S will be split by window before resolution whereas $PMWQ$ is just split by window. Figure 4 is an example.

Case 4: $(r1_q \geq r1_s$ and $r2_q \geq r2_s)$ or $(r1_q \leq r1_s$ and $r2_q \leq r2_s)$

In this case, the resolution range of $PMWQ$ intersects the resolution range of S . Thus, both $PMWQ$ and S need to be split by resolution and window. Figure 5 gives an example.

After trimming a $PMWQ$ with a MSC region S , the $PMWQ$ will be divided into two parts – the probe query and the remainder query; and the S will also be divided into two parts – the intersection part and the difference part. The intersection part and the probe query are in fact the same.

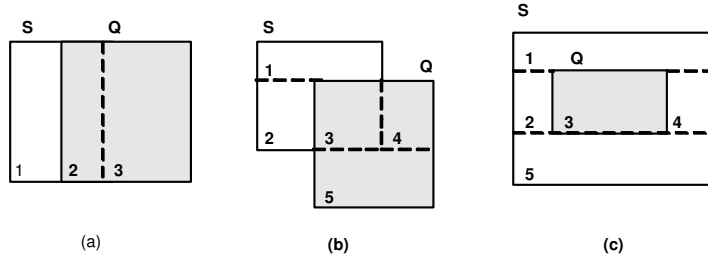


Fig. 2. Examples of Case 1

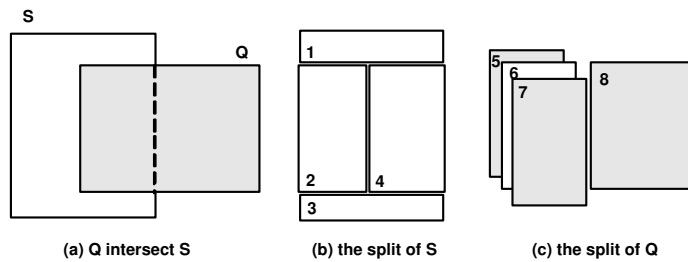


Fig. 3. Example of Case 2

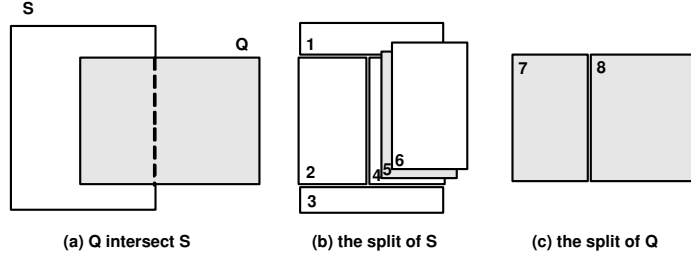


Fig. 4. Example of Case 3

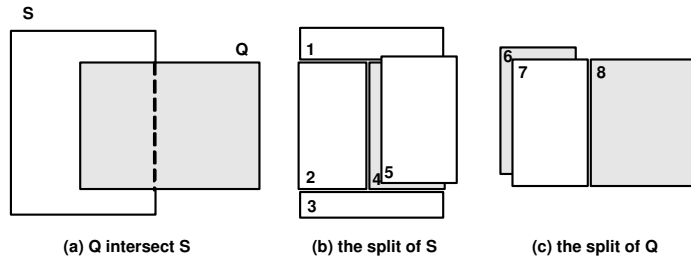


Fig. 5. Example of Case 4

4.3 Processing a MWQ with a MSC Cache

As the query usually intersects more than one region in a MSC cache, processing a query with a MSC cache includes two steps.

1. The input query will be trimmed with all cache regions. This step produces three outputs. One is the probe query set Q_p ; one is the remainder query set Q_r ; and the last one is the difference of cache from the query, denoted as C_d . Q_p and Q_r are two sets of sub-queries, while C_d is a set of sub-regions. The cost of step 1 is usually linear in the number of regions [5].
2. According to the outputs of step 1, the query will be classified into a category and the corresponding process will be handled according to the category.

Traditional query processing classifies the relationship between a query and a cache into three categories: No Match, Partial Match and Exact Match. In order to improve the system performance, MSC enriches them into five categories – **No Match**, **Exact Match**, **Assumable Exact Match**, **Approximate Match** and **Partial Match**. Now, we will discuss each category in detail.

Category 1: No Match

It is the simplest category and happens only when $Q_p = \emptyset$. In this case, the original query will be posted to the server to get the result.

Category 2: Exact Match

In our paper, the cache exactly matches a query if and only if $Q_r = \emptyset$, which means that the cache is a Full Window Answer to the query. In this case, the result of Q_p is the query result.

