An Interactive Framework for Spatial Joins: A Statistical Approach to Data Analysis in GIS

Shayma Alkobaisi
Faculty of Information Technology
United Arab Emirates University
shayma.alkobaisi@uaeu.ac.ae

Wan D. Bae
Mathematics, Statistics and Computer Science
University of Wisconsin-Stout
baew@uwstout.edu

Petr Vojtěchovský
Department of Mathematics
University of Denver
petr@math.du.edu

Sada Narayanappa
Advanced Computing Technology
Jeppesen, Inc
sada.narayanappa@jeppesen.com

June 16, 2011

Abstract

Many Geographic Information Systems (GIS) handle a large volume of geospatial data. Spatial joins over two or more geospatial datasets are very common operations in GIS for data analysis and decision support. However, evaluating spatial joins can be very time intensive due to the size of datasets. In this paper, we propose an interactive framework that provides faster approximate answers of spatial joins. The proposed framework utilizes two statistical methods: probabilistic join and sampling based join. The probabilistic join method provides speedup of two orders of magnitude with no correctness guarantee, while the sampling based method provides an order of magnitude improvement over the full indexing tree joins of datasets and also provides running confidence intervals. The framework allows users to trade-off speed versus

bounded accuracy, hence it provides truly interactive data exploration. The two methods are evaluated empirically with real and synthetic datasets.

Index Terms

interactive queries, spatial join, join probability, probabilistic joins, incremental sampling, quad-tree, R-tree, GIS

1 Introduction

Geographic Information Systems (GIS) have been widely used in many applications for storage, manipulation and retrieval of large geospatial datasets. Each dataset is usually called a layer in GIS. Examples of layers may be roads, rivers, land elevation, etc. Layers are related if they have the same geographic coordinates. Spatial joins between two or more datasets are one of the most common GIS queries for data analysis. An example might be finding all roads within 100 feet of rivers. The result of such a query can be useful in determining roads that might be affected by floods.

GIS are often used to visualize results for the end user to assist in decision making processes. In many applications, obtaining an approximate join result in a reasonably short time is far more important than evaluating an exact join over a long time period. Fast response time is especially important for user-driven data exploration in GIS. Therefore GIS users should be given the chance to identify the interesting pairs of datasets for joins without having to wait to compute the actual full joins.

Interactive spatial joins provide the user with a "big picture" (visualized intermediate results) that allows the user to estimate the final result as well as stop or refine the query if the result does not seem to be interesting. Unfortunately, the large dataset size makes interactive spatial joins difficult. Also current query processing techniques deal with spatial queries in a blocking manner; users have no control over the query processing and they have to wait until the final result

is returned. Our goal is to achieve fast query response times by estimating results of spatial data joins, and to provide visualization tools that allow the user to see the intermediate results, adjust the query parameters as appropriate and drill down to more interesting data (regions).

In this paper we develop a novel spatial join method based on the join probability for raster datasets. We then expand this method to support spatial joins over vector datasets. The proposed probabilistic join allows users to get approximate answers in near instantaneous time. Our proposed framework works as follows: users specify queries and get near instantaneous visualizations of the answer using our proposed probabilistic join method. These result visualizations are approximations with no guaranteed bound of correctness. For queries with interesting results, users can either use our proposed sampling method to get a confidence bounded answer estimate, or compute the full join.

Our probabilistic join method is based on a simple observation of joining two raster datasets; Figure 1 shows examples of two raster datasets; R and S represented by 4×4 raster grid cells having the same geographic boundaries. In this example, R has 8 non-zero data cells (density: 8/16) while S has 9 non-zero data cells (density: 9/16). Then R and S must intersect regardless of the shapes and locations of their data cells. The main idea is to calculate the join probability of the given datasets that have the same geographic coordinates using the density of each dataset.

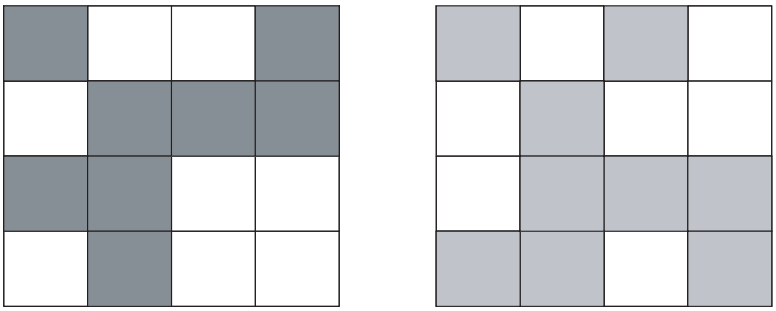


Figure 1: Raster cells of datasets R and S

When the data is stored in vector representation, one may be interested in either the number

of the joined objects or the area size of the joined objects. This paper focuses on the latter case and provides a solution to a faster estimation of the size of the joined area. Figure 2 illustrates an example of two vector datasets. Let A be the minimum bounding rectangle (MBR) that contains all polygonal objects in dataset 1 and let B be the MBR that contains all the polygonal objects in dataset 2. When A and B intersect, we are interested in knowing the probability that the actual polygons in A join with the polygons in B. Let the dotted rectangle C be the intersecting region of A and B where intersections of the datasets can occur. Assuming that the data polygons are uniformly distributed in A and B, the density of dataset 1 in C is the same as the density of dataset 1 in A. Similarly, the density of dataset 2 in C is the same as the density of dataset 2 in B. Given the two datasets' densities (density in A and density in B) and their overlapping area size (the size of rectangle C), we can calculate the join probability of the two datasets and estimate the size of their joined area.

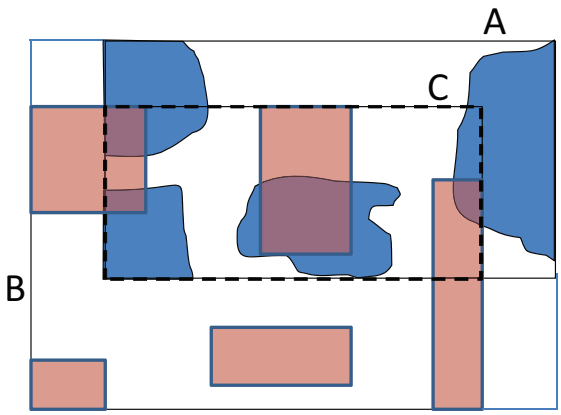


Figure 2: Dataset1 intersecting dataset2 in region C

Our sampling join method is based on stratified random sampling [Bae et al.(2009)] and performs joins using incremental samples to calculate estimates of the final answer of spatial joins. This method provides bounded confidence intervals with a given probability.

Our interactive framework combines the two statistical methods to speed up the process of obtaining estimates of spatial join results. This paper expands upon our previous work [Bae et al.(2010)]

which provided an interactive framework for raster joins. In this paper, we present the join probability for both raster and vector spatial joins and provide the equations and their proofs. We also provide an architectural design of the proposed framework and discuss challenges of implementing the framework in existing databases. Experimental results for both synthetic and real datasets demonstrate the efficiency of the proposed methods.

The remainder of this paper is organized as follows: In Section 2, we discuss related work. Section 3 presents a new interactive framework for spatial joins and proposes an architecture for this framework. In Section 4, we introduce the join probability and provide formulae and their proofs. We then present two statistical methods: probabilistic join and sampling-based join. The proposed methods are experimentally evaluated in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

GIS are used to store, manipulate and retrieve large spatial data that are used to describe the geometry and location of various types of geographic phenomena [Medeiros and Pires(1994)]. Spatial indexes are used to efficiently support geographic or spatial queries that depend on spatial relationships between data items [Larson(1996)]. Such spatial relationships include intersection, containment, and adjacency. In this paper we focus on proposing a framework for estimating intersections between two datasets, the number of intersected cells of raster datasets and the size of intersected area of vector datasets. This work can be further extended to support multiway joins where more than two datasets are joined.

The quad-tree is a very popular hierarchical data structure for the representation of binary images and maps, and it is commonly used in spatial databases [Samet(1990), Vassilakopoulos and Manolopoulos(1
i.e., indexing for query processing, and optimizing decomposition. Our work assumes raster datasets
are indexed by quad-trees. Quad-tree based sampling has been proposed in [Olken(1993), Vassilakopoulos and M
In [Vassilakopoulos and Manolopoulos(1997)], the authors presented the analysis of four different

sampling methods proposed by [Olken(1993)]. They applied sampling algorithms to specific quad-tree implementations to obtain approximate aggregate query results. They proposed two models for analyzing the sampling cost while our incremental sampling approach provides a faster approximation of the join result with a bounded confidence interval.

Vector datasets are usually indexed using R-trees [Guttman(1984)]. An R-tree is a height balanced tree structure adapted from the B-tree to support spatial data. An R-tree stores the minimum bounding rectangles (MBRs) of objects. When performing a window query on an R-tree indexed dataset, all rectangles that intersect the query region are retrieved. This is done in a recursive way starting from the root and following the paths down to the leaf level. A spatial join computes the pairwise intersection of all data objects in two spatial datasets. Many blocking spatial join algorithms based on R-trees have been proposed with perhaps two of the most common being found in [Brinkhoff et al.(1993a), Papadias et al.(1999)]. When performing spatial join on two R-tree indexed datasets using traditional algorithms such as these found in [Brinkhoff et al.(1993a), Papadias et al.(1999)], all data level MBRs in the R-tree of the outer dataset which intersect the data MBRs in the R-tree of the inner dataset are retrieved as pairs into a candidate set (filtering step). Then the pairs are pruned if their actual data (polygons) do not overlap (refinement step). This is done in a blocking recursive way, starting from the root and following the paths down to the leaf level. In this paper, we calculate the join probability of the two datasets and estimate the size of the joined (overlapped) area by joining the MBRs positioned at the same level in the R-trees of the two datasets while the granularity increases by time or by a user input.

