Alternative Strategies for Performing Spatial Joins on Web Sources

Cyrus Shahabi[†], Mohammad R. Kolahdouzan[†] and Maytham Safar[‡]

[†] Department of Computer Science, University of Southern California, Los Angeles, CA, 90089;
[‡] Department of Electrical & Computer Engineering, Kuwait University

Abstract. With the current information explosion on the Web, numerous applications require access to a collection of different but related pieces of distributed geospatial data. In this paper, we focus on one set of such applications that requires efficient support of spatial operations (specifically, spatial join) on distributed non-database sources. The main challenge with this environment is that remote sources are usually read-only and/or do not support spatial queries. Moreover, several of these Web-based applications can tolerate either some level of inaccuracy or progressively filtered (or polished) results. Therefore, conventional distributed spatial join strategies are not applicable or efficient in this environment. To address these challenges, we first break down the process of distributed spatial join operation into three steps: 1) local to remote transfer, 2) remote spatial selection, and 3) local refinement. Then, for each step, we propose and study alternative techniques and by varying their combinations, we generate several query plans. Each plan strives to strike a compromise between efficiency and accuracy. Since the techniques proposed for the first step have significant impact on the overall performance of the query, we specially focus our attention on this step. We propose two heuristics for the first step to reduce either the number of selection queries or the area covered by each selection query. Within a realistic experimental setup, we show that one heuristic is more appropriate with fast networks and a powerful local server, while the other one is superior in the opposite situation. Our experiments also show that both heuristics outperform approaches based on transmitting either the actual spatial objects or their bounding boxes. Note that the intention of this paper is not to propose a query optimizer to choose one plan over the others. Instead, it serves as a first step towards the design of such an optimizer by concentrating on the design and evaluation of several alternative plans within a realistic experimental setup.

Keywords: Spatial Join; Distributed Non-Database Sources; Query Plan

1. Introduction

Due to the recent proliferation of the World Wide Web, numerous data-intensive applications can obtain their required information from publicly available web sources. Many of these sources contain geospatial information. For example, detailed satellite images can be obtained from *www.terraserver.com*; maps from *www.mapquest.com*; airport information from *www.airbroker.se*; geolocated points of interest from *www.nima.mil*; international yellow pages that contain addresses of companies and government agencies from *www.infospace.com*; etc. The number of these types of sites, the quality, and detail of the information available are continually growing all around the globe. The challenge is how to efficiently and accurately integrate the related information retrieved from these different sources and present them to an end-user in a unified manner.

One of the requirements for this *Web-based geo-integration* is the efficient support of spatial operations on distributed non-database sources. For example, to support a query such as *"Find all the restaurants in the city of Torrance with more than two AMC theaters within their 2 mile distance"*, one need to efficiently support the *distance* spatial operation across three different information sources such as: 1) Yahoo movies for theaters' addresses, 2) Cuisine Net for restaurants' addresses, and 3) Etak Geocoder for address to coordinate transformations.

To support these distributed geo-spatial queries, we focus on an architecture where a Web mediator such as TSIMMIS (Hammer et al 1995), Information Manifold (Kirk et al 1995), The Internet Softbot (Etzioni et al 1994), InfoSleuth (Bayardo et al 1996), Infomaster (Genesereth et al 1997), DISCO (Thomasic et al 1997), HERMES (Adali et al 1997) or Ariadne (Knoblock et al 1998), is connected to both a local database system with spatial query capabilities (e.g., Informix Universal Server), and remote sources containing geospatial information. In this paper, we discuss and evaluate different ways of supporting distributed spatial queries within this architecture by conducting several experiments in a realistic setup.

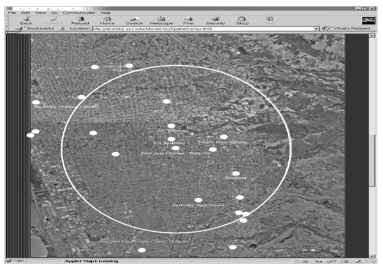
We focus on two types of spatial *join* queries. This is because spatial join is one of the most frequently used operations for combining spatial objects. It is also the most expensive operation, since it combines spatial objects of several relations, in contrast to window queries that can be based on a single relation. Spatial join is the operator that combines spatial objects from multiple data sets according to their geometric attributes, i.e., these attributes have to satisfy a spatial predicate (e.g., intersect or contain). The response set is a subset of the Cartesian product of two or more relations.

Numerous efficient algorithms for (distributed) spatial join have been proposed in the literature (Kim et al 1995, Bhowmicket al 1998, Bronkhoff et al 1996, Huang b et al 1997, Patel et al 1996, Papadias et al 1999, Koudas et al 1997, Aref et al 1996). However, there is one major distinction between our assumed web applications and traditional database applications; and one major difference between web architecture and the conventional architectures, which render the conventional approaches either not applicable or not efficient. Let us consider each distinction in turn.

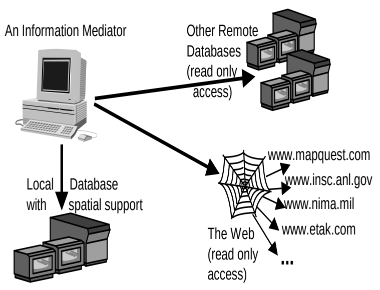
The users of web applications are more tolerant to inaccuracy and in that sense the applications are closer to information retrieval applications rather than traditional database applications. For these types of applications, it becomes more critical to obtain results faster as opposed to 100% accurate. We propose a series of approaches that take advantage of this fact in order to generate a superset of results very efficiently and then clean it up progressively.

Within the web architecture, the client does not have full control on the remote servers. In particular, it only has read access rights and cannot access the internal structures of a spatial index on the remote servers. In addition, it may need to read some of the spatial data into the local server at the time of query processing. Hence, it may not even assume the existence of a spatial index structure in the local site. While these restrictions render almost all of the approaches based on semi-joins and spatial-index joins (Arge et al 2000, Bronkhoff et al 1993, Bronkhoff et al 1996, Huang a et al 1997, Huang b et al 1997, Martynov 1995, Rotem 1991) impractical, it also results into three more challenges. First, the join operation should be translated to a series of spatial window queries (i.e., selections) on the remote site(s). Several of the conventional approaches (e.g., (Papadias et al 1995)) propose techniques to estimate polygon objects with bounding boxes (i.e., Minimum Bounding Rectangles, *MBR*). While performing selections based on MBRs in the remote site(s) decreases the complexity of the window queries, we show that a more effective approach is to reduce the *number* of window queries by *"merging"* the nearby polygons. Our proposed *Dynamic-MBR* technique with its *merge heuristics* are devised particularly to address this challenge. Second, the remote site may not have support for spatial queries. In this case, the spatial window query should be converted to a conjunction of regular predicates on coordinates of MBR's corners. Third, with the MBRs constructed to address either of the two above mentioned challenges, several false hits will be generated at the remote site. Since we do not have full access to the remote site, the false hits must be transfered back to the local site to be either displayed and/or eliminated after refinement. Hence, it seems like that the approach which results into fewer false hits will be the more efficient one considering the response time. However, we show that again the dominant factor is the remote processing time. Therefore, a grid based approach (such as *Fixed-BR*), which results in a better approximation of the polygons and hence less false hits, is not superior to Dynamic-MBR that is less accurate but results in less number of remote window queries.

In brief, our approach consists of three steps to perform a distributed spatial join operation. The first step is termed *Local to Remote Transfer*, in which representations of the query objects are computed and transferred to the remote site. Here, there is a trade-off between the complexity and the number of the transferred objects and the number of generated *false hits.* We propose Dynamic-MBR and one of the variations of Fixed-BR termed *Fixed-MBR*, as alternative data representations for local to remote data transfer and investigate their performances as compared to *No-MBR* and *Fixed-BR.* With No-MBR, the polygon representations of data objects are transferred. While with Fixed-BR, an approximation of objects by bounding rectangles resulted from a fixed partitioning of space is utilized. Finally, with Dynamic-MBR, dynamically generated minimum bounding rectangles of the objects are used as their approximate representations. We also propose two heuristics for Dynamic-MBR, *Minimum-CentroidDistance* and *Minimum-WastedArea*, that allow for merging of MBRs in order to reduce the number of transferred representations, while minimizing the number of *false hits* at the remote site. During the second step, termed *Spatial Selection*, a *window selection* operation is performed at the remote site to identify a set of candidate objects. The trade-off here is between the remote processing cost and the size of the data transferred back to the local site. Finally,



a. Restaurants, theaters and schools within 2 miles from an input point at Berkeley, CA


An Information Mediator
Other Remote Databases (read only access)
Local Database with spatial support
www.mapquest.com
www.insc.anl.gov
www.nima.mil
www.etak.com
...
The Web (read only access)

b. A mediator with a local spatial database

Fig. 1. Sample application and architecture

the third step is termed *Local Refinement*, in which a refinement operation is performed on the candidate objects at the local site to eliminate the false hits and produce the final set of results. We propose three alternative methods to execute the refinement step: *local spatial join, pipelined refinement,* and *statistical refinement* depending on whether the user is interested in a slow and exact or fast but approximate response to the query.