Category 3: Assumable Exact Match

As discussed in section 4.1, Full Data Answer is also a kind of full answer to a query. However, due to the limitation of the constraint formula of region, it can not be figured out by query trimming directly. We propose **Assumable Exact Match** to solve this problem.

Before explaining Assumable Exact Match, **Negligible Window** needs to be introduced. According to the analysis in section 4.1, when a window is so small that it does not contain any whole polygon, the window can be ignored in point of data set because all polygons intersect this window also intersect its neighboring windows. Thus, we can assume if the width or height of a window is less than the average width or height of polygons in the data space, it is a negligible window. To avoid the situations that the user is focusing on this small window, the width and height of a negligible window should be less than the width and height of the original query.

Definition 3. (*Negligible Window*) Given a MWQ (W_q, r_q) and a data space R , a window W_i is a negligible window if and only if

$$(\text{Width}(W_i) < \lambda * \text{Avg}W \text{ and } \text{Width}(W_i) < \text{Width}(W_q)) \text{ or } (\text{Height}(W_i) < \lambda * \text{Avg}H \text{ and } \text{Height}(W_i) < \text{Height}(W_q)),$$

where $\text{Width}()$ and $\text{Height}()$ are two functions to get the width and height of a spatial object respectively; $\text{Avg}(W)$ and $\text{Avg}(H)$ represent the average width and the average height of spatial objects in the data space R respectively; and λ is a small number to adjust the confidence of negligibility.

If the window of each sub query in the remainder query set Q_r is a Negligible Window, we classify the query into Assumable Exact Match. In other words, we assume the query is fully data answered by local data. To answer Assumable Exact Match, the local data make up the query result; Q_r does not need to be posted to the server.

Category 4: Approximate Match

Approximate Match happens when the data in local cache do not fully answer a query, but it still satisfies the user who submits it. To handle Approximate Match, the result of the probe query is represented to the user first. Then the remainder query is posted to the server. As it is not so urgent as normal remainder queries, it is assigned with a lower priority. By the chance that users are not satisfied with Approximate Match answer, they can choose to submit a finer query or to wait for the answer of the remainder query.

As a multiresolution window query is sensitive to two factors (window and resolution), both aspects should be taken into account when deciding whether

a query falls into Approximate Match category. We use Algorithm 1 to make the judgement, in which λ_1 and λ_2 are used to adjust the confidence of approximation. Our experiments prove that when $\lambda_1 < 0.1$ and $\lambda_2 < 0.2$, Approximate Match can insure the quality of results.

Algorithm 1 Judge_Approximate_Match(Q_r) // Q_r is the remainder query set

```

 $Q_{window} = \emptyset$  {initialize two query sets}
 $Q_{resolution} = \emptyset$ 
while  $Q_r \neq \emptyset$  do
     $q \leftarrow$  any sub-query in  $Q_r$ 
    if  $q.r2 = q.r1 + 1$  then
         $Q_{resolution} \leftarrow q$ 
    else
         $Q_{window} \leftarrow q$ 
    end if
end while
 $S_W = Sum\_Area(Q_{window})$  {Sum up the area of queries in  $Q_{window}$ }
 $S_R = Sum\_Area(Q_{resolution})$  {Sum up the area of queries in  $Q_{resolution}$ }
if  $(S_W < \lambda_1) \& (S_R < \lambda_2)$  then
    Approximate Match
else
    No Approximate Match
end if

```

Category 5: Partial Match

Partial Match happens when the query does not fall into any of the above four categories. In this case, the remainder query set is posted to the server and its result, together with local data, answers the user's requirement. It is worthy of note that during the transmission of the remainder query result, data in client side may become Approximate Match or Assumable Exact Match to the query. The result will be shown before the completion of the transmission.

4.4 Retrieval, Transmission and Coalescence

When the database receives the remainder query set, it retrieves these sub queries according to their window sizes and resolution ranges. The query with a larger window will be executed earlier. And the data at a lower resolution will be accessed before the data at a higher resolution. Transmission also obeys the same rule. The motivation is to access and transmit minimum data to achieve Assumable Exact Match or Approximate Match in the client side. When all data arrive at the client, they will be organized together with local data to coalesce a new MSC region.

5 Performance Evaluation

In this section, we examine the performance of our MSC query processing scheme through a simulation study. Before presenting the experiment results we first introduce the simulation model.

5.1 Spatial Query Patterns in Mobile Environments

Spatial queries in mobile environments are usually not isolated to each other but lie in typical patterns as shown in the following examples.