Spatial joins are more complex than relational joins because spatial datasets tend to be much larger than relational datasets and because the spatial joins are specified by intersections and not equality. One common raster data spatial join technique is map overlay [Tveite(1997)]. Map overlay is straightforward when the input rasters have the same cell boundaries. The resulting raster can be obtained cell by cell from the originals using the relevant operations on the cell values. Since GIS can reach gigabytes and possibly terabytes in size, full layer overlays could take hours and even

days to complete. This necessitates the need for approximation techniques. The significant body of work on relational database join approximations can not be directly applied to spatial databases. In [Zimbrão and de Souza(1998), Azevedo et al.(2006)] the authors presented an approximation technique of vector spatial joins. First they converted vector data to raster format and filtered the possible joined pairs using the Four Color Raster Signature in [Zimbrão and de Souza(1998)] and the Three Color Raster Signature in [Azevedo et al.(2006)]. They combined progressive and conservative approximations [Brinkhoff et al.(1993b)] in a single approximation to speed up the filtering step in identifying intersecting polygons. Their problem is to estimate selectivity of vector spatial joins using raster-based techniques. Their proposed techniques motivated us to obtain the join probability of raster datasets and to extend the solution of raster data to the join probability of vector datasets.

A non-blocking parallel spatial join algorithm was proposed in [Luo et al.(2002)]. Their proposed algorithm runs in phases; in the first phase the authors used a duplicates avoiding algorithm to partition the spatial objects of the datasets that are approximated by their MBRs. Then partitions are distributed and indexed using R-trees among several nodes to be joined. In the second phase, bucket pairs of partitions of both datasets are joined. However, to provide approximate answers to users, buckets are joined randomly and non-repeatedly.

Histogram-based methods to calculate special join estimates was proposed in [An et al.(2001)]. Such methods result in fast query estimates, however, these methods do not provide any bounds on the error of the estimations. The error bound they provide was determined through a series of experiments. Also the histogram methods provide an estimate of the number of joins, but they do not provide actual join results. They estimated the number of the joined objects for vector data while our proposed framework estimates the area size of joined objects.

Online query processing was first presented by [Hellerstein et al.(1997), Haas and Hellerstein(1999), Hellerstein et al.(2000)]. The authors proposed the "ripple join", a non-blocking join method, for relational databases. The "ripple join" was used to calculate running aggregates to provide the

user with estimates that are bounded by confidence intervals. They also showed how the user can control the query execution process through an interface. The main objective was to minimize the time needed to obtain an approximate query answer instead of computing the exact answer.

The idea of incremental sampling techniques on vector data was proposed in [Bae et al.(2006), Bae et al.(2009)] to provide interactive spatial join processing. The authors proposed two R-tree based sampling methods that were used to incrementally refine the estimated join result while providing a bounded confidence interval. In addition to providing the estimated results, they also displayed the actual intermediate intersections of the jointed datasets. Their approach was applied for vector-based data. In this paper, the proposed sampling method in our framework for raster data follows the same framework but using quad-trees instead of R-trees and with a more sophisticated sampling method. Both sampling methods [Bae et al.(2009)] and histogram methods [An et al.(2001)] provide an estimate on the number of joins while our proposed framework for vector data estimates the size of the joined area between two datasets.

[Cheng et al.(2003)] studied probabilistic query evaluations for uncertain continuously changing data in relational databases. In [Cheng et al.(2006)], the authors propose probabilistic join over uncertain data. They provided techniques to answer queries that return results that have probability exceeding a given threshold. To the best of our knowledge, our probabilistic join method is the first attempt to apply probabilistic approaches to estimate raster-based spatial joins and to estimate the joined area for vector spatial joins.

3 An Interactive Framework for Spatial Joins

In this section, we present a query processing framework for spatial joins on raster and vector datasets. We also present an architectural design for the proposed framework and discuss its design and implementation challenges.

3.1 Framework Overview

Our design of the spatial join framework is based on the following two statistical methods:

- Probabilistic Join (PJ): PJ is based on the augmented tree data structures, quad-trees for raster data and R-trees for vector data. The join probability is calculated using data density of the joined tree nodes. The data density is defined as the ratio of number of non-zero data cells to the total number of data cells in each node of the quad-tree for raster data and defined as the ratio of the total area of data objects to the area of the MBR in each node of the R-tree for vector data.

- Incremental Stratified Sampling Join ($ISSJ$): Using quad-trees for raster data and R-trees for vector data, overlapping regions are used to filter candidate pairs in order to speed up the joining process. $ISSJ$ is based on stratified random sampling from indexing trees and spatial joins of the incremental samples are conducted to estimate the final answers of spatial joins.

The proposed framework consists of three main processes: Probabilistic Join (PJ), result visualization and Incremental Stratified Sampling Join ($ISSJ$). The main idea is to use PJ and a visualization technique to allow users to discover interesting pairs of datasets and areas for further data exploration. Once users identify interesting pairs of datasets, they can have the system perform $ISSJ$ in order to produce tighter running estimates of join results, or users can have the system complete the full tree join (full quad-tree join for raster data and full R-tree join for vector data) to obtain the exact answer. Figure 3 shows the overview of the framework for raster data.
1) *Probabilistic Join* (PJ): Given the user's input datasets, all higher level nodes (from level 0 to level 3 in our experiments) of the two datasets' quad-trees are loaded in memory. Then the join probability of each pair of the corresponding nodes is obtained from a look-up table (step 2).
2) *Visualization and user interface in* PJ: Based on a visualized result of probabilistic joins, the user can identify interesting pairs of datasets (step 3 and step 4).
3) *Incremental Stratified Sampling Join* ($ISSJ$): $ISSJ$ starts the incremental sampling process

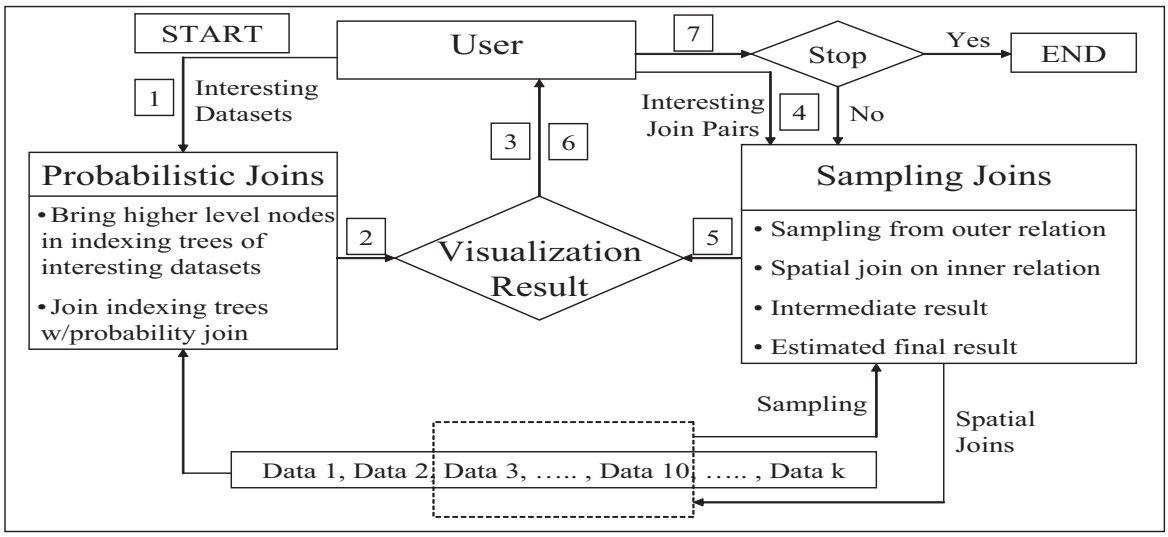


Figure 3: Interactive spatial join framework

with the interesting pairs. Samples (non-zero cells) are randomly chosen from the outer relation R using stratified random sampling, then spatial joins on the corresponding cells of the inner relation S are performed. The number of joined cells in each step is used to calculate a running estimate and a confidence interval for the final result. The calculated running estimate and confidence interval are combined with the intermediate result into a query result through a visualization process (step 5).

4) *Visualization and user interface in ISSJ*: The query result is reported to the user. The user can stop the query process if the given confidence interval is sufficient or if the user sees satisfying trends from the visualized actual join locations (intermediate result); otherwise, each step of the process is repeated in an incremental manner to calculate new estimates until a desired confidence interval is achieved. Thus, the time to get join estimates needs to be compared to the time required for the full quad-tree join (step 6 and step 7).

The framework of vector spatial joins is similar to that of raster joins and the details are discussed in Section 4.1.3. Figure 4 shows an example of spatial query processing interface for vector datasets using the proposed framework.

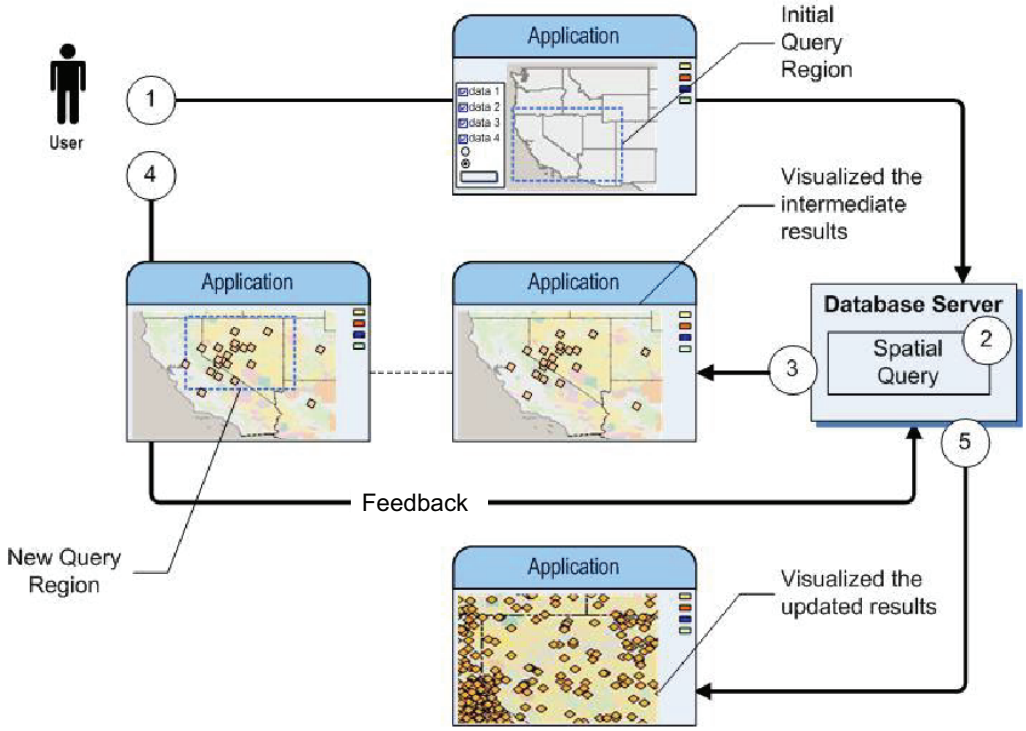


Figure 4: Spatial query processing of vector datasets

3.2 Architecture Design and Challenges

Existing database (DB) engines execute long running spatial queries in blocking manner that has many disadvantages such as long turnaround time for final results, inability to view intermediate results, and inability to alter the submitted query to explore more interesting data. One solution is to stop the running query and resubmit a new query based on the current result; this results in discarding already computed results and multiple iterations of resubmitting new queries until the user obtains the desired result. On the other hand, database engines could be designed to

accept users' inputs (feedback) while the queries are running in order to allow the users to explore more interesting data. The goals of our proposed architecture are: 1) enable existing DB engines to support interactive queries, 2) enable the DB users to alter query plans by providing feedback to the running query instances on the interesting regions or to cancel the ongoing query, 3) by combining the above two items, the user can observe the pattern or trend of the results and interactively redirect the query to interesting regions.