The remainder of this paper is organized as follows. In Section 2, we present overviews of some target applications that can benefit from Web-based geo-integration, the assumed architecture, the different types of spatial join queries considered in this paper, and different query processing strategies. Section 3 proposes three techniques to transfer data between servers, and Section 4 shows how to simulate spatial joins as window selections and filter steps. In Section 5 we explain how to eliminate false hits and produce the final set of results. Section 6 contains our experimental results. Section 7 discusses the related work in this area. Finally, Section 8 concludes the paper and provides an overview of our future plans.

2. Overview

The spatial join of two relations R and S, denoted by $R \bowtie_{i\theta j} S$, contains those tuples in the Cartesian product $R \times S$ where the *ith* column of R and the *jth* column of S are of some spatial data type, and θ is a binary spatial operator (e.g., intersection, equality (equijoin), topological (overlap), directional (north)).

One target application for spatial join is Web-based spatial data integration systems. *Distributed geographic information systems* (Wang et al 1999) and *TheaterLoc* (Barish et al 2000) are examples of such applications. Distributed geographic information systems usually store digital maps and aerial images and provide functions for retrieving and manipulating data. A distributed *GIS* consists of autonomous sites connected with a network. TheaterLoc is an information integration client/server application that allows users to retrieve information about theaters and restaurants for different cities. The goal of these applications is to integrate the spatial data from existing Web sites and to provide transparent and efficient access in which a user can access data stored at remote Web GIS systems as though all the data and functions reside on a local site.

In this paper, we assume that the underlying architecture, Figure 1b, consists

Query Processing Phases

1: Local : Local To Remote Transfer { Polygon | Fixed-BR | Dynamic-MBR}
2: Remote: Spatial Selection { Window-Selection | Aggregate-Window-Selection }
3: Remote: [Remote Filter]
4: Remote: Send to Local (Intermediate-Results)
5: Local : Refinement { Spatial-Join | Iterative-Refinement | Statistical-Refinement }

Fig. 2. The phases of alternative query processing strategies

of an Information mediator, which has access to a local database server and other remote information sources. The mediator has full control over the local database and can perform spatial operations. On the other hand, the mediator has only read access to the remote servers and therefore can only perform window selections.

For the purpose of illustration, we use two example queries throughout the paper. The first query *(SQ-1)*, a *Point-Poly* query, is between a point and a polygon data sets: "*Find all zip codes with more than X number of Chinese restaurants*". The table containing the information about the zip codes and their respective coordinates/regions are obtained from the Web sources and stored/cached in the mediator, while information about the restaurants reside on the remote servers. The second query example *(SQ*-2), a *Poly-Poly* query, is between two polygon data sets: "*Find all telephone area codes with population more than Y, given the population of their overlapping zip codes*". In this case either data set can be cached in the local database to perform the spatial join.

In a single server environment, both data sets of a spatial join reside on the same system and the query response time depends only on the cost of the spatial join, which depends on the size of the data sets and the complexity of the spatial predicate. However, in a distributed environment with restricted access to the remote servers, performing spatial join queries must be simulated by spatial select operations after transferring whole/partial data sets from the local to the remote server. In addition, the query response time is a function of size and complexity of the data transferred between the servers. Our approach to perform spatial join in such environments consists of three steps: a) local to remote transfer, in which a representation of the query objects is selected and transferred to the remote server, b) spatial selection, which selects the set of candidate objects to be transferred to the local server, and c) local refinement, in which the candidate objects are refined to produce the final set of results.

Three query processing strategies that differ in the way they filter the false hits and whether the user is interested in exact but slow or approximate and fast response can be utilized. The strategies are as follows:

Strategy-1 First, transfer the query objects to the remote site. Second, perform window selection at the remote site. Third, apply remote filter. Finally, transfer the candidate objects back to the local server for refinement.

Strategy-2 First, transfer the query objects to the remote site. Second, perform window selection at the remote site. Finally, transfer the candidate objects back to the local server for refinement. Note that no filtering is used with this method.

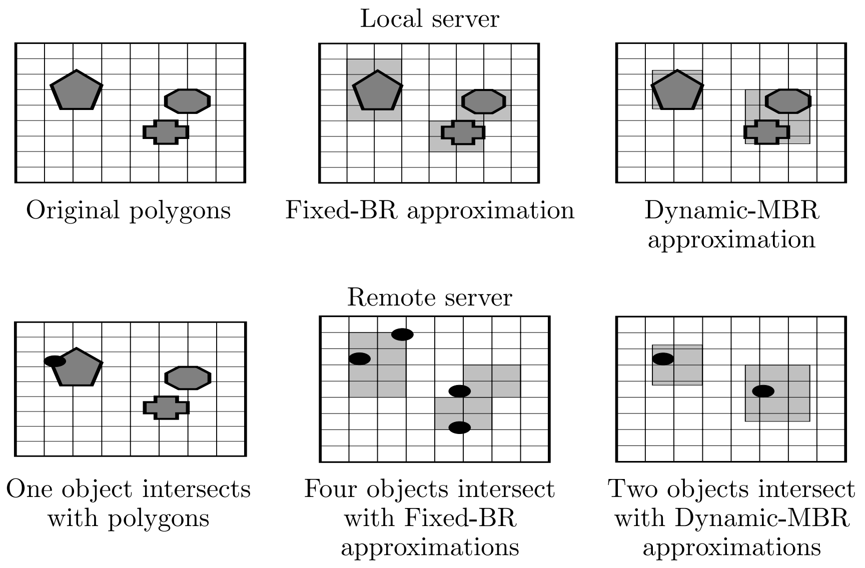

Local server
Original polygons
Fixed-BR approximation
Dynamic-MBR approximation
Remote server
One object intersects with polygons
Four objects intersect with Fixed-BR approximations
Two objects intersect with Dynamic-MBR approximations

Fig. 3. Local to remote transfer methods

Strategy-3 First, transfer the query objects to the remote site. Second, perform aggregate window selection at the remote site. Finally, perform statistical refinement. This plan could be used to speed up query processing and provides approximate answers to the queries.

Figure 2 provides the pseudo-code to execute the three strategies, where "[...]" is an optional phase, and "|" is used as an "OR" operator. We describe each phase of the above strategies in Sections 3, 4 and 5 and focus on the effect of our proposed algorithms on the first phase of the above strategies (i.e., local to remote transfer) in our experiments.

3. Step 1: Local to Remote Transfer

The first step in simulating a spatial join query in a distributed environment is termed *Local to Remote Transfer*. This is a very critical step since as shown in Section 6, it has significant impact on the overall performance of the three query processing strategies mentioned in the previous section. In this section, we review two methods, No-MBR and Fixed-BR, and propose two new methods, Fixed-MBR and Dynamic-MBR, to transfer data sets between the local and remote servers, where *MBR* stands for minimum bounding rectangle. With No-MBR, the entire data sets are sent to the remote server while with Fixed-BR, Fixed-MBR and Dynamic-MBR methods, only approximations of the objects are transmitted to the remote server.

3.1. No-MBR

Both query examples *Point-Poly* and *Poly-Poly* described in Section 2 assume that the local database contains the polygon data set. No-MBR is a straightforward method to perform a spatial join between the local and remote server where it transmits a subset of the local database in its entirety to the remote site. That is, the coordinates of the polygon representations of all objects at the local database (large data) are transferred to the remote server. Subsequently, polygon intersection with points/polygons is performed at the remote server as shown in Figure 3-(No-MBR). Even though this method is simple, the data set (polygons) may be too many and its processing too complex. Hence, both the transmission cost of the data between the two servers and the remote query processing cost may be high. Moreover, No-MBR is not applicable to the remote servers that do not support spatial operations. On the other hand, since No-MBR utilizes the exact representation of the local data set, the returned data set from the remote server will contain no irrelevant data (i.e., false hits). Hence no refinement operation is required and the results are 100% accurate.