Example 1.1 Suppose Mike is a new international student of the University of Queensland and he wants to make a travel plan of Australia. His typical search process will be querying the place near his university first, then extending his search area to the whole Australia gradually. When he finds an interesting site, he may inquire more details of this spot and inquire about areas around it.

The above example represents a typical spatial pattern, named ‘Zoom in/Zoom out Pattern’ in this paper. Users focus on a small area first, then they extend their search area. At the same time, they may reduce the requirement of precision to adapt to the limit resource in mobile environments. When they find their interesting place, they will refine the precision and pan around the place. Queries belonging to ‘Zoom in/Zoom out Pattern’ usually jump in window sizes and resolutions. Figure 6(a) gives an illustration of such queries, where the number represents the order of query and the color represents its resolution. The darker a window is, the more details it queries.

Example 1.2 Suppose Mike chooses a scenic spot and he is travelling there now. Because he is not familiar with this place, he queries the map around his location periodically.

It is an example of another typical spatial query pattern. We name it ‘Travel Pattern’. In this pattern, a user requires dynamic map generalisation according to his location. It is a special kind of location-dependent queries, which is defined as queries with constraints on the location of the mobile unit users [7]. Queries belonging to ‘Travel Pattern’ usually focus on small windows and high resolutions. And the resolutions of different queries are similar. Figure 6(b) is an example, in which the lines represent the route of a moving object.

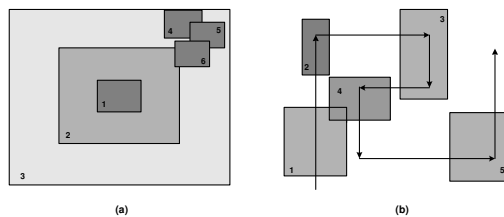


Fig. 6. Two basic query patterns.

‘Zoom in/Zoom out Pattern’ and ‘Travel Pattern’ are two typical patterns of spatial queries in mobile environments. Most spatial queries lie in one of the two patterns or the union of these two patterns, which is named as ‘Hybrid Pattern’ in this paper.

5.2 Workload Specification

Based on the three typical spatial query patterns, we develop three different workloads. And we assume the query issuing interval is 1 minute.

- **Workload 1 – Zoom in/Zoom out:** This workload further includes three types. a) Short Zoom in/Zoom out. This kind of query group has 3-5 queries and only zooms on one direction. b) Middle Zoom in/Zoom out. This kind of query group has 6-8 queries and pans after zoom in or zoom out. c) Long Zoom in/Zoom out. This kind of query group has 10 - 14 queries, which includes zoom in, zoom out and panning at the same time. The query window size is random from twenty times the average size of polygons to one thirtieth of the whole data space.
- **Workload 2 – Travel:** This workload also includes two types. a) Directed Movement. This kind of query group has 15-25 queries. The mobile client keeps moving in one direction with a random speed. b) Random Movement. This kind of query group has 30-50 queries. The mobile unit randomly chooses a destination and moves to it with random speed. After arrival, it randomly chooses the next destination again. According to our daily life, both two kinds of movement should be limited in a range. In our experiment, the range varies between 0.5 percent to five percent of the whole data space.
- **Workload 3 – Hybrid:** Randomly create a kind of query group in workload 1 or workload 2.

In our experiments, 100 groups of queries are randomly generated for each workload and the workload 1 has 863 queries totally, workload 2 and workload 3 have 3,040 and 2,874 queries respectively.

5.3 Simulation Model

This simulation emulates a mobile client issuing multiresolution window queries to a single server. Table 1 shows the main parameters of the model.

We use a large-scale data set, Real Estates Layer, provided by the Environmental Protection Agent, Queensland, Australia. This data set contains 398,464 polygons which are composed of 39,770,682 points. The simplest polygon has 4 points whereas the complex one has 275,531 points. For the experiment convenience, we cast away the polygons having less than 15 points or more than 10,000 points. Our experiment data set contains 309,868 polygons which are composed of 32,579,926 points. The average number of points of a polygon is 105. As the storage of the data set is 455.125 MB in Bit_map format, we choose 4MB as the cache size, around 1% of the total data set size. The client has a 19.2 Kbps wireless channel, which is standard for the mobile network.