The proposed architecture is shown in Figure 5. The query interface continues to accept the users' input even after the initial query is submitted and while it is being processed. This can be accomplished using the same channel (connection) used for submitting the initial query. An alternative is to let the DB server issue a query identifier (QID) which notifies the query plan evaluator about the changes that are passed through parameters. The intermediate results can be returned by the existing channel or by other means (such as User Datagram Protocol (UDP) messages), which can be an optional parameter to the query. The application environment in Figure 5 shows the query initiation, the query alterations, and the query's intermediate results as independent components for clarity. The query initiators show the various user environments (which may consist of different applications). The results returned to the user can be segmented so that it takes into account the network packet size, thus the network utilization is neither degraded nor is it burdened. In fact, it may be a better utilization of the bandwidth since the data is returned as it is processed.

There exist design and implementation challenges to implement the proposed architecture in both existing and new DB engines. Traditional DB engines use *request-response* protocols; i.e. once a query is submitted by a client to a server, no further communication occurs other than notifications, a response is computed by the server and sent back to the client. During query processing, the connection to the DB engine is maintained until the result set is returned to the user. Some DB engines (e.g., Postgres) provide mechanisms to connect back to the server in case of a connection failure.

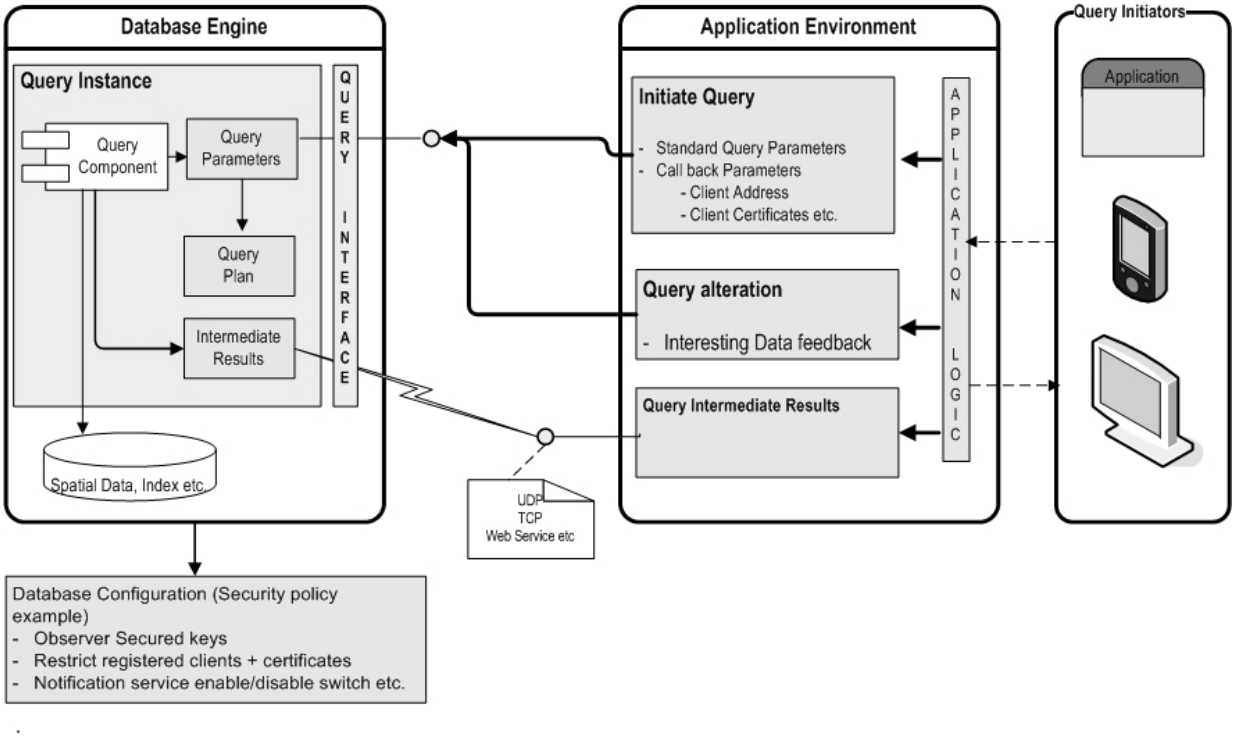


Figure 5: Architecture of interactive spatial query processing

4 Spatial Join Algorithms using Statistical Methods

Statistical approaches can be used to calculate estimations of certain characteristics of datasets in spatial databases and GIS. Our approach uses information associated with data density, samples drawn from the population and distribution of the samples. Sampling methods are used to estimate the final result from a subset (samples) of the data, providing a bounded confidence interval. On the other hand, probabilistic methods are used to show a faster result that is not bounded.

In this section, we first define the join probability and present the equations and their proofs. We then present two proposed methods, *Probabilistic Join* (PJ) and *Incremental Stratified Sampling Join* ($ISSJ$). Augmented quad-trees with non-zero data cells are used in our algorithms for raster datasets. For vector datasets, augmented R-trees with the total area of data objects are used.

4.1 Probabilistic Joins (PJ)

4.1.1 Join Probability

The fundamental question that we address here is: *Given two subsets A, B of a set X, what is the probability $p = P(A \cap B \neq \emptyset)$ that A intersects B?*

When X is a finite set, $|X| = n$, $|A| = a$, $|B| = b$, the answer is

$$p \quad = \quad 1 - \frac{\binom{n-a}{b}}{\binom{n}{b}}, \tag{1}$$

because $A \cap B = \emptyset$ precisely when the b-element subset B is contained in the $(n-a)$-element subset $X \setminus A$.

When X is infinite, we run into well-known probability paradoxes, and the answer depends on the way the experiment is conducted and on the way we measure the size of sets. Nevertheless we can give an exact answer if we restrict the shapes of A, B, X to (d-dimensional) intervals.

Theorem 4.1 *(Join probability for 1-dimensional intervals)*
Let $X = [0, 1]$, and let A, B, be randomly chosen intervals in X of length a, b, respectively. Then

the probability that $A \cap B \neq \emptyset$ depends only on a, b, and can be calculated by

$$p(a,b)_1 \;=\; \frac{1}{(1-a)(1-b)} \int_0^{1-b} (\min\{x+b, 1-a\} - \max\{0, x-a\})\ dx. \tag{2}$$

Proof: Let $A = [a_l, a_h]$, $B = [b_l, b_h]$ be randomly chosen subsets of X such that $|A| = a$, $|B| = b$. Equivalently, let a_l be chosen at random in $[0, 1-a]$ and let b_l be chosen at random in $[0, 1-b]$ (with $a_h = a + a_l$, $b_h = b + b_l$). Then $A \cap B \neq \emptyset$ if and only if $a_h = a + a_l \geq b_l$ and $a_l \leq b_h = b + b_l$, which happens if and only if $\max\{b_l - a, 0\} \leq a_l \leq \min\{b_l + b, 1-a\}$. Thus, for a fixed randomly chosen $b_l \in [0, 1-b]$, the two intervals intersect if and only if $a_l \in [\min\{b_l + b, 1-a\}, \max\{b_l - a, 0\}]$. The formula (2) follows.

Before we proceed to generalize Theorem 4.1 to 2-dimensional intervals, note that the probabilities obtained from the discrete and continuous formulae (1), (2) differ in general, even if we assume that the sets A, B form 1-dimensional "intervals" in the discrete case. For instance, suppose that $|X| = 6$, $|A| = 3$ and $|B| = 2$. Then the formula (1) yields

$$p \;=\; 1 - \frac{\binom{3}{2}}{\binom{6}{2}} = \frac{4}{5}.$$

If we assume that X is discrete 1-dimensional and A, B are intervals in X, then it is easy to check that $p = \frac{7}{10}$. Finally, if we assume that X is a continuous 1-dimensional interval, we can normalize (in order to apply Theorem 4.1) to $|X| = 1$, $|A| = \frac{1}{2}$, $|B| = \frac{1}{3}$, and then Theorem 4.1 yields

$$\begin{aligned} p \;&=\; \frac{1}{\left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)} \int_0^{1-\frac{1}{3}} \min\left\{x + \frac{1}{3}, 1 - \frac{1}{2}\right\} - \max\left\{0, x - \frac{1}{2}\right\}\ dx \\ &=\; 3\left(\int_0^{\frac{1}{6}} \left(x + \frac{1}{3}\right)\ dx + \int_{\frac{1}{6}}^{\frac{1}{2}} \frac{1}{2}\ dx + \int_{\frac{1}{2}}^{\frac{2}{3}} (1-x)\ dx\right) \\ &=\; 3 \cdot \frac{11}{36} = \frac{11}{12}. \end{aligned}$$

Theorem 4.2 *(Join probability for 2-dimensional intervals)*
Let $X = [0,1]\times[0,1]$, and let A, B be two randomly chosen rectangles in X of area a, b, respectively. Then the probability that $A \cap B \neq \emptyset$ depends only on a, b, and can be calculated by

$$p(a,b)_2 \quad = \quad \frac{1}{(1-a)(1-b)} \int_a^1 \int_b^1 p(a_1,b_1)_1 \cdot p\left(\frac{a}{a_1}, \frac{b}{b_1}\right)_1 \, da_1 db_1. \tag{3}$$

Proof: To choose an $a_1 \times a_2$ rectangle A of area a in X at random, we must first randomly choose $a_1 \in [a,1]$ and let $a_2 = a/a_1$. Similarly for the $b_1 \times b_2$ rectangle B of area b in X. Since $A \cap B \neq \emptyset$ if and only if the 1-dimensional projections of A and B (onto the x-axis and the y-axis) intersect, formula (3) follows.