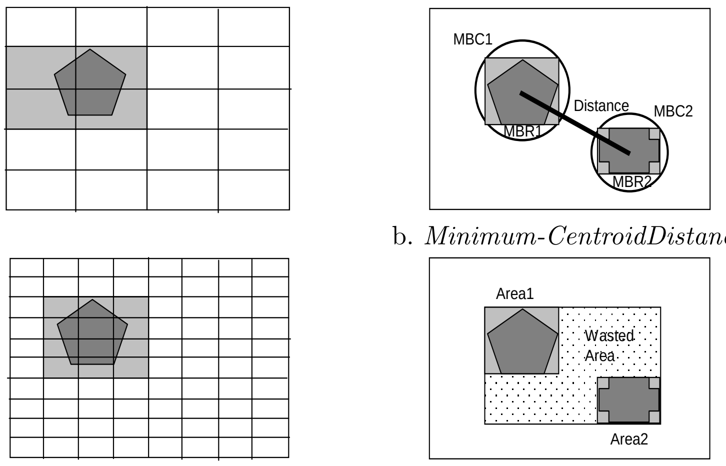
3.2. Fixed-BR

Fixed-BR transmits the bounding rectangles of the polygons as their approximation to the remote server. The bounding rectangles approximate the polygons by the union of the partitions of a pre-specified fixed grid in which they lay (see Figure 3-(Fixed-BR)). This produces some false hits, a subset of the results of the query that satisfy the join operation for the approximate data set but not for the original data set. Therefore, a refinement step is required to filter out the false hits. This step must be performed at the local server after receiving the false hits (more on this in Section 5).

The Fixed-BR method affects the overall query response time in four stages:

1. Reduces the transmission time from the local server to the remote server by transmitting the approximation rectangles (i.e., MBR) instead of the actual polygons.
2. Reduces the query processing time in the remote server by reducing the complexity of the query since simple rectangles are used instead of complex polygons.
3. Increases the time to transmit the results from the remote server to the local server because of the generated false hits in the result set.
4. Depending on the application, may increase the overall response time by incurring an overhead for refining the results to filter out the false hits.

The number of the partitions used to decompose the globe into grids affects the precision of the approximation of the objects, which in turn would impact the number of false hits produced at the remote server. Figure 4a shows a polygon object represented by two grids with different sizes. The smaller the size of the grids, the more accurately the object is approximated. Moreover, when the network is slow, the saving obtained by stage 1 may be surpassed by the overhead incurred by stage 3. This is because the false hits may introduce larger volume of data than the amount of data saved by the Fixed-BR approximation technique. Our experiments (see Section 6) show that Fixed-BR has a better performance over No-MBR only when stage 4 (i.e., refinement step) can be avoided.



b. *Minimum-CentroidDistance*

a. Fixed-BR with different grid sizes c. *Minimum-WastedArea*

Fig. 4. Mesh partitions for Fixed-BR and heuristics for Dynamic-MBR

3.3. Fixed-MBR

When the size of the grids in Fixed-BR becomes very small as compared to the size of the objects, the Fixed-BR approximation method becomes similar to *MBR* (or *Fixed-MBR*), that is when the polygons are approximated with their exact minimum bounding rectangles. The advantages of Fixed-MBR over Fixed-BR are: a) fewer number of false hits are generated by Fixed-MBR since the area covered by Fixed-MBR is always less than or equal to the area covered by Fixed-BR for any polygon, and b) there is no need for a pre-specified fixed-grid partitioning of the space.

3.4. Dynamic-MBR

Dynamic-MBR is similar to Fixed-BR as it also approximates the polygons with their bounding rectangles, but differs from Fixed-BR since it does not restrict the approximations to pre-specified mesh of bounding rectangles. As a result, it incurs an extra overhead to compute the minimum bounding rectangles for each polygon at the query time. However, our experiments show that this overhead (as well as the overhead for merging close polygons) is negligible as compared to the overall query response time. After Dynamic-MBR computed the minimum bounding rectangles of the polygons, it merges the bounding rectangles of the close polygons to reduce the number of window queries (i.e., remote spatial selections). As a result of merging the polygons, a wasted area is generated (see Figure 3-(Dynamic-MBR)). This results in even more false hits in the result set as compared to Fixed-MBR. Hence, local refinement is required to filter out the false hits. There is a trade-off between the number of polygons merged together and the number of false hits generated as a result of merging: the more polygons merged, the more wasted area in the search space and hence the more false hits are generated. We propose two heuristics to decide when to merge the polygons.

The Dynamic-MBR method merges two minimum bounding rectangles, MBR1 and MBR2, into one bounding rectangle that contains both of them, NewMBR, when they meet the conditions required by one or both of the following two heuristics:

Minimum-CentroidDistance: The distance between the centroids of *MBR1* and *MBR2* to the summation of the radii of their minimum bounding circles (*MBC*) is less than *Threshold1* (see Figure 4b):

$$\frac{Distance(Centroid(MBR1),Centroid(MBR2))}{Radius(MBC(MBR1))+Radius(MBC(MBR2))} < Threshold1$$

This heuristic tends to merge polygons when they are close.

Minimum-WastedArea: The ratio of the area in the new bounding rectangles that was not covered by *MBR1* and *MBR2* (i.e., wasted area) to the total area of the new bounding rectangle is less than *threshold2* (see Figure 4c):

$$\frac{Area(NewMBR)-Area(MBR1)-Area(MBR2)}{Area(NewMBR)} < Threshold2$$

This heuristic tends to merge polygons when the new bounding rectangle does not introduce large area as compared to the original polygons.

Similar to Fixed-BR, Dynamic-MBR also affects the overall query response time in four stages. In stage 1, in addition to the reduction obtained from approximating polygons by rectangles, it also reduces the size of the data to be transmitted by reducing the number of polygons as a result of merging. In stage 2, in addition to reducing the complexity by using simple rectangles, it reduces the number of window queries as a result of merging polygons together. Stages 3 and 4 are identical to Fixed-BR. Our experiments (see Section 6) show that the impact of reducing the number of window queries is greater than the impact of reducing the queries' complexity by using rectangles.

The two proposed heuristics have different impacts on the four stages of overall query response time. Minimum-CentroidDistance is more concerned with stages 1 and 2 by trying to increase the probability of merging polygons and hence reducing the number of both transmitted MBRs (i.e., stage 1) and window selection queries (i.e., stage 2). Instead, Minimum-WastedArea is more focused on stages 3 and 4 by striving to reduce the wasted areas and hence decreasing the number of false hits. Therefore, for an environment with fast network and powerful local server, Minimum-CentroidDistance should potentially outperform Minimum-WastedArea. Our experiments in Section 6 confirm this intuition.

4. Step 2: Spatial Selection

This section focuses on the second step in simulating a spatial join query in the Web environment, which is termed *spatial selection*. Spatial selection retrieves from a dataset the entries that satisfy some spatial predicate with respect to a reference object. Spatial selection operation is usually processed in two steps, termed *window selection* and *filter* steps that are discussed in details in Sections 4.1 and 4.2 respectively.

4.1. Window Selection Step

The most common type of spatial selections are window selections, where the reference object defines a window in the workspace. In the window selection step, all the objects at the remote server that intersects with the query objects at the local server are found. This would generate a list of candidate join pairs and eliminates objects that could not possibly satisfy the query. At this step, a spatial predicate is performed on the representation of the query objects by their exact geometry (polygons), or their spatial approximations (*BR*, or *MBR*). Besides the traditional window selection method, we propose another spatial selection method called *aggregate window selection.* With aggregate window selection we only collect statistical information about the objects within the window instead of their exact information. Subsequently, *statistical refinement* could be used to get the final query answer. This type of operation is useful when the user is interested in approximate answers to the query and desires faster query response time.

For example, with a *Point-Poly* query as in *SQ-1* described in Section 2, using the traditional window selection operation we find all the Chinese restaurants inside the query window and then maintain their location coordinates. Subsequently, these location coordinates are examined against the exact representation of the query object to eliminate the false hits. On the other hand, with the *aggregate window selection*, only the total number of the restaurants is computed. Consequently, by using the exact and the approximate representations of the query object we could approximately compute the probability that this zip code area has more than X Chinese restaurants. *Statistical refinement* is explained in details in Section 5.3.

4.2. Filter Step

The second step of spatial selection is the filter step, in which the candidate join pairs are examined under the predicate of the query. Thus, some data items from the list of candidates generated by the window selection step can be discarded. Filter step can be executed at the remote server and candidate data sets can be transmitted to the local server for the final refinement. An alternative approach is to send the data set generated after the window selection to the local server to perform filtering and refinement in a single step at the local server. The tradeoff here is between the cost of extra transfer operations, for the data that was not filtered, and the cost of extra operations required to perform the filter remotely. Note that depending on the functionality of the remote server we may or may not be able to perform the filter step remotely.