Table 1. Model Parameters and Default Settings

Parameter	Description	Value
ServerFrequency	server clock frequency(MHz)	2800
ClientMips	mobile client clock frequency(MHz)	996
DataSetSize	the size of data set (MB)	455.125
ClientCache	client side cache size (MB)	4
BandWidth	wireless network bandwidth (Kbps)	19.2

5.4 Experiments and Results

The experiments are designed for two objectives. First, we studied the effect of adding Assumable Exact Match to the traditional query processing. Then, we compared our MSC model with the traditional cache scheme. The primary metric used was the *response time*, which was the elapsed time from MWQ being issued till an *acceptable query result* presented in the mobile client. This is a fairer metric than the response time for all data transmitted to the client, as users may be satisfied with the approximate match. Other metrics, such as the amount of queries in the remainder query set and the size of data transmitted, were also used.

Assumable Exact Match We first studied the impact of Assumable Exact Match to traditional schemes. We choose the amount of queries in the remainder query set, the accuracy of the query result and the size of transmission data as performance metrics. Our experiment results are shown in figure 7, where x -axis represents the value of λ , to illustrate the result more visually, y -axis is the ratio of two schemes in the above three metrics respectively. In other words, $\lambda = 0$ represents the traditional scheme. As it is the baseline of our comparison, it is always 100% in y -axis. Figure 7 (b) shows the accuracy of the query result. Because Assumable Exact Match ignores Negligible Windows in the remainder query set, it might miss some query results. With the increase in λ , we can see that the query accuracy continues decreasing because the restriction of Negligible Window is looser correspondingly. However, as this is insignificant even in the worse case with a large λ value at 4.0, the lowest accuracy of the three patterns is still higher than 93%. It proves that Assumable Exact Match is reliable on the query accuracy. An interesting phenomena is that the accuracy drops slightly with lower λ whilst it drops faster with higher λ . That is because the relationship between Negligible Window and λ is nonlinear. Figure 7 (c) plots the results in the ratio of the size of data transmitted. It is apparent that with the increasing of the λ , the size of data transmitted from the server is reduced. That is because Assumable Exact Match avoids redundant transmission of data that have existed in the client cache. The ratio in the amount of remainder queries with corresponding λ is plotted in figure 7 (a). Obviously, the curve plunges with a great downward trend at lower λ , while when λ approaches 4 the amount almost does not decrease. Synthesizing these three figures together, we

can determine an optimal λ value according to the users' requirements. When $\lambda = 1.2$, the accuracy of query results is higher than 99%, which is generally acceptable for uses, and the number of remainder sub-queries is reduced more than 35%. This is close to our assumption that when the width or height of a window is less than the average width or height of polygons in the data space, it can be ignored without affecting the accuracy of query results. The λ value may be affected slightly by the distribution, density and size of polygons. But it is likely close to 1. In the following experiments, we set $\lambda = 1.2$.

Comparing those curves for three different query patterns, we can see that the influences of our method on travel queries are greater than that of Zoom in/Zoom out queries. This is because of the essential differences between patterns. Compared to queries in Zoom in/Zoom out Pattern, those in Travel Pattern are generally smaller in window size and higher at resolution, which more easily produces Negligible Windows.

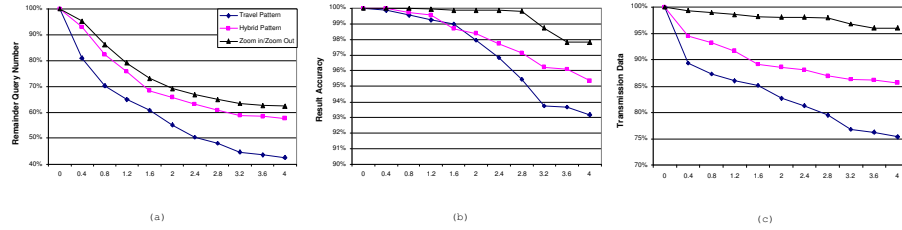


Fig. 7. Assumable Exact Match vs. λ

Five Categories vs. Three Categories Besides Assumable Exact Match, Approximate Match also affects the performance of query processing. In this experiment, we compare the performance of three categories with five categories. Figure 8 (a) demonstrates the average response time of three different query patterns based on three categories and five categories. It can be seen that the response time of five categories is always less than that of three categories despite of the query patterns used. However the reduction in response time of five categories is greater for travel queries, which is only 73% of that of three categories. Whilst, the improvement of Zoom in/Zoom out pattern is only 9%. Moreover, the average response time of queries in Zoom in/Zoom out pattern is 43 seconds, which is much more than that of queries in Travel pattern with only 14 seconds. That is because in Zoom in/Zoom out patterns, queries are usually executed to explore interesting areas, the average data retrieved by each query is much more than that in Travel pattern. Figure 8 (b) and (c) display the average transmission time and retrieval time. We can see that the data transmission time is far more than retrieval time which implies that data transmission is the dominant factor in wireless environments (Note the different scales of y axis). As we adopt

the low transmission bandwidth, 19.2 Kbps (the standard bandwidth of mobile environments), the average response time of less than 1 minute is considered to be acceptable to users.