Theorem 4.2 can be easily generalized to $d > 2$ dimensions, but we do not pursue this generalization here.

We now turn our attention to the expected size of the intersection of two randomly chosen subsets of prescribed sizes. Let $X = \{1, \ldots, n\}$, and let A, $B \subseteq X$ be two randomly chosen subsets of X with $|A| = a$, $|B| = b$. For $1 \leq i \leq n$, let x_i be the random variable satisfying $x_i = 1$ if $i \in A \cap B$, and $x_i = 0$ otherwise. Let $x = \sum_{i=1}^n x_i$. The expected size of $A \cap B$ is then the expected value of x, which can be calculated as $E[x] = E[\sum_{i=1}^n x_i] = \sum_{i=1}^n E[x_i]$. Out of the $\binom{n}{a}$ subsets of X of size a precisely $\binom{n-1}{a-1}$ contain a fixed element i. We therefore have

$$E[x] = \sum_{i=1}^n E[x_i] = \sum_{i=1}^n \frac{\binom{n-1}{a-1}\binom{n-1}{b-1}}{\binom{n}{a}\binom{n}{b}} = \sum_{i=1}^n \frac{ab}{n^2} = \frac{ab}{n}.$$

We conclude that the expected size of $A \cap B$ is

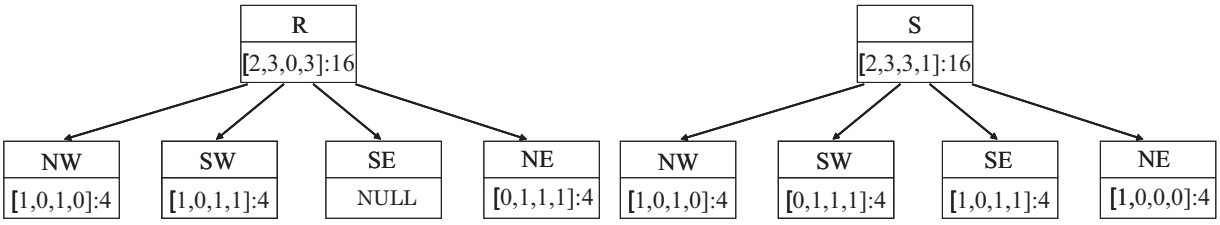
$$\frac{ab}{n}. \tag{4}$$

When X is a subset of the Euclidean plane of area n (not necessarily an integer), and A, B are randomly chosen subsets of X of areas a, b, respectively, then the expected area of $A \cap B$ is also

obtained by (4), which can be justified by a probabilistic argument.

4.1.2 Raster Spatial Joins using Join Probability

Augmented quad-tree data structure is used in PJ and $ISSJ$ for raster spatial joins. Specifically, we augment nodes to include the total number of non-zero data cells of the subtree below. The proposed statistical approaches use these augmented quad-trees for obtaining information associated with the population. Figure 6 (a) and (b) show augmented quad-trees of the raster data set examples in Figure 1. The nodes of the quad-trees are displayed in counter clock-wise order starting from the north-west quadrant. In the framework, all datasets are indexed by augmented quad-trees.



(a) Augmented quad-tree for dataset R (b) Augmented quad-tree for dataset S

Figure 6: Examples of augmented quad-trees of datasets R and S

In PJ, the augmented value (number of non-zero data cells) of each node of two given datasets is used to calculate the join probability and the expected number of joined data cells for each pair of subregions in the two joined datasets. PJ accesses nodes from the top to the bottom; hence, PJ is a top-down approach. PJ does not need to access all levels of a quad-tree to calculate an estimate. It is sufficient to access only a small number of top levels as demonstrated by the experimental results. Thus, it can greatly reduce the time-consuming disk I/O operations in practice. The number of levels to be accessed can be set as a parameter. The greater the number of levels accessed, the more accurate the estimation can be. However, this results in a larger number of I/Os. In our experiments (Section 5), we set the number of levels to 4 resulting in only 64 nodes needed in memory. Thus, it is practical to store required top level nodes of quad-trees for all joined datasets

in memory. Also our experimental results of PJ on synthetic and real datasets show the error bound is reasonably tight, e.g., a 9% error for 4^{th} level join (Section 5).

4.1.3 Vector Spatial Joins using Join Probability

We expand the join probability formulae in Section 4.1.1 for estimation of vector spatial joins. Unlike raster data join where the corresponding nodes of quad-trees have the same geographic coordinates, MBRs of the nodes at the same level in R-trees may have different geographic coordinates resulting in an overlapping rectangular region defining the MBRs of the two joined datasets. Based on the data density information of the current nodes to be joined (a pair of MBRs) and their intersecting area, information about their spatial join can be calculated. First we augment each node of the R-trees with the total area of the actual data of the subtree having that node as its root. When the data object is added to the database, we calculate its MBR. We also augment each node with the area of the object it encloses. Then the augmented information (density) can be used for calculating the join probability of pairs of nodes and the expected joined area size. We use the formula (3) for the join probability of pairs of nodes and the formula (4) for the expected value.

PJ accesses the R-trees' nodes of the given datasets from the top to the bottom starting with the root nodes. An example of two R-trees indexed datasets to be joined is shown in Figure 7. Each node of the R-trees is augmented by its data density. Initially, the overlapping area of the two MBRs of the roots of the R-trees with the calculated join probability is displayed to the user (1^{st} level join). Similarly pairwise join results of the MBRs at the 2^{nd} level of the R-trees with the corresponding join probability are returned to the user. Figure 8 illustrates the spatial joins of the two example datasets: (a) 1^{st} level (root node level) join and (b) 2^{nd} level join. The bold-lined rectangular regions represent the possible joined area at each level. All possible joined regions with their join probabilities as well as the estimation of the size of the joined area will be displayed to the user.

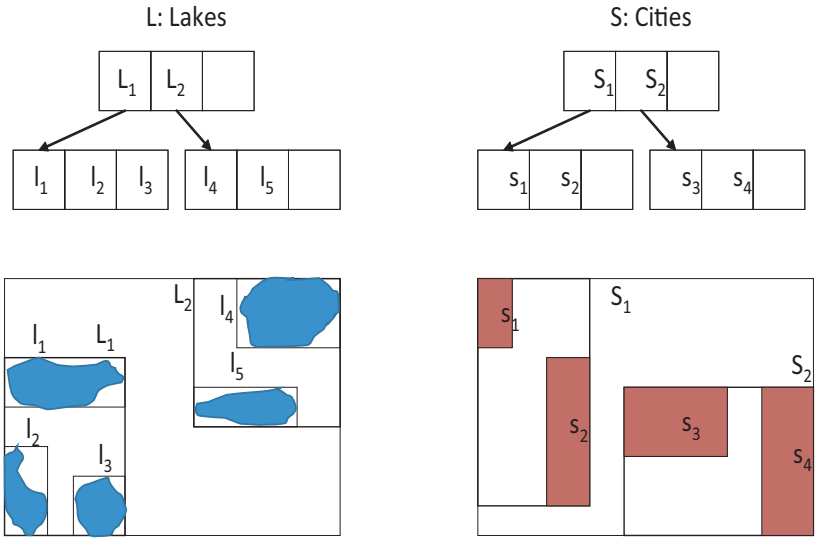
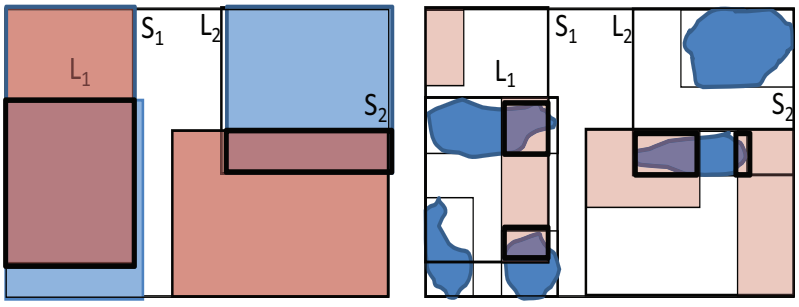


Figure 7: Lakes and cities datasets and their R-trees



(a) 1^{st} level join (b) 2^{nd} level join

Figure 8: An example of vector spatial join

Based on the visualized result of the probabilistic joins, the user can specify interesting region by drawing a rectangle (window query) around the interesting area, or by picking intersecting rectangles of the MBRs as the interesting area. Our algorithm would accordingly eliminate other regions and process only the user's input of interesting region(s). This enhances the overall query processing, resulting in a faster user driven interface. The user can also decide to continue the probabilistic join on R-trees or change the method either to sampling or to full R-tree join if the interesting region is small enough. For the former case, the algorithm drills down to the next level (3^{rd} level) nodes' join and provides a new visualized result. The result should include a better estimation of the second level's joined area (7% - 10% error in our experiments) and an even better estimation of the root level joined area (about 3% - 6% error).

4.2 Incremental Stratified Sampling Join ($ISSJ$)

We study the stratified random sampling for spatial joins without replacement by adopting the framework for vector data proposed in [Bae et al.(2009)]. Each sampling is conducted in an incremental manner and the performance is evaluated with varying datasets and buffer sizes. A weighted random sampling method, *Acceptance/Rejection* [Olken(1993)], is used in our proposed framework. [Bae et al.(2009)] proposed incremental sampling methods for vector data spatial joins. Any of these methods can be utilized in our framework for vector data spatial joins, hence we omit the discussion of the sampling method for vector data. In this section, we present the *Incremental Stratified Sampling Join* ($ISSJ$) method for raster data spatial joins.

In $ISSJ$, stratified random sampling is used to estimate the final answer of spatial joins. An accuracy guarantee is provided in the form of error bound confidence intervals. In contrast to PJ, $ISSJ$ is performed on sampled leaf level data cells. Although fewer I/Os are required in $ISSJ$ compared to a full quad-tree join, obtaining a reasonable confidence interval requires a significantly greater number of I/Os compared to PJ.