Both query types Point-Poly and Poly-Poly discussed in Section 2 have an auditing/filtering aspect. For example, with a Point-Poly query as in *SQ-1*, if a polygon, BR or MBR, is identified within the window selection step that has less than X number of Chinese restaurants, it is guaranteed that it will not appear in the final result. Similarly, with a Poly-Poly query as in *SQ-2*, if a polygon, BR or MBR, is found after the window selection step such that the total population of all the overlapping zip codes for this area code is less than Y, it is guaranteed that the population of the area code cannot be more than Y. It is important to note the difference between the filter and refinement steps. The filter step can be performed using a regular non-spatial predicate and is aimed to discard the polygons' MBRs that do not satisfy the query predicate before sending their contained points to the local server. While the refinement step that requires spatial functionality will discard the false hits that are resulted due to the utilization of objects' approximations such as BR or MBR. For example,

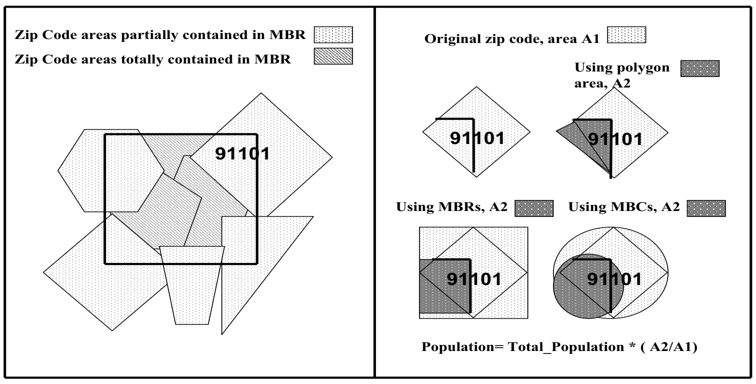

Zip Code areas partially contained in MBR
Zip Code areas totally contained in MBR
91101
Original zip code, area A1
Using polygon area, A2
91101
91101
Using MBRs, A2
Using MBCs, A2
91101
91101
Population= Total_Population * (A2/A1)

Fig. 5. Probability computation

with the Point-Poly query as in *SQ-1*, the filter step discards the polygons that have less than X number of Chinese restaurants in their BR or MBR, while the refinement step eliminates those restaurants that are outside the polygons that passed the filter step, but are inside their BRs or MBRs.

Choosing whether to perform the remote filter step or not depends on:

– The functionality provided at the remote server.
– The size of the data set generated from the window selection step.
– The CPU load and the power of the local and remote servers.
– The workload and the available bandwidth of the network.
– The tradeoff between the cost of remote processing and transmitting the data.

5. Step 3: Refinement

The final step of simulating a distributed spatial join query is the *refinement* step that must be executed at the local site since there is no write access to the remote site. This operation eliminates all the false hits that were produced from the window selection at the remote server. We propose three alternative methods to execute the refinement operation: *local spatial join, pipelined refinement, statistical refinement.*

5.1. Local Spatial Join

With this method, the candidate set of objects is joined with the exact representations of the query objects using traditional spatial join operator $Rl' \bowtie_{i\theta j} Rr'$ where Rl' is the query objects relation and Rr' is the candidate set of objects relation. For example, with query example 1 described in Section 2, Rl' would contain the zip codes and their respective coordinates and Rr' would contain the restaurants' names and their coordinates (locations).

5.2. Pipelined Refinement

With this method, instead of executing spatial join operator, we refine the candidate objects by evaluating the predicate of the query as each (or a block of) candidate object becomes available. For example, for the same previous query, we return Rr' tuples to the local server in a pipelined manner and for each tuple (or a group of tuples) we examine if the coordinates of the restaurant falls inside the query zip code area. Note that we do not have to construct Rr' in its entirety before performing the pipelined refinement. Each candidate object from the window selection could be transferred to the local site immediately to perform the pipelined refinement without first waiting to identify all candidate objects.

5.3. Statistical Refinement

This type of refinement is useful when the user is interested in approximate instead of exact answers to the query. This is the case when we use *aggregate window selection* at the remote site. For example, with query example SQ-1 in Section 2, Rl' would contain zip codes and their respective coordinates and Rr' would only contain the number of the restaurants inside the query window instead of the names or the coordinates (locations) of the restaurants. Consequently, we employ statistical refinement that could estimate the number of restaurants in a zip code area based on the areas of the original zip code and its approximation (that was used for window selection). For example, suppose that the area of zip code *Z1* is *A1*, and its Fixed-BR approximated area is *A2* and the total number of restaurants inside Fixed-BR is *N*. Thus, we could estimate the number of restaurants inside *Z1* as *N*A1/A2*, assuming that the restaurants are uniformly distributed in the space. In other cases (e.g., *Poly-Poly* queries), the processing cost can be reduced as well if the user is not concerned with the accuracy of the results. For example, consider the spatial query example 2 discussed in Section 2 with changing the relation *"less"* to *"roughly less"*. In this case, some of the zip codes will be fully contained inside the query area codes, while others will be partially included. Given a uniform distribution of the population within n zip codes, and each zip code (Z_i) with a population density of Z_i^{pd}, then the population of an area code (PAC_j) is computed as:

$$PAC_j = \sum_{i=1}^{n} Z_i^{area} \times Z_i^{pd}$$

where Z_i^{area} is the area of Z_i that is included in the area code region. For those zip codes that are partially included in an area code, polygon intersection algorithm must be invoked to compute their intersection area. However, instead of computing the exact area of the intersection, the area can be approximated by bounding the intersection region using *MBR* or minimum bounding circle *MBC*. Consequently, the population of an area code (PAC_j) could be approximately computed as:

$$PAC_j = \sum_{i=1}^{n} Z_i^{b-area} \times Z_i^{pd}$$

where Z_i^{b-area} is the area of *MBR* (or *MBC*) of that portion of Z_i that is included in the area code region.

Figure 5 shows a sample query that utilizes objects' approximations to estimate the total population of zipcode areas. It shows how to calculate the population of a sub-area of the zipcode "91101" that is included inside the query window using three different methods. In the first method, it computes the population using exact polygon areas. While in the second and third methods, it computes the population using *MBR* and *MBC* approximations of the areas, respectively.

6. Performance

As we argued in Section 2, the response time of a spatial join query depends on the cost of the spatial join operation, which depends on the size of the data sets and the complexity of the spatial predicate. However, in the web environment the response time also depends on the speed of the network which connects the local and remote servers, the processing and functionality power of the remote servers, and the tolerance of the application towards inaccuracy. The behavior of such system cannot be captured and analyzed accurately by analytical models or simulations. Hence, we decided to implement a real experimental setup. To compare our alternative techniques we use query response time, both including and excluding the local refinement step, as our performance measure, and percentage of false hits as our accuracy measure. We conducted four sets of experiments to evaluate the performance and accuracy of distributed spatial joins on the Web environment using our alternative query processing strategies:

1. The first set of experiments evaluates the impact of number of partitions/grids on Fixed-BR.
2. The second set of experiments studies the impact of different threshold values for the Minimum-CentroidDistance and Minimum-WastedArea heuristics on Dynamic-MBR.
3. The third set of experiments compares No-MBR, Fixed-MBR and Dynamic-MBR. We use Fixed-MBR since it is similar to the optimum configuration for Fixed-BR (i.e., when the size of the grids are very small as compared to the size of the objects).
4. Finally, we investigate the performance of the alternative techniques for remote servers with no spatial support that can only support non-spatial predicates such as *"X between (x1,x2) and Y between (y1,y2)"*.

In all the experiments, we assumed window selection with no filtering for step 2 and local spatial join as the refinement step (i.e., step 3) and report the results for the *Point-Poly* queries.

6.1. Experimental Setup

Figure 1b depicts our experimental setup. It consists of two servers (sites), *Local* and *Remote*, which are connected through a 100 mpbs LAN. The local server is an IBM ZPro with two Pentium III Xeon processors, 768 MB RAM and 18 GB SCSI disk running Windows 2000. The remote server is a SUN Ultra Enterprise 450 with two Sparc II processors, 1 GB RAM and 50 GB RAID system running Solaris 2.6. Both systems run Informix Universal Server 9.2 with Geodetic Spatial Datablade 2.12.TC1, which uses R-Tree access method for spatial indexing. The Informix database server uses pipelined method to send the results of a query to the client as they become available.