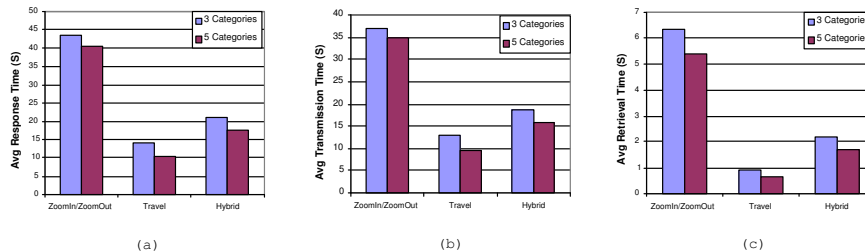


Fig. 8. Comparing the performance between 3 Categories and 5 Categories

6 Conclusion

In this paper, we have analyzed the characteristics of multiresolution spatial data and multiresolution window queries in mobile environments and proposed a new approach to improve the performance of query processing in our MSC model based on their characteristics. We refined the three categories in traditional schemes to five detailed ones and propose new processing methods against each of them. Our experiments, that were undertaken on a large real spatial data set show that the application of Assumable Exact Match can greatly reduce the amount of remainder sub-queries and the size of data transmission with very little sacrifice on the query result quality. Together with Approximate Match, it can reduce the response time by 27%. Our future work will focus on two parts: one is to extend this approach to more complex spatial queries, the other is to study the replacement policies.

Acknowledgment: The work reported in this paper has been partially supported by grant DP0345710 from the Australian Research Council.

References

1. A.U.Frank and S. Timpf. Multiple representations for cartographical objects in a multi-scale tree — an intelligent graphical zoom. *Computers and Graphics*, 18(6):823–929, 1994.
2. D. Barbara. Mobile computing and databases — a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, January/February 1999.
3. Ken. C.K.Lee, H.V.Leong, and Antonio Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):108–117, April 1999.

4. S. Dar, M.J. Franklin, B.T. Jonsson, D. Srivatava, and M. Tan. Semantic data caching and replacement. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, Bombay, India, Sept 1996.
5. P. M. Deshpande, K. Ramasamy, A. Shukla, and J. F. Naughton. Caching multi-dimensional queries using chunks. In *Proceeding of the 1998 ACM Conference on Management of Data*, pages 259–270, Seattle, USA, June 1998.
6. H. Hu, J. Xu, W. S. Wong, B. Zheng, D. L. Lee, and W.-C. Lee. Proactive caching for spatial queries in mobile environments. In *Proceedings of the 21th IEEE Int. Conf. on Data Engineering*, Tokyo, Japan, April 2005.
7. T. Imielinske and B.R. Badrinath. Querying in highly mobile and distributed environments. In *Proceedings of 18th International Conference Very Large DataBases*, Vancouver, B.C., Canada, Aug 1992.
8. M.Horhammer and M.Freeston. Spatial indexing with a scale dimension. In *Proceeding of the 6th International Symposium on Large Spatial databases*, Hong Kong, China, July 1999.
9. Q.Ren and M.H.Dunham. Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, MA, USA, August 2000.
10. Q. Ren, M. H. Dunham, and V. Kumar. Semantic caching and query processing. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):192–210, January/February 2003.
11. S. Sun, S. Prasher, and X. Zhou. A scaleless data model for direct and progressive spatial query processing. In *Proceedings of ER2004 Workshop on Conceptual Modeling for GIS*, Shanghai, China, Nov 2004.
12. S. Sun and X. Zhou. Semantic caching for web-based spatial applications. In *Proceeding of the 7th Asia Pacific Web Conference*, Shanghai, China, March 2005.
13. S.Zhou and C.B.Jones. Design and implementation of multi-scale databases. In *7th International Symposium on Spatial and Temporal Databases*, Los Angeles, CA, July 2001.
14. J. F. Yao and M. H. Dunham. Caching management of mobile dbms. *Journal of Integrated Computer-Aided Engineering*, 8(2), April 2001.
15. J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In *Proceeding of the 18th ACM SIGMOD Conference*, San Diego, California, USA, June 2003.
16. S. Zhou and C. B. Jones. A multi-representation spatial data model. In *Proceeding of the 8th International Symposium on Spatial and Temporal Databases*, Santorini Island, Greece, July 2003.