4.2.1 Stratified Random Sampling on Raster Data

Stratified random sampling is chosen for raster spatial joins because its property matches the property of quad-trees that provides systematic decomposition of a space with no overlaps between subregions. In stratified random sampling, the given region (population of all data cells) is divided into a number of non-overlapping subregions called strata. Then each stratum contains a set of raster data cells. Stratified random sampling can result in smaller error bounds on an estimation and can reduce the sampling cost [Serfling(2002), Bae et al.(2009)].

In our algorithm for raster data, stratification is based on non-overlapping geometric forms such as rectangles (nodes at each level). We define the internal nodes of the quad-tree for a given level as strata, i.e., the second level nodes of quad-tree are used as strata in our experiments. We assume that the strata is pre-defined in our experiments. Algorithm 1 describes the $ISSJ$ algorithm for raster data and the notation used in the algorithm is summarized in Table 1.

Notation	Description
N	the size of population (the total number of data cells in R)
k	the total number of strata
ST_i	stratum i, where $i = 1, .., k$
N_i	the total number of data cells in stratum i (ST_i), where $i = 1, .., k$
n_{init}	the initial incremental sample size for a sampling step
n_s	the sample size for a sampling step
S	the incremental sample size for the current sampling step
n_i	the sample size of stratum i for a sampling step, where $i = 1, .., k$
s_i	the incremental sample size of strata i for a sampling step, where $i = 1, .., k$
I	the current total number of joined cells in a sampling step
I_i	the current total number of joined cells in a sampling step for stratum i, where $i = 1, .., k$
C_I	a confidence interval
EV	an estimate of the total number of joined cells

Table 1: Notation used in $ISSJ$

Samples (non-zero cells) are then randomly chosen from each stratum by conducting simple random sampling. The incremental sample size of each stratum s_i, i=$1, .., k$, is calculated using n_{init} for every sampling step, and it is proportional to the total number of non-zero cells within that stratum. The sum of the incremental sample sizes of all strata is the value of the incremental

sample size denoted by S for the current step of the sampling. The total sample size n_s and the sample size of each stratum n_i, i=1, .., k, are then updated. If the value of the chosen data cell is 1, searching the corresponding joined cell of the inner data set is performed in the quad-tree of the inner data set (line 15 of Algorithm 1). If the value of the corresponding cell is 1, then the two data cells join. For each stratum, we obtain the number of joined cells, and this number is used to calculate the estimate and confidence interval for the corresponding stratum. The sum of the joined cells of each stratum is the current intermediate result, and the estimates and confidence intervals of all strata are combined for an estimate and a confidence interval of the final answer. The user can stop the query process if the given confidence interval is sufficient, otherwise the process repeats.

4.2.2 Estimates for Stratified Random Sampling on Raster Data

To provide bounds on the accuracy of $ISSJ$, we incrementally calculate the current estimate with a confidence interval. The estimates and confidence intervals of $ISSJ$ are based on population proportion and *the Central Limit Theorem* (CLT) [Haas(1997), Serfling(2002)]. We use the binomial probability distribution [Serfling(2002)] for statistics of $ISSJ$. In $ISSJ$, the population is the number of non-zero cells of the outer relation R and $\hat{p}$ is the fraction of the elements in the sample that possess the characteristic of interest ("join" in our algorithm). Hence $\hat{p}$ is the fraction of cells in the sample that joins with the corresponding cell of the inner relation S. Confidence intervals depend on the size of samples and the distribution of the sample space (i.e., *Student t-distribution*).

Let N be the size of population (total number of non-zero cells of the outer datasets) and n_s be the sample size for a sampling step. If N_i is the number of non-zero cells in stratum i, and n_i is the sample size for stratum i, then $N = \sum_{i=1}^{k} N_i$, and $n_s = \sum_{i=1}^{k} n_i$, where k is the number of strata. Let I_i be the total number of cells that join the corresponding cells of S in stratum i. The following equations are used for a sampling step for $ISSJ$:

Algorithm 1 $ISSJ(R, S, ST)$

1: $ST = \{ST_1, \ldots, ST_k\}$ // ST is a set of strata
2: $I_1, \ldots, I_k \leftarrow 0$ // current number of joined cells for stratum i
3: $n_s \leftarrow 0$; $n_{init} \leftarrow 30$ // sample size; initial incremental sample size for a sampling
4: $n_1, \ldots, n_k \leftarrow 0$; $s_1, \ldots, s_k \leftarrow 0$ // sample size; incremental sample size for stratum i
5: **repeat**
6: compute $s_1, s_2, \ldots, s_k$ for $ST_1, ST_2, .., ST_k$ // incremental sample size for each stratum
7: $S \leftarrow \sum_{i=0}^{k} s_i$ // the incremental sample size for the current sampling step
8: $n_s \leftarrow n_s + S$ // update the sample size
9: **for** $i = 1$ to k **do**
10: $n_i \leftarrow n_i + s_i$
11: **for** $j = 1$ to s_i **do**
12: $L \leftarrow$ choose a leaf from ST_i at random
13: $c_r \leftarrow$ choose a cell from L at random
14: **if** cell c_r's value is 1 **then**
15: $P_r \leftarrow$ the center point of the chosen cell c_r
16: $c_s \leftarrow$ findJoinedCell(S, P_r)
17: **if** cell c_s's value is 1 **then**
18: $I_i \leftarrow$ add 1
19: **end if**
20: **end if**
21: remove c_r from L
22: **end for**
23: remove L from ST_i if L is empty
24: **end for**
25: $I \leftarrow \sum_{i=0}^{k} I_i$
26: $C_I \leftarrow$ Compute a confidence interval w/all I_i and n_i
27: $EV \leftarrow$ Compute an estimate w/all I_i and n_i
28: report EV, C_I, and I
29: **until** C_I is sufficient to the user or all ST_i are empty

Estimator of the population proportion, where $\hat{p}_i = \frac{I_i}{n_i}$ and $\hat{q}_i = 1 - \hat{p}_i$:

$$\hat{p} = \frac{1}{N}(N_1\hat{p_1} + N_2\hat{p_2} + ... + N_k\hat{p_k}) = \frac{1}{N}\sum_{i=1}^{k} N_i\hat{p}_i. \tag{5}$$

Estimate variance of $\hat{p}$:

$$\hat{V}(\hat{p}) \quad = \quad \frac{1}{N^2}\sum_{i=1}^{k} N_i^2 \left(\frac{N_i - n_i}{N_i}\right)\left(\frac{\hat{p_i}\hat{q_i}}{n_i - 1}\right) \tag{6}$$

Confidence interval:

$$E = t_c\sqrt{\hat{V}(\hat{p})}, \tag{7}$$

where t_c is the critical value for confidence level c taken from a Student t-distribution. Equations (5), (6) and (7) are valid for the incremental stratified sampling process. The proof of incremental equations can be found in [Bae et al.(2009)].

5 Experiments

In this section, we present experimental results of the *Probabilistic Join* (PJ) and *Incremental Stratified Sampling Join* ($ISSJ$) with synthetic and real GIS datasets in both raster and vector formats. The performance of PJ and $ISSJ$ are compared with each other as well as with the full tree join. In this paper, we present only the most illustrative subset of our experimental results due to space consideration. Similar qualitative and quantitative trends were observed in all other experiments.

5.1 Experimental Evaluation of Raster Spatial Joins

5.1.1 Datasets and Experimental Methodology

In our experiments, we consider both synthetic and real raster datasets shown in Table 2 and Table 3. We generated four sets of uniformly distributed data and four sets of exponentially distributed data (a mean of 0.3 and a standard deviation of 0.3). Our real datasets are from the 2001 and 2005 U.S. Geological Survey [USGS(2001, 2005)]: six datasets (datasets of minerals, stream sediments, water sediments, rocks, pluto geochemical sediments and unconsolidated sediments) from Arizona, Colorado, Oregon and Wyoming in the US. Each dataset was converted into raster format. In Table 2, the total number of data cells (pixels) is presented along with the total number of non-zero data cells and the data density for the synthetic datasets. Table 3 presents the information about water sediments datasets of the four states.

	synthetic datasets							
	uni1	uni2	uni3	uni4	exp1	exp2	exp3	exp4
# total cells	65536	65536	262144	262144	65536	65536	262144	262144
# N.E. cells	17325	28365	39120	48298	14256	24736	36290	45231
density	0.26	0.43	0.15	0.18	0.22	0.38	0.14	0.17
description	uniformly distributed data				exponentially distributed data			

Table 2: Synthetic raster datasets: N.E. = non empty data cells

	real datasets			
	AZ	CO	OR	WY
# total cells	65536	65536	65536	65536
# N.E. cells	10202	23030	23821	42321
density	0.26	0.43	0.15	0.18
description	water sediment datasets from USGS			

Table 3: Real datasets (water sediment): N.E. = non empty data cells

It is necessary that all datasets are indexed by augmented quad-trees and that they have the same number of data cells as well as the same size of cells. Our experiments were conducted using the following parameters: Augmented quad-trees are implemented for PJ and $ISSJ$ while

nonaugmented quad-trees are used for the full tree join. The page size of the quad-tree was set to 4 Kbytes, resulting in 100 nodes and 64 nodes for the non-augmented tree and augmented tree, respectively. We performed comparisons assuming an LRU replacement policy with buffer sizes of 5%, 10% and 20% of the size of one of the joined datasets. For all presented results, the estimates and the corresponding confidence intervals are shown with a 95% confidence level.