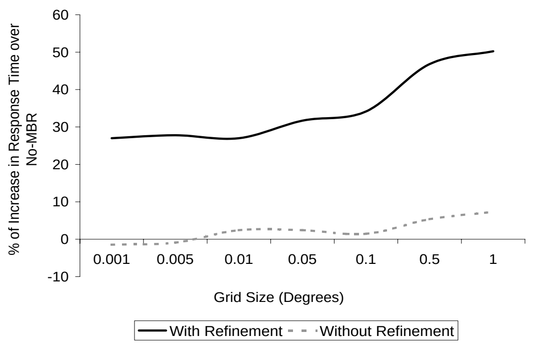
We performed our experiments with a set of real data. The data contains information about USA and includes 170,000 points (e.g., schools, hospitals, etc.) stored in the remote server. It also includes 41000 polygons with an average of 16 vertices per polygon, stored in the local server, representing zip codes in the USA. A Java program, as our client, uses JDBC to submit the queries to and retrieves the results from both servers. The client has full access (read/write) rights to the local site but only has read access to the remote site in order to mimic a web source. We apply local refinement step to eliminate the false hits from the final results. Similar to Informix, JDBC also uses pipelined method to retrieve the results of a query from the database server as they become available. Since both the database server and JDBC use pipelined method to send/retrieve results of a query, it is not possible to separate the network transmission and remote processing times. But because of the very fast network we used in our experiments and considering the size of the queries and results, the network transmission time is negligible as compared to the remote processing time.

6.2. Results

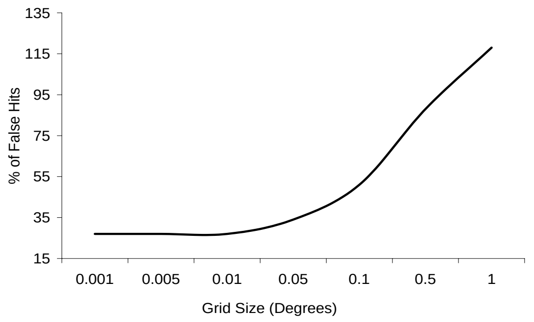
We used the following query for all the experiments: “Find all points that are inside a set of n randomly selected polygons”. We run the experiments with n=5, 20, 100, 250 and 500 polygons. We selected the polygons from randomly selected regions with areas equal to four times than the total average area of n polygons. Consequently, some of the selected polygons are expected to be close enough to satisfy either of the heuristics of Dynamic-MBR and merge together. This is done to observe the impact of Dynamic-MBR in merging the polygons. Note that when the polygons are far enough from each other that the heuristics cannot reduce the number of polygons by merging them, the results for Dynamic-MBR will be similar to Fixed-MBR.

6.2.1. Impact of Grid Size on Fixed-BR

The average size of the polygons in the local database is 0.25×0.1 degrees. We varied the grid size (X-axis) of Figure 6 for Fixed-BR from 0.001° to 1°. The figure illustrates the impact of grid size in query response time for the scenario where the remote server searches for points inside 100 polygons. The Y-axis of Figure 6a is the percentage of increase in query response time over No-MBR as the baseline. As shown in the figure, Fixed-BR without the refinement step marginally outperforms No-MBR when the grid size is small relative to the polygons’ average size. The overall response time (i.e., including the refinement step) is between 20% to 30% slower, which is because of the overhead of the local refinement. By increasing the grid size, the response time of Fixed-BR without refinement becomes worse, up to 10% slower than No-MBR, and the response time with refinement becomes up to 50% slower than No-MBR. This is because by using larger grid size, more false hits are generated that must be transmitted and refined. As a result, the savings described in items “1” and “2” of Section 3.2 become less significant as compared to overheads introduced by items “3” and “4”. In other words, reducing the complexity of the queries by using Fixed-BR has less impact on the response time (both with and without refinement) as compared to the impact of the increase in number of false hits. This becomes worse when: 1) the polygons are far from each other, since in this case even more



a. Increase in query response time over No-MBR



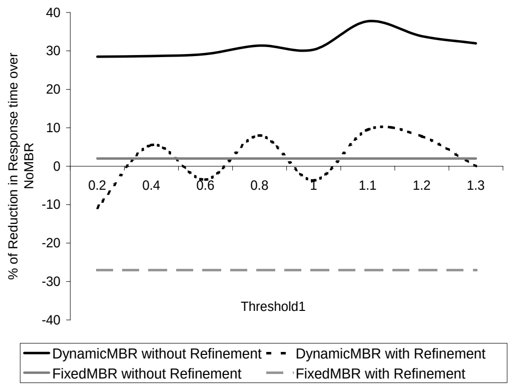
b. Percentage of false hits

Fig. 6. Impact of Grid Size on Fixed-BR (n=100)

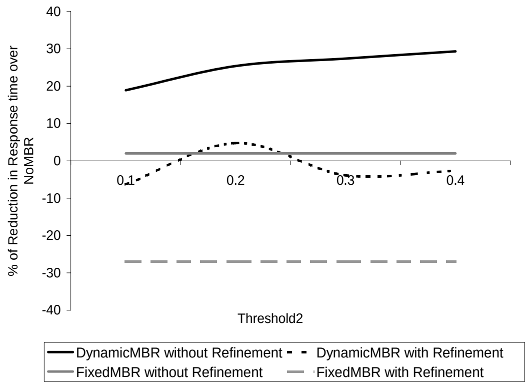
false hits are generated and 2) when the network is slow, because saving of time in item “1” becomes even less important as compared to the overhead of item “3”. As we increase the grid size from 0.001° to 1°, the percentage of false hits also increases from 30% to 115%. Note that when the polygons are close to each other, Fixed-BR of neighboring polygons may overlap and hence the points in the overlap areas are selected and transmitted only once. Instead, for far polygons the possibility of overlaps is less, resulting in coverage of larger area and generation of even more false hits. Figure 9a shows the percentage of the false hits for the Fixed-MBR method, that performs similar to the optimum configuration of Fixed-BR, for 5 to 500 polygons. Our experiments show between 22% to 30% false hits for Fixed-MBR. We conclude that Fixed-MBR, and hence Fixed-BR, behave inferior to No-MBR especially when 100% accuracy is required and/or in an environment with a slow network.

6.2.2. Impact of Heuristics on Dynamic-MBR

Impact of Minimum-CentroidDistance Heuristic: In order to study the impact of the Minimum-CentroidDistance heuristic on Dynamic-MBR, we varied



a. *Minimum-CentroidDistance* heuristic



b. *Minimum-WastedArea* heuristic

Fig. 7. Impact of merging heuristics on efficiency of Dynamic-MBR (n=100)

the value of threshold1 from 0.2 to 1.3. These values for threshold1 are selected in order to merge polygons that are fairly close and avoid merging polygons that are far from each other to avoid generating large number of false hits. Figure 7a shows the performance of the Dynamic-MBR method for different values of threshold1 where the remote server searches for points inside 100 polygons. As shown in the figure, Dynamic-MBR with threshold1 can substantially reduce the response time without the refinement step, between 27% to 37%, as compared to No-MBR. For certain values of threshold1, it even outperforms No-MBR considering the total response time with refinement. The reason is that Dynamic-MBR with threshold1 reduces the number of polygons that are to be transmitted to the remote server, between 30% to 36.2%, which significantly reduces the number of the window queries in the remote server. As shown in the figure, Dynamic-MBR has a better performance as compared to Fixed-MBR. Figure 9a shows the percentage of the false hits for Dynamic-MBR with threshold1=1.1 for 5 to 500 polygons. As shown in the figure, Dynamic-MBR with threshold1 generates between 38% to 60% false hits, which is more than that of Fixed-MBR. This is because by merging polygons the approximation area of Dynamic-MBR increases as compared to Fixed-MBR. Our experiments suggest that 1.1 is the optimum value for threshold1 for our data set, since it provides the best decrease in response time while shows only 8% more false hits than the least percentage of false hits incurred by other values of threshold1.

We conclude that even though Dynamic-MBR generates a large amount of false hits, it is still highly superior to No-MBR when the false hits are tolerable (i.e., Dynamic-MBR without refinement step). In addition, depending on the environment and the nature of the data, with finely tuned parameters, Dynamic-MBR can even outperform No-MBR when 100% accuracy is necessary (i.e., Dynamic-MBR with refinement step). This shows that not only the saving in processing time of the queries, as a result of reducing the number of remote window queries, is much more than the overhead of transferring false hits, but also it may surpass the total overhead including the refinement step. The advantage of Dynamic-MBR with the Minimum-CentroidDistance heuristic may become less magnified when the network is very slow. This is because with slow networks, the overhead of transferring large amount of false hits may become more significant as compared to the saving in query processing time. It is important to note that if the remote server has no spatial support, then Dynamic-MBR will be the superior choice (more on this in Section 6.2.4).