5.1.2 Results of Raster Spatial Joins

First, we present the accuracy of the expected number of joined cells for raster datasets using the formula (4) discussed in Section 4.1.1. The expected numbers of joined cells were compared with the total numbers of actual joins. We randomly selected two corresponding nodes from the quad-trees of two real datasets. We checked the occupancy rates (non-zero data cells/total data cells) in the two chosen nodes and obtained the expected numbers of join cells. We repeated this process for varying sizes of sample pairs: 5%, 10%, 20% and 50% of the total quad-tree nodes. We ran the experiment 10,000 times with each of the sample sizes and calculated the average. In Table 4, we show the results of joining unconsolidated sediments dataset with minerals dataset in CO. The table entries are actual number of the joined cells error values, thus, for example, an error of 0.1060 is a 10.60% error.

sample size	actual join	expected number of joined cells (error)
5 %	54	48 (0.1060)
10 %	109	99 (0.0917)
20 %	218	197 (0.0963)
50 %	545	494 (0.0936)

Table 4: Accuracy of the expected number of joined cells w/ different sample sizes

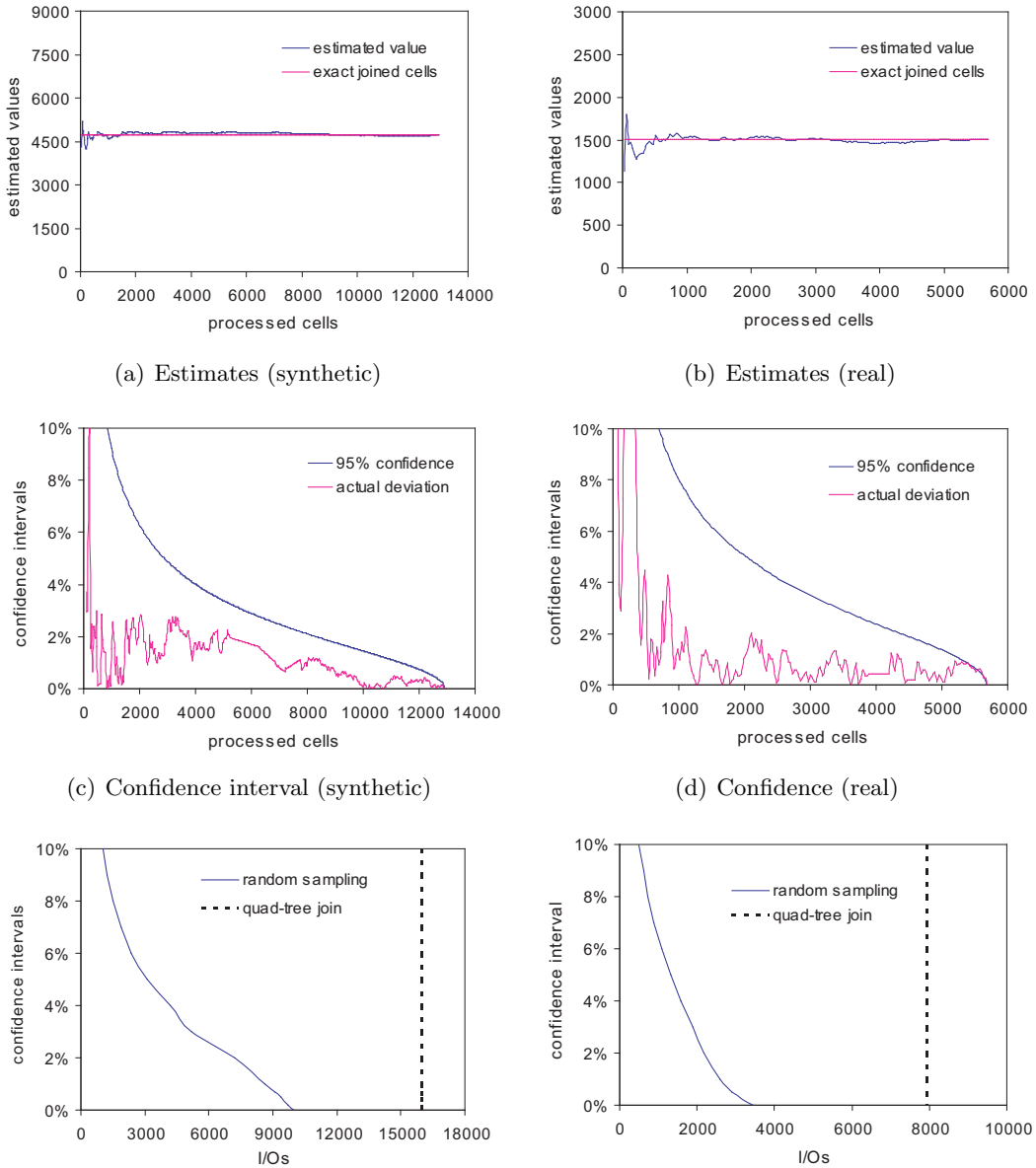
To evaluate the quality of the "big picture" visualization obtained by PJ, we calculated the expected number of joins using the 4^{th} level tree nodes. When using the 4^{th} levels of two quad-trees, only 64 subregions are joined. As a result, users can obtain the approximate result visualization in near instantaneous time with a truly interactive manner. For the real datasets we compared PJ and

joined datasets	real datasets				synthetic datasets	
	AZ	CO	OR	WY	group1	group2
average diff.	0.0060	0.0087	0.0049	0.0058	0.0032	0.0024
minimum of max. diff.	0.0047	0.0038	0.0045	0.0014	0.0018	0.0015
maximum of max. diff.	0.1208	0.0973	0.0849	0.1143	0.0410	0.0312
average max. diff.	0.0329	0.0237	0.0214	0.0199	0.0201	0.0182
average error of estimates	0.1105	0.0729	0.0629	0.0904	0.0324	0.0229

Table 5: Accuracy of the expected number of joined cells for raster datasets

$ISSJ$ for all fifteen possible pairwise joins of the six datasets from each state. We grouped the synthetic datasets into two: group 1 (uni1, uni2, exp1, exp2) and group 2 (uni3, uni4, exp3, exp4). We computed all possible six pairwise joins of each of the two groups. In Table 5, we present the average differences in the join density. The maximum difference illustrates how far the estimation is from the actual answer for each test indicating an upper error bound. In order to provide the performance of our methods, we present both the minimum of maximum difference (the smallest value of the upper bound error) and the maximum of maximum difference (the largest value of the upper bound error) as well as the average maximum difference (average of the upper bound error). Finally, we calculated the average error in the expected number of joins of all the pairwise joins. As can be seen, PJ is reasonably accurate in all the cases of both real and synthetic datasets. With real datasets, PJ resulted in less accuracy due to the scattered clusters found in the datasets. As shown in Figure 11, for the data we explored, these modest inaccuracies have little effect on the overall visual join-result appearance.

Next, we present the performance of $ISSJ$ compared to the augmented full quad-tree join. Figure 9 shows the result using the synthetic and real datasets (minerals $\bowtie$ unconsolidated sediments from Colorado). The estimates and confidence intervals are plotted versus the number of samples (non-zero data cells) processed as well as the exact answer. Figure 9 (a) and (b) show the estimated values of the final joins calculated by $ISSJ$ for the synthetic and real datasets, respectively. Figure 9 (c) and (d) present the confidence intervals for 95% confidence level. The results show how fast the confidence intervals converge. By showing the deviations from the actual joins, we demonstrate



(a) Estimates (synthetic)

(b) Estimates (real)

(c) Confidence interval (synthetic)

(d) Confidence (real)

(e) I/Os (synthetic)

(f) I/Os (real)

Figure 9: Estimates, confidence intervals for 95% confidence level and I/Os of *ISSJ*: synthetic (uni1 $\bowtie$ exp1) and real datasets (unconsolidated sediment $\bowtie$ mineral in CO)

that $ISSJ$ provides good estimates of the final answer. In Figure 9 (e) and (f), we show how fast an accurate estimation could be calculated compared to the time required for the full quad-tree join. For example, it took about 1900 I/Os to reach an estimate with a 5% confidence interval while 8,000 I/Os were required for the exact answer by using the full quad-tree join.

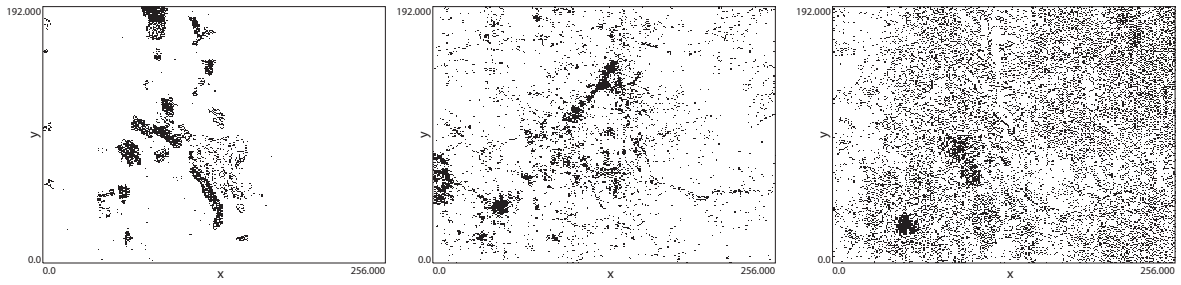
We next show how accurately the proposed approaches provide a "big picture" of the actual join. Figure 10 (a), (b) and (c) show three datasets for the state of Colorado: chemical sediments (P), minerals (Q) and water sediments (S). We compared the results of PJ with those of $ISSJ$ and the actual joins. For discrete values of join probability, we created a lookup table (20×20). Table 6 illustrates a portion of 2-d join probabilities such as entries from 2-d lookup table used in the experiments.

P	0.2	0.4	0.6	0.8	1.0
0.2	0.7683	0.9277	0.9903	1.0	1.0
0.4	0.9277	0.9937	1.0	1.0	1.0
0.6	0.9903	1.0	1.0	1.0	1.0
0.8	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0

Table 6: Example of a lookup table for 2-d join probability

The results of PJ and $ISSJ$ for $P \bowtie Q$ and $Q \bowtie S$ are presented as well as that of the actual join in Figure 11. The result from the top to the bottom corresponds to: $ISSJ$ with a 10% confidence interval (a), $ISSJ$ with a 5% confidence interval (b), actual joins (c) and finally PJ of the 4^{th} level nodes (d). PJ and $ISSJ$ with a 5% confidence interval provide a reasonably accurate approximation of the actual join. The results illustrate the accuracy of PJ using the formula (2) discussed in Section 4.1.1.

In Figure 12, we present I/O comparisons between PJ and $ISSJ$ varying the confidence intervals, as well as with the full nonaugmented quad-tree join (QT). All possible pairwise joins of the six datasets from Arizona were calculated and the number of I/Os were plotted for buffer sizes of 5%, 10% and 20% of the size of one dataset quad-tree. We also conducted the same experiment on all possible pairwise joins of the six datasets from Colorado. We plot the average total number of



(a) Chemical sediments (P) (b) Minerals (Q) (c) Water sediments (S)

Figure 10: Real datasets: Mineral resources in Colorado in the U.S.

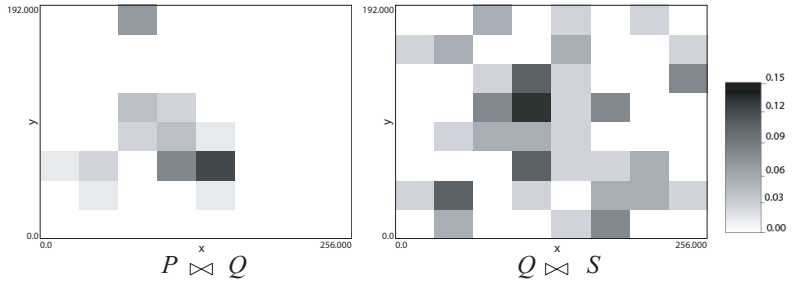
I/Os of each method averaged over all 15 pairwise joins. The results for PJ are on the left, then $ISSJ$ for confidence interval bounds of 10, 7, 5, 3, 2 and 1%, and finally the results for the full quad-tree join on the right. Note that the performance difference varying buffer sizes is very small since there is few re-visiting of the leaf nodes, and thus little opportunity to benefit from buffer caching exits.

The PJ algorithm resulted in up to two orders of magnitude fewer I/Os than QT for both datasets. The $ISSJ$ algorithm obtained a very reasonable confidence interval (e.g. 5%) with far fewer I/Os compared to QT. PJ is significantly faster than $ISSJ$ but does not provide correctness bounds. However, PJ does provide a good overall picture for the data explored.

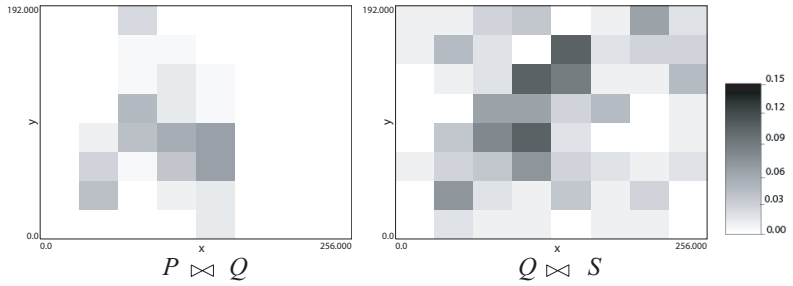
5.2 Experimental Evaluation of Vector Spatial Joins

5.2.1 Datasets and Experimental Methodology

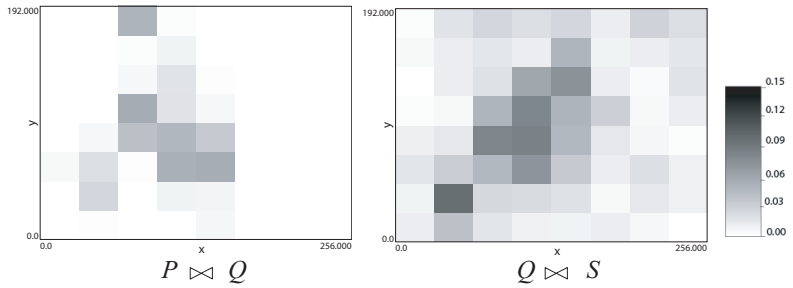
In this section, we evaluate PJ and $ISSJ$ for vector data using both synthetic and real datasets. Our synthetic datasets were generated using random function and hyper-exponential function for uniform and skewed datasets, respectively. For the uniform datasets, data is distributed uniformly and independently between 0 and 1. The data in the skewed datasets is independently drawn from a hyper-exponential distribution with the mean 0.3 and the standard deviation 0.5. The number of data polygons in each dataset varies from 50 K to 600 K increasing by 50 K. The notation u



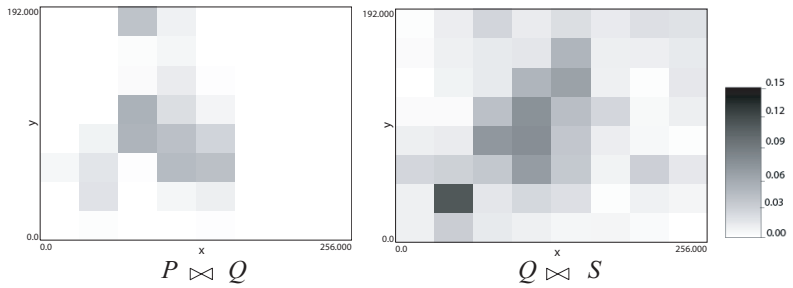
(a) $ISSJ$ w/10% CI



(b) $ISSJ$ w/5% CI

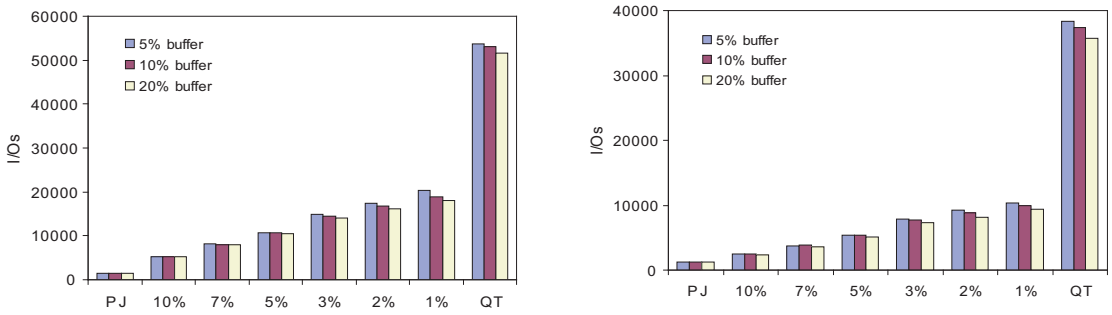


(c) Actual joins



(d) PJ

Figure 11: $ISSJ$ vs. actual joins vs. PJ for real datasets in CO



(a) Colorado (b) Arizona

Figure 12: Number of I/Os of PJ, $ISSJ$ and the full quad-tree join

and h are used to denote uniform and skewed datasets, respectively. The real datasets are from the 2001 and 2005 U.S. Geological Survey [USGS(2001, 2005)]: four datasets are chosen from Colorado and Wyoming, data1 (minerals), data2 (stream sediments) , data3 (unconsolidated sediments), and data4 (water sediments). The data density of our synthetic and real datasets varies from 11% to 32%. In our experiments, all possible join combinations of the datasets were conducted. Table 7 shows some of our datasets' R-tree structure information.

# nodes	synthetic datasets						real datasets			
	u200K	h200K	u500K	h500K	u600K	h600K	data1	data2	data3	data4
Tree levels	3	3	4	4	4	4	3	3	3	3
Index nodes	52	53	104	112	130	136	6	9	5	7
Leaf nodes	3615	3621	7241	7291	8675	8689	341	599	260	450

Table 7: Synthetic and real datasets

All the datasets are indexed by augmented R-trees. However, it is not necessary that datasets have the same size nor that their R-trees have the same height. When $ISSJ$ reaches the leaf level of the lower R-trees, this leaf level is used for joining with the lower levels of the other tree. The page size of the R-tree was set to 4 Kbytes with fan-out size of 100 and minimum capacity of 40. We used an LRU replacement policy and our experiments were conducted with the buffer size of 5% and 10% of the two R-tree total nodes.

5.2.2 Results of Vector Spatial Joins

In this section, we first show the accuracy of probabilistic joins on R-trees. Next, we evaluate the performance of PJ compared with an incremental sampling method presented in [Bae et al.(2006)] as well as with the full R-tree join algorithm [Guttman(1984)].

The formula (4) in Section 4.1.1 was used to calculate the expected joined area size of the two corresponding R-tree nodes. The joined area sizes of actual joins were compared with the estimated values of the joined areas. First we randomly selected two corresponding nodes (at the same level) from the two R-trees of the datasets. We then calculated their intersecting area, either 0 or a rectangle that defines the joined area. Finally, we obtained the joined area sizes using the data density (polygon data area/node MBR area) in the two chosen nodes. We repeated this process for varying sizes of samples: 5%, 20% and 50% of the total R-tree nodes. We ran the experiment 10,000 times with each of the sample sizes with different combinations of datasets joins: uniform $\bowtie$ uniform, skewed $\bowtie$ skewed, uniform $\bowtie$ skewed, and all combinations of real datasets joins.

Table 8 presents the averages of each group comparing the errors in the expected number of joined cells for quad-trees shown in Table 4. The errors of the expected joined area sizes for R-trees are lower than the errors on quad-trees. Note that each node of R-trees has about 70 children nodes on average while each quad-tree node only contains 4 children nodes. Hence many of the leaf level nodes can have more accurate information, which contribute to the sampling step, resulting in more accurate results.

sample size	errors on R-trees				errors on quad-trees
	uni $\bowtie$ uni	skew $\bowtie$ skew	uni $\bowtie$ skew	real datasets	
5 %	0.0689	0.0691	0.0739	0.0806	0.1060
20 %	0.0520	0.0515	0.0723	0.0633	0.0917
50 %	0.0495	0.0449	0.0604	0.0631	0.0936

Table 8: Accuracy of the expected joined area size w/different sample sizes

Next we evaluate the "big picture" visualization with different granularity levels provided by PJ. We calculated the expected joined area of the current level and that (those) of its parent(s)'s

level(s). We then compared these areas with the actual joined areas in Table 9. We present the average errors of all the pairwise joins among the synthetic datasets (from 50 K and 400 K) and the real datasets. PJ provides reasonably accurate estimation for the 2^{nd} level and 3^{rd} level joins in all the results of both synthetic and real datasets. The results show that the estimates of the second level's joined area result in 7% - 10% error and the estimates of the root level joined area which is the estimation of total joined region result in 3% - 6% error for both the synthetic and real datasets. With the real datasets, PJ resulted in less accuracy due to mainly the scattered clusters in the datasets and the size of the datasets. However, the "big picture" visualization of spatial join result may not be affected much by these modest inaccuracies. The granularity of the join result is based on the number of levels accessed. The higher the number of levels accessed, the better the estimations get but this would result in more I/Os. However, assuming that our buffer sizes are 10% of the total number of the two R-trees, the number of nodes accessed for the index nodes fit in the main memory while we provide join estimates with a reasonably tight error.