Impact of Minimum-WastedArea Heuristic: We varied the value of Threshold2 from 0.1 to 0.4 to study the impact of the Minimum-WastedArea heuristic on the performance and accuracy of Dynamic-MBR. Similar to threshold1, these values for threshold2 are selected to merge close polygons in order to avoid generating large number of false hits. Figure 7b shows the percentage of reduction in response time, with and without refinement, for Dynamic-MBR with threshold2 as compared to No-MBR for 100 polygons. As depicted in the figure, when we increase the value of threshold2, the decrease in response time without the refinement step steadily increases, from 18.9% to 29.3%. This is because the higher the value, the more polygons are merged together and hence the less the number of window queries. Our experiments show that threshold2=0.2 provides the best performance for total response time. The reason is that the overhead associated with processing time of local refinement of false hits with threshold2 greater than 0.2 surpasses the gain obtained from reducing the number of selection queries. As shown in the figure, Dynamic-MBR with threshold2 always outperforms the Fixed-MBR method. Figure 9a depicts the accuracy of Dynamic-MBR with threshold2=0.2 for 5 to 500 polygons. The figure shows that the false hits generated by Dynamic-MBR with threshold2, 29% to 36%, is always less than that of threshold1 but more than that of Fixed-MBR. The trend observed for other values of threshold2 is similar to this figure.

We conclude that Dynamic-MBR with Minimum-WastedArea, similar to Minimum-CentroidDistance, is always superior to No-MBR when the application can tolerate false hits. By selecting appropriate values for threshold2, Dynamic-MBR with threshold2 outperforms No-MBR even in total response time. The advantages gained by Dynamic-MBR with Minimum-WastedArea become more magnified when network is slow due to the reduction in the number of the false hits.

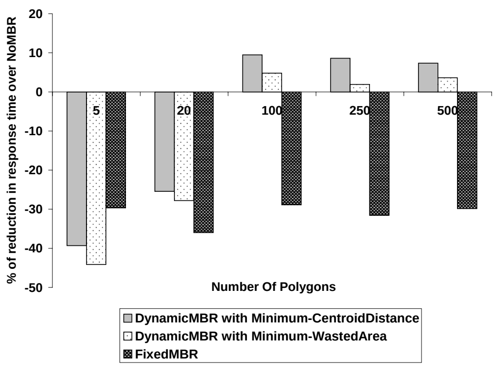
6.2.3. Comparison Among Alternative Approximation Techniques

Figure 8 compares the percentage of decrease in response time, both with and without refinement step, for Fixed-MBR and Dynamic-MBRs with threshold1=1.1 and threshold2=0.2, over No-MBR. We selected Fixed-MBR since its performance is similar to the optimum configuration of Fixed-BR. We varied the number of polygons from 5 to 500 in the X-axis. As illustrated in Figure 8a, when 100% accuracy is required (i.e., response time with refinement), Fixed-MBR is always inferior to No-MBR but Dynamic-MBRs show superiority over No-MBR for larger values of n. In addition, the impact of threshold1 for Dynamic-MBR always outperforms threshold2. However, when accuracy is tolerable (i.e., response time without refinement), as depicted in Figure 8b, Dynamic-MBRs always show superiority over No-MBR, with threshold1 always outperforming threshold2. In this case Fixed-MBR performs very similar to No-MBR.

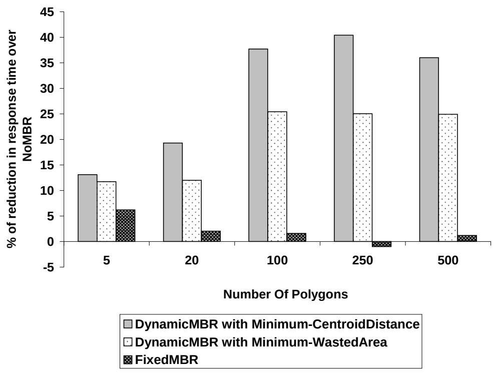
Figure 9 depicts the tradeoffs between percentage of false hits and number of transmitted polygons when we transmit n=5 to 500 polygons to the remote server. As shown in Figure 9a, Dynamic-MBR with Minimum-CentroidDistance always generates more false hits and Fixed-MBR generates the least number of false hits.

By comparing the results of our experiments we conclude that:

- The saving in query processing time by reducing the complexity of the queries in the remote server when we simply approximate complex polygons with rectangles (i.e., Fixed-MBR), will eventually become ineffective because of two factors: a) overhead of transmitting the false hits, and b) The overhead associated with filtering out the false hits, which is more important than the transmission cost specially when the network is fast. As shown in Figure 8a, even though Fixed-MBR reduces the complexity of the queries, it is always between 29% to 36% slower than No-MBR.
- The impact of reducing the number of the window queries by merging polygons (i.e., Dynamic-MBRs) on query processing time is higher than only reducing the complexity of the queries (i.e., Fixed-BR). As shown in Figure 9, even though Dynamic-MBR with the Minimum-CentroidDistance heuristic always generates more false hits than the other methods (between 10% to 26% more than threshold2 and 16% to 29% more than Fixed-MBR), it always outperforms Dynamic-MBR with Minimum-WastedArea and Fixed-MBR in terms of efficiency, both with and without refinement (Figure 8). The reason is that Fixed-MBR is not aimed to merge the polygons, and Dynamic-MBR with Minimum-CentroidDistance always merges more polygons than Minimum-WastedArea (between 5% to 11% more).
- When the existence of the false hits in the result set is not critical for an application, the Dynamic-MBR methods are the preferred techniques. With fine tuning the parameters of these methods depending on the network's speed and distribution of the polygons and points, the Dynamic-MBR methods can replace No-MBR even for applications with 100% accuracy requirements.
- Depending on the environment, either the Minimum-CentroidDistance or Minimum-WastedArea heuristic can be selected for Dynamic-MBR. Dynamic-MBR with Minimum-WastedArea may show better performance when the network is very slow and the results are not pipelined to the client, which causes the overhead of transmitting more false hits surpasses its saving in processing time. In other situations, Dynamic-MBR with Minimum-CentroidDistance is the preferred



a. With refinement



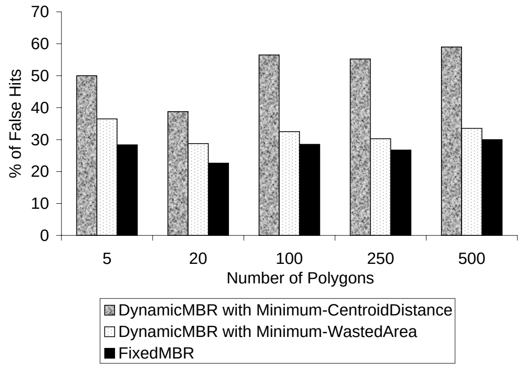
b. Without refinement

Fig. 8. Reduction of response time over No-MBR

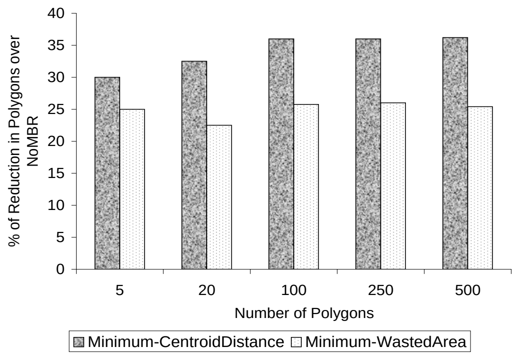
technique since it provides better response time for both with and without refinement cases.

6.2.4. Remote Servers with No Spatial Support

We run the experiments described in the above sections for scenarios where the remote server can only support non-spatial predicates such as *"X between (x1,x2) and Y between (y1,y2)"*. Note that in this case No-MBR method is not applicable. Figure 10a shows the response time for Fixed-MBR as well as Dynamic-MBR with threshold1=1.1 and threshold2=0.2. As shown in the figure, the response time of different methods increases linearly. In addition, even though the response times of all the methods are substantially more as compared to the cases that the remote server supports spatial operations, but Dynamic-MBR still shows improvement over Fixed-MBR. The improvement of Dynamic-MBR with threshold1 as compared to Fixed-MBR in this case is only 6% to 27%, as opposed to 7% to 41% for the case with spatial support. Even though the percentage of false



a. False hits



b. Reduction in transmitted polygons (boxes) over No-MBR (or Fixed-MBR)

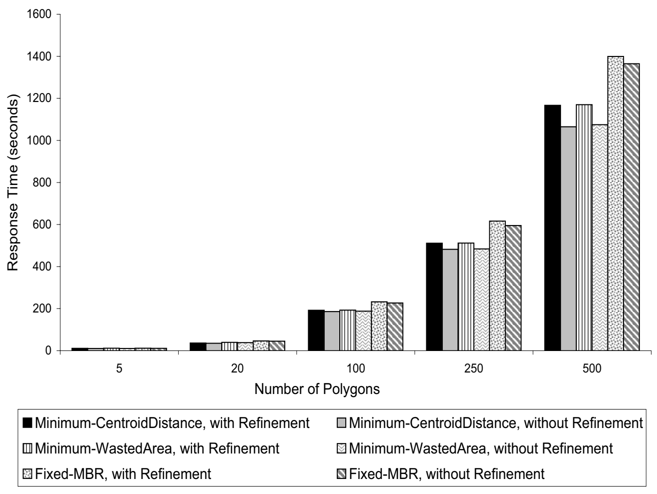
Fig. 9. Tradeoffs between percentage of false hits vs. number of transmitted polygons/boxes

hits for different methods is identical to the case with spatial support, Dynamic-MBR with threshold1 and threshold2 show similar performances which makes threshold2 a better candidate for approximation.

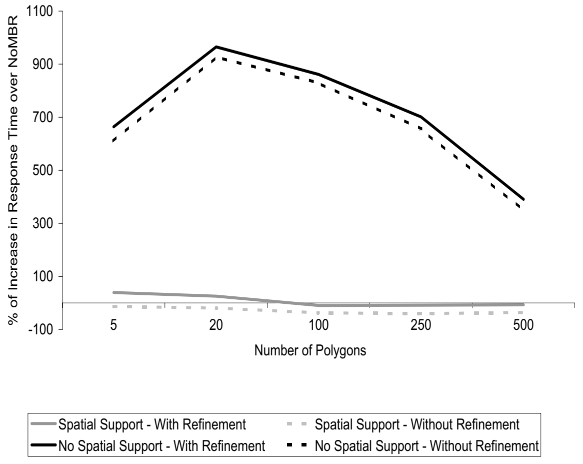
Figure 10b shows the percentage of increase in response time for Dynamic-MBR when the remote server does not support spatial functionality over No-MBR and Dynamic-MBR for the same set of data when the remote server supports spatial functionality. As depicted in the figure, lack of spatial support in the remote server dramatically increases the response time.

7. Related Work

Algorithms for the spatial join operation can be categorized into two groups. The first group includes algorithms that utilize pre-constructed indices to improve the performance of the spatial join operation. While, the second group includes algorithms that assume no indices are available on the data sets. In general,



a. Response time



b. Increase in response time for Dynamic-MBR

Fig. 10. Comparison of different methods for remote servers with no spatial support

our assumed architecture restricts access to the remote site, which renders the techniques proposed in either category impractical. In addition, given our application, set of spatial objects needed for the join operation become available (or maybe fetched) only at the time of query processing. Hence, we cannot assume that we have access to intermediate nodes and levels of an index structure on this data set. Other studies assume access to intermediate nodes and levels of an index structure (e.g., *R-Tree*) to perform spatial join. The first group (Koudas et al 1996, Bhowmicket al 1998, Abel et al 1995, Huang b et al 1997, Bronkhoff et al 1996, Aref et al 1996, Zhou et al 1997, Papadopoulos et al 1999), use already existing access method (e.g., *R-tree* and *R*-tree*) or special data structures to process spatial join. The justification is that a spatial access method should attempt to store objects, which are close together on a common page to preserve spatial locality. In addition, spatial access methods only manipulate the approximation of objects as opposed to the exact representation.

A technique to decluster a spatial method on a shared-nothing multi-computer architecture that uses *R-tree* as the underlying spatial access method was proposed in (Koudas et al 1996). The major goal of this technique is to find the optimal capacity of the leaf nodes of the *R-tree*. Koudas in (Koudas et al 1996) focused on how this would optimize the response time on range queries while ignoring join queries. In order to improve the efficiency of query operations such as spatial join, Huang in (Huang b et al 1997) present a new *R-tree* based join method *"BFRJ"* that synchronously traverses both relations' *R-trees* in breadth first order while processing join computation one level at a time. An intermediate join index, which deploys global optimization strategies, is created at each level and is stored in main memory or disk. However, Huang et. al assume that the datasets are not distributed and reside on one server. In addition, Huang et. al assume existence of R-tree indices for both relations. This is not true in our environment since the local server (e.g., a Web mediator) does not have R-tree index for the spatial data gathered from other Web servers, and even though the remote server may have an R-tree index, the mediator does not have access to its internal nodes. Brinkhoff in (Bronkhoff et al 1996) show how to process spatial joins on a parallel hardware platform with shared virtual memory that uses *R-tree* as the underlying spatial access method. The parallel approach of join processing is based on the idea of partitioned parallelism, which requires data replication and processor communication. Consequently, an appropriate distribution of objects to processors becomes critical. In general, the studies in (Koudas et al 1996, Bronkhoff et al 1996, Huang b et al 1997), require that the users have write access rights to all servers, which is not feasible within our architecture. A new operator called *Web join* was introduced in (Bhowmicket al 1998). It combines information from two Web tables, based on some criteria and stores the information in a separate Web table to be used for future queries. Its queries have graph like structure and it is used to match the portions of the WWW satisfying a predicate. Web join assumes that the data sets on all servers are described by a data model called Web Data Model. This assumption is too general and may not be provided by all databases. Finally, Abel in (Abel et al 1995) propose a spatial join algorithm based on the concept of spatial semijoin that eliminates objects prior to transmission in order to reduce both transfer and local processing costs. Abel et. al use a B^+*-tree* to provide direct access to the spatial objects based on *locational keys* generated from the objects' *MBR*s. The spatial objects are sorted in ascending sequence by the locational key values, and spatial join between two relations is performed by a merge-like operation along two sorted lists of locational keys. Abel et. al's work also has several differences with our techniques. First, it requires full control (i.e. read/write access) over the remote server to perform normal join between locational keys using B^+*-tree*. Second, it assumes the existence of spatial indices for the data sets (based on sorted single dimensional objects). In addition, it exploits the indices to perform the join operation by using the approximations of the spatial objects provided access to the intermediate nodes of an index structure (e.g., B^+*-tree*).

The second group of studies (Patel et al 1996, Lo et al 1996, Koudas et al 1998) assume that no pre-computed index structures are available on the data sets and hence, alternative spatial join algorithms (*PBSM, SHJ,*S^3J) were proposed to divide the data space into partitions (regularly in *PBSM*, irregularly in *SHJ*, hierarchically in S^3J) and proceed with joining partition pairs. These studies are orthogonal to our work. Even though we are focusing on data transfer between local and remote site, however, once the intermediate results are transferred to the local server, we can use the techniques proposed by any of these studies to perform the local join. In (Patel et al 1996), a generalization of the sort merge join algorithm called partition based spatial merge join *PBSM* was proposed. The algorithm computes the number of partitions into which the data space is divided. Consequently, the corresponding partitions in different data sets are joined together using hash join. Spatial Hash Join (*SHJ*) algorithm in (Lo et al 1996) starts by computing the number of partitions into which the data space should be divided. Subsequently, it uses nearest center heuristic, which depends on the centers of the objects and the partitions, to assign spatial objects to the partitions. Finally, SHJ joins the pairs of corresponding partitions. In contrast, we propose to apply the heuristics dynamically to only those subsets of the data that satisfy the query predicate. Size Separation Spatial Join S^3J (Koudas et al 1998), imposes a hierarchical decomposition of the data space that partitions spatial data sets by size. Since spatial objects may belong to more than one partition, both *PBSM* and *SHJ* introduce replication of the objects in partitions in order to compute the join. In contrast to *PBDM* and *SHJ*, S^3J and our methods require no replication of objects in the input data sets. The previous methods do not study the applicability of their algorithms in a distributed environment, and their data objects may belong to more than one partition. However, in our method data sets reside on more than one server with restricted read/write accesses, and an object belongs to only one partition *BR* or *MBR*.

8. Conclusion

In web-based environments, access to remote servers is restricted, progressive or inaccurate results can be tolerated, and existence of full spatial capabilities (e.g., spatial index structures and operations) cannot be assumed. Therefore, support for spatial joins in these environments becomes challenging. We proposed a three step simulation of spatial join on web-based environments. We put together an experimental setup with real database servers to evaluate our different plans. We demonstrated that Dynamic-MBR, which dynamically approximates and merges polygons at the local site is the superior approach for the first step. We also proposed two alternative heuristics for Dynamic-MBR and we showed that the Minimum-CentroidDistance heuristic results in more merges while the Minimum-WastedArea heuristic results in less number of false hits. Hence, in an environment with fast network and powerful local server that can deal efficiently with false hits, Minimum-CentroidDistance is the superior heuristic since it minimizes the remote query processing time.

We intend to extend this work in four ways. First, we plan to study the impact of remote filtering and aggregate window selection, as well as the statistical refinement process, in order to compare the three proposed query processing strategies. Second, we plan to study other types of spatial queries (e.g., poly-poly) and spatial operations (e.g., inside operation). Third, we are also looking in to utilizing the semantics of the data in our proposed techniques. Finally, we would like to utilize the lessons learned from the above mentioned experiments to design a query optimizer for spatial joins on web sources.

Acknowledgements. This research has been funded in part by NSF grants EEC-

9529152 (IMSC ERC) and IIS-0082826, NIH-NLM grant nr. R01-LM07061, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash gifts from Okawa Foundation, Microsoft, and NCR.