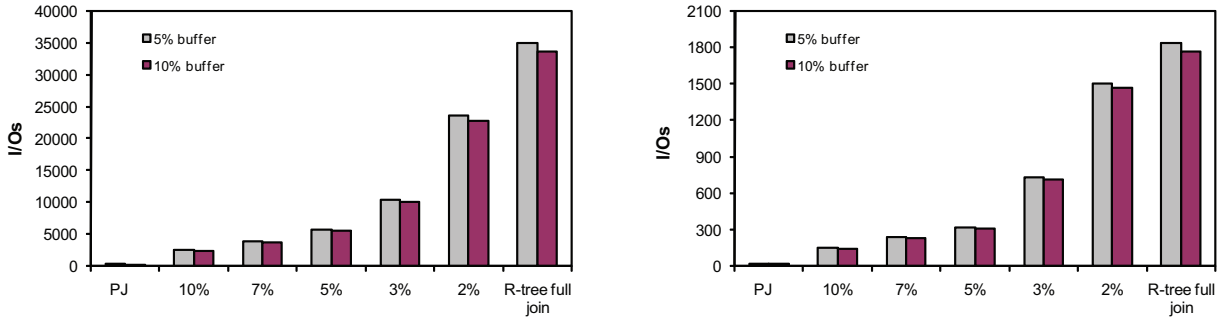
Join level	Level for estimates	uni $\bowtie$ uni	skew $\bowtie$ skew	uni $\bowtie$ skew	real datasets
1^{st} level join	level 1	0.0667	0.1103	0.0770	0.1713
2^{nd} level join	level 1	0.0637	0.0910	0.0561	0.1025
	level 2	0.1168	0.1020	0.1394	0.1858
3^{rd} level join	level 1	0.0347	0.0512	0.0457	0.0619
(leaf level join)	level 2	0.0748	0.0952	0.0735	0.1018
	level 3	0.0027	0.0031	0.0020	0.0034

Table 9: Accuracy of the expected joined area size for vector datasets

We now present the number of I/Os required for PJ compared with the incremental random sampling method proposed in [Bae et al.(2006)] as well as with the full R-tree join. We conducted all possible pairwise joins from our synthetic datasets (u500K, u600K, h500K, h600K) and real datasets. We calculated the number of I/Os to join index nodes, up to the 2^{nd} level joins and 3^{rd} level joins for the synthetic datasets and real datasets, respectively.

In Figure 13 the number of I/Os was plotted for buffer sizes of 5% and 10%. We calculated the average total number of I/Os resulted by each method over all pairwise joins. The results of

PJ with the index nodes are on the left, then the incremental sampling for confidence interval bounds of 10%, 7%, 5%, 3%, and 2%, and finally the results for the full R-tree join on the right most. The PJ algorithm resulted in up to two orders of magnitude less I/Os than the full R-tree join for both synthetic and real datasets. Also, PJ is significantly faster than the sampling algorithm. Although PJ provides no bounded interval for the error, all our experimental results on synthetic and real datasets show the error is reasonably tight, e.g., a 7-9% error for the estimation of 1-upper level joined area and a 3-5% error for that of 2-upper level joined area.



(a) Synthetic datasets

(b) Real datasets

Figure 13: Number of I/Os of PJ, $ISSJ$ and the full R-tree join

Considering PJ with the leaf nodes joins that require 50% of the number of I/Os for the full R-tree, the number of I/Os for PJ is more than the number of I/Os for the incremental sampling with a 3% confidence interval. Yet, this is a lot less than the number of I/Os for the sampling method with a 2% confidence interval. PJ with the leaf node joins provide very low error values, 0.3 % error for the estimates of the current level and 3-6 % error for the estimates of the entire region (see Table 5). Hence the PJ algorithm provides a fast "big picture" of the join result with reasonably accurate estimations.

6 Conclusions

Due to the large dataset size, spatial joins of GIS data may take unreasonably long time to complete. The traditional map overlay joining method does not provide any idea of how the final result will look like until the join is completed. Hence, to enable an interactive data exploration, it is essential to allow a user to get a fast estimation, ideally a "big picture" visualization, of the join result. Users can be more comfortable in using approximations by a method that also provides a confidence interval bound on the estimate.

In this paper, we proposed two statistical methods for estimating spatial joins on both quad-tree indexed raster data and R-tree indexed vector data, namely, Probabilistic Joins (PJ) and Incremental Stratified Sampling Joins ($ISSJ$). We proposed a framework that combines the two statistical methods to allow fast interactive data explorations. Our framework provides tools to report query result estimates while simultaneously providing the user with the ability to modify the input parameters to the database. The user's inputs are allowed for identifying interesting areas to further drill down into, resulting in a faster and more informative data exploration. These two methods are independent, hence GIS applications can implement these methods separately by taking advantages of each method.

Experimental evaluations on real and synthetic datasets in raster format showed that our proposed PJ method resulted in reasonably accurate results with near zero response time. Our $ISSJ$ method, while not as fast as PJ, provided results with bounded confidence intervals up to an order of magnitude faster than full quad-tree join. Experiments on real and synthetic data in vector format showed two orders of magnitude response time improvement relative to the exact answer obtained when using the full R-tree join. Although the PJ method provides no bounded interval for the error, all our experimental results on different synthetic and real datasets showed a reasonably tight error, e.g., a 7-10% error for the estimation of 1-upper level joined area and a 3-6% error for that of 2-upper level joined area. Hence our framework can be used to build an end-user query visualization tool that allows true interactive exploration of large GIS databases.

References

[An et al.(2001)] N. An, Z. Yang, and A. Sivasubramaniam. Selectivity estimation for spatial joins. In *Proceedings of International Conf. on Data Engineering (ICDE)*, pages 368–375, 2001.

[Azevedo et al.(2006)] L. G. Azevedo, R. H. Güting, R. B. Rodrigues, G. Zimbrão, and J. M. de Souza. Filtering with raster signatures. In *Proceedings of ACM GIS*, pages 187–194, 2006.

[Bae et al.(2006)] W. D. Bae, S. Alkobaisi, and S. T. Leutenegger. An incremental refinining spatial join algorithm for estimating query results in GIS. In *Proceedings of International Conf. on Database and Expert Systems Applications (DEXA)*, pages 935–944, 2006.

[Bae et al.(2009)] W. D. Bae, S. Alkobaisi, and S. T. Leutenegger. *IRSJ*: Incremental refining spatial joins for interactive queries in GIS. *GeoInformatica*, Vol. 14(4): 507–543, 2010

[Bae et al.(2010)] W. D. Bae, P. Vojtěchovský, S. Alkobaisi, S. T. Leutenegger, and S. H. Kim. An Interactive Framework for Raster Data Spatial Joins. In *Proceedings of ACM International Symposium on Advances in Geographic Information Systems*, pages 19–26, 2007

[Brinkhoff et al.(1993a)] T. Brinkhoff, H. Kriegel, and B. Seeger. Efficient processing of spatial joins using r-trees. In *Proceedings of ACM SIGMOD*, pages 127–246, 1993a.

[Brinkhoff et al.(1993b)] T. Brinkhoff, H. P. Kriegel, and R. Schneider. Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems. In *Proceedings of International Conf. on Data Engineering (ICDE)*, pages 40–49, 1993b.

[Cheng et al.(2003)] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of ACM SIGMOD*, pages 551–562, 2003.

[Cheng et al.(2006)] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter. Efficient join processing over uncertain data. In *Proceedings of ACM CIKM*, pages 738–747, 2006.

[Guttman(1984)] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD*, pages 45–57, 1984.

[Haas(1997)] P. J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proceedings of International Conf. Scientific and Statistical Databases Management (SSDBM)*, pages 51–63, 1997.

[Haas and Hellerstein(1999)] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. In *Proceedings of ACM SIGMOD*, pages 287–298, 1999.

[Hellerstein et al.(1997)] J. M. Hellerstein, P. J. Hass, and H. J. Wang. Online aggregation. In *Proceedings of ACM SIGMOD*, pages 171–182, 1997.

[Hellerstein et al.(2000)] J. M. Hellerstein, R. Avnur, and V. Raman. Informix under control: Online query processing. *Data Mining and Knowledge Discovery*, Vol. 12:281–314, 2000.

[Larson(1996)] R. R. Larson. *Geographic Information Retrieval and Spatial Browsing.* GIS and Libraries, University of Illinois, 1996.

[Luo et al.(2002)] G. Luo, J.F. Naughton, and C.J. Ellmann. A non-blocking parallel spatial join algorithm. In *Proceedings of Internation conference on Data Engineering*, pages 697–705, 2002.

[Mamoulis and Papadias(1999)] N. Mamoulis and D. Papadias. Integration of spatial join algorithms for processing multiple inputs. In *Proceedings of ACM SIGMOD*, pages 1–12, 1999.

[Medeiros and Pires(1994)] C. B. Medeiros and F. Pires. Databases for GIS. *ACM SIDMOD Record*, Vol. 23(1):107–115, 1994.

[Olken(1993)] F. Olken. *Random Sampling from Databases.* PhD thesis, University of California at Berkeley, 1993.

[Papadias et al.(1999)] D. Papadias, N. Mamoulis, and Y. Theodoridis. Processing and optimization of multiway spatial joins using r-trees. In *Proceedings of ACM PODS*, pages 44–55, 1999.

[Samet(1990)] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, MA, 1990.

[Serfling(2002)] R. J. Serfling. *Basic Statistics for Business and Economics.* McGraw-Hill, 2002.

[Tveite(1997)] H. Tveite. *Data Modeling and Database Requirements for Geographical Data.* PhD thesis, Norwegian University of Science and Technology, Norway, 1997.

[USGS(2001, 2005)] U.S. Geological Survey. USGS: Mineral resources on-line spatial data. URL `http://tin.er.usgs.gov/`.

[Vassilakopoulos and Manolopoulos(1997)] M. Vassilakopoulos and Y. Manolopoulos. On sampling regional data. *Data and Knowledge Engineering*, Vol. 22:309–318, 1997.

[Zimbrão and de Souza(1998)] G. Zimbrão and J. M. de Souza. A raster approximation for the processing of spatial joins. In *Proceedings of VLDB*, pages 558–569, 1998.