References

Abel D. J., Ooi B. C., Tan K.-L., Power R., Yu J. X. (1995) Spatial Join Strategies in Distributed Spatial DBMS. Proceedings of the 4th International Symposium on Large Spatial Databases (SSD), Portland, Maine, USA, August 6-9 1995, pp. 348-367

Adali W., Candan K. S., Papakonstantinou Y., Subrahmanian V. S. (1997) Query caching and optimization in distributed mediator systems. Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, AZ, 1997

Aref W. G., Samet H. (1996) Cascaded Spatial Join Algorithms with Spatially Sorted Output. Proceedings of the fourth ACM workshop on Advances in Geographic Information Systems,Rockville, Maryland, USA, November 15-16, 1996 pp. 17-24

Arge L., Procopiuc T., Ramaswamy S., Suel T., Vahrenhold J., Scott Vitter J. (2000) A Unified Approach for Indexed and Non-Indexed Spatial Joins. The VII Conference on Extending Database Technology (EDBT), Konstanz, Germany, March 27-31, 2000

Barish G., Chen Y.-S., DiPasquo D., Knoblock C. A., Minton S., Muslea I., Shahabi C. (2000) TheaterLoc: Using Information Integration Technology to Rapidly Build Virtual Applications. 16th International Conference on Data Engineering (ICDE), San Diego, CA, USA, February 29-March 3, 2000

Bayardo R., Bohrer W., Brice R., Cichocki A., Flowler G., Helal A., Kashyap V., Ksiezyk T., Martin G., Nodine M., Rashid M., Rusinkiewicz M., Shea R. , Unnikrishnan C., Unruh A., Woelk D. (1996) Semantic integration of information in open and dynamic environments. Technical Report MCC-INSL-088-96, MCC, Austin, Texas, 1996

Bhowmick S. S., Ng W. K., Lim E. P., Madria S. K. (1998) Join Processing in Web Databases. Proceedings of the 9th International Conference on Database and Expert Systems Applications (DEXA), Vienna, Austria, August 24-28, 1998, pp. 647-657

Brinkhoff T., Kriegel H. P., Seeger B. (1993) Efficient Processing of Spatial Joins Using R-Trees. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993, pp. 237-246

Brinkhoff T., Kriegel H. P., Seeger B. (1996) Parallel Processing of Spatial Joins Using R-trees. Proceedings of the Twelfth International Conference on Data Engineering (ICDE), New Orleans, Louisiana, February 26 - March 1, 1996, pp. 258-265

Etzioni O., Weld D. S. (1994) A softbot-based interface to the Internet. Communications of the ACM, 37(7), 1994

Genesereth M., Keller A., Duschka O. (1997) Informaster: An information integration system. Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, AZ, 1997

Hammer J., Garcia-Molina H., Ireland K., Papakonstantinou Y., Ullman J., Widom J. (1995). Information translation, mediation, and mosaic-based browsing in the tsimmis system. Proceedings of the ACM SIGMOD International Conference on Management of Data, San Jose, CA, 1995

Huang Y.-W., Jing N., Rundensteiner E. A. (1997) Spatial Joins Using R-trees: Breadth-First Traversal with Global Optimizations. Proceedings of 23rd International Conference on Very Large Data Bases (VLDB), Athens, Greece, August 25-29, 1997, pp. 396-405

Huang Y.-W., Jing N., Rundensteiner E. A. (1997). A Cost Model for Estimating the Performance of Spatial Joins Using R-trees. Proceedings of the 9th International Conference on Scientific and Statistical Database Management (SSDBM), Olympia, Washington, USA, 1997, pp. 30-38

Kim S. W., Cho W. S., Lee M. J., Whang K. Y. (1995) A New Algorithm for Processing Joins Using the Multilevel Grid File. Proceedings of the 4th International Conference on Database Systems for Advanced Applications (DASFAA), Singapore, April 11-13, 1995, pp. 115-123

Kirk T., Levy A. Y., Sagiv Y., Srivastava D. (1995) The information manifold. In Working Notes of the AAAI Spring Symposium on Information Gathering in Heterogeneous, Distributed Environment, Technical Report SS-95-08, AAAI Press, Menlo Park, CA, 1995

Knoblock C. A., Minton S., Ambite J.-L., Ashish N., Modi P. J., Muslea I., Philpot A. , Tejada

S. (1998) Modeling web sources for information integration. Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI), Madison, WI, 1998

Koudas N., Faloutsos C., Kamil I. (1996). Declustering Spatial Databases on a Multi-Computer Architecture. Proceedings of the 5th International Conference on Extending Database Technology (EDBT), Avignon, France, March 25-29, 1996, pp. 592-614

Koudas N., Sevcik K. C. (1997) Size Separation Spatial Join. Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 13-15, 1997, pp. 324-335

Koudas N., Sevcik K. C. (1998) High Dimensional Similarity Joins: Algorithms and Performance Evaluation. Proceedings of the Fourteenth International Conference on Data Engineering (ICDE), Orlando, Florida, USA, February 23-27, 1998, pp. 466-475

Lo M. L., Ravishankar C. V. (1996) Spatial Hash-Joins. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996, pp. 247-258

Martynov M. G. (1995) Spatial Joins and R-trees. Proceedings of the Second International Workshop on Advances in Databases and Information Systems (ADBIS), Moscow, 27-30 June, 1995, pp. 295-304

Papadias D., Theodoridis Y., Sellis T. K., Egenhofer M. (1995). Topological Relations in the world of minimum bounding rectangles: A study with R-trees. SIGMOD Conference, 1995, pp. 92-103

Papadias D., Mamoulis N., Theodoridis Y. (1999). Processing and Optimization of Multiway Spatial Joins Using R-trees. Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Philadelphia, Pennsylvania, May 31 - June 2, 1999, pp. 44-55

Papadopoulos A., Rigaux P., Scholl M. (1999) A Performance Evaluation of Spatial Join Processing Strategies. Symposium on Large Spatial Databases (SSD), Hong Kong, China, 1999

Patel J. M., DeWitt D. J. (1996) Partition Based Spatial-Merge Join. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996, pp. 259-270

Rotem D. (1991) Spatial Join Indices. Proceedings of the Seventh International Conference on Data Engineering (ICDE), Kobe, Japan, April 8-12, 1991, pp. 500-509

Thomasic A., Raschid L., Valduriez P. (1997) A data model and query processing techniques for scaling access to distributed heterogeneous databases in disco. Invited paper in the IEEE transaction on Computers, special issue on Distributed Computing Systems, 1997

Wang F. J., Jusoh S. (1999) Integrating Multiple Web-based Geographic Information Systems. IEEE Multimedia, Vol. 6, No. 1, January-March 1999

Zhou X., Abel D. J., Truffet D. (1997) Data Partitioning for Parallel Spatial Join Processing.Proceedings of the 5th International Symposium on Large Spatial Databases (SSD), Berlin, Germany, July 15-18, 1997, pp. 178-196

Author Biographies

Cyrus Shahabi is currently an Assistant Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department and also a Research Area Director at the Integrated Media Systems Center (IMSC) at the University of Southern California. He received his Ph.D. degree in Computer Science from the University of Southern California in August 1996. He has two books and more than sixty articles, book chapters, and conference papers in the areas of databases and multimedia. Dr. Shahabi's current research interests include Streaming Architectures and Multidimensional Databases. He has served as a referee for prestigious database conferences (e.g., SIGMOD, VLDB) and journals (e.g., IEEE TKDE). He was the program committee chair of ACM WIDM'99 workshop and the local chair of ACM SIGMETRICS'2002. He is currently serving on many conference program committees such as IEEE ICME 2002 and ACM CIKM 2002. He is also on the editorial board of SIGMOD DiSC 2002.

Mohammad Reza Kolahdouzan received the B.S. degree in Electrical Engineering from the Sharif University of Technology in Tehran, in 1991, and the M.S. degree in Electrical Engineering (Computer Networks) from the University of Southern California, Los Angeles, CA, in 1998. He is currently working towards the Ph.D. degree in Computer Science at the University of Southern California. He is currently a research assistant working on Multidimensional and Spatial Databases at the Integrated Media Systems Center (IMSC) - Information Laboratory at the Computer Science Department of the University of Southern California.

Maytham Safar is currently an Assistant Professor at the Computer Engineering Department, Kuwait University. He received his B.S. degree in Computer Engineering from Kuwait University in 1992, his M.S. degree in Electrical Engineering from University of Colorado at Boulder in 1995, and Ph.D. degree in Computer Science from the University of Southern California in 2000. He has articles, book chapters, and conference papers in the areas of databases and multimedia. Dr. Safar's current research interests include image retrieval, shape representation, spatial index structures, spatial databases, multimedia databases, and data mining.