Estimating the Overlapping Area of Polygon Join

Leonardo Guerreiro Azevedo[1,*], Geraldo Zimbrão[1,2,**], Jano Moreira de Souza[1,2], and Ralf Hartmut Güting[3]

[1] Computer Science Department, Graduate School of Engineering, Federal University of Rio de Janeiro, PO Box 68511, ZIP code: 21945-970, Rio de Janeiro, Brazil
[2] Computer Science Department, Institute of Mathematics, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[3] LG Datenbanksysteme für neue Anwendungen, FB Informatik, Fernuniversität Hagen, D-58084 Hagen, Germany
{azevedo, zimbrao, jano}@cos.ufrj.br, rhg@fernuni-hagen.de

Abstract. Traditional query processing provides exact answers to queries trying to maximize throughput while minimizing response time. However, in many applications the response time of exact answers is often longer than what is acceptable. Approximate query processing has emerged as an alternative approach to give to the user an answer in a shorter time than the traditional approach. The goal is to provide an estimated result very close to the exact answer, along with a confidence interval, in a short time. There is a large set of techniques for approximate query processing available in different research areas. However most of them are only suitable for traditional data. This work is concerned with approximate query processing in spatial databases. We propose a new algorithm to estimate the overlapping area of polygon join using raster signatures. We executed experimental tests over real world data sets, and the results demonstrated our approach effectiveness.

1 Introduction

A main issue in the database area is to process queries efficiently so that the user does not have to wait a long time to get an answer. However, there are many cases where it is not easy to accomplish this requirement, for example: to process a huge volume of data requires a large number of I/O operations that can demand tens of minutes or hours; to access remote data can be reasonably time-consuming due to a slow network link or even temporary non-availability.

Environments for which providing an exact answer results in undesirable response times motivated the research for techniques in the approximate query processing field. The goal is to provide an estimated response in orders of magnitude less time than the time to compute an exact answer, by avoiding or minimizing the number of disk accesses to the base data [20].

* Leonardo Guerreiro Azevedo is supported by CNPq under grant number 200241/2004-4.
** Geraldo Zimbrão da Silva is supported by CAPES under grant number 3294-04-08.

There are many scenarios and applications where a slow exact answer can be replaced by a fast approximate one, provided that it has the desired accuracy. [13] emphasizes that in Decision Support Systems the intensification in business competitiveness that requires an information-based industry to make more use of its accumulated data, and thus techniques, of presenting useful data to decision makers in a timely manner, to be held as crucial. They also propose the use of approximate query processing during a drill-down query sequence in ad-hoc data mining, where the earlier queries in the sequence are used solely to determine what the interesting queries are. [14] and [21] present the need for performance and scalability when accessing very large volumes of data during the analysis process in data warehousing environments. [29] and [20] propose the use of approximate query processing techniques to define the most efficient access plan for a given query. [1] proposes their use in selectivity estimation in Spatial Database Management Systems (SDMS). An approximate answer can also be used as a tentative answer when the data is unavailable in warehousing environments and in distributed data recording as pointed by [20], [3] and [8] or in mobile computing as highlighted by [25]. [2] points to the use of approximate query processing in order to make decisions and infer interesting patterns online, such as over continuous data streams.

There is a large set of techniques for approximate query processing available in different research areas. However, most of them are only suitable for relational databases. Good surveys of techniques for approximate query processing are presented in [4] and [12]. On the other hand, providing a short time answer to users' queries becomes a bigger challenge in spatial database area, where the data usually have high complexity and is available in huge amounts. Furthermore, this subject is a hot research issue in spatial-temporal databases as pointed by [15]. Moreover, spatial query processing techniques assume that the positional attributes of spatial objects are precisely known. In practice, however, they are known only approximately, with the error depending on the nature of the measurement and the source of data, as pointed by [5] and [16]. So the "exact answer" is actually an approximation, although it is close to the real answer.

[23] defines a spatial database system as a full-fledged database system with additional capabilities for representing, querying, and manipulating geometric data. Such a system provides the underlying database technology needed to support applications such as geographical information systems and others. Spatial data types like point, line, and region provide a fundamental abstraction for modeling the structure of geometric entities, their relationships, properties, and operations.

Efficient evaluation of spatial queries is an important issue in spatial database. Among spatial operations, spatial join operations are very useful but costly to evaluate. Spatial joins have been well studied in the literature, and there are many approaches to process spatial join operations. [9] emphasizes that traditional approaches to performing spatial join processing in two steps ([11] and [24]), and proposes efficient algorithms to be used in the second step. In the two-step approach, the first step employs a Spatial Access Method (SAM) in order to reduce the search space. The Minimum Bounding Rectangle (MBR) is usually used by SAM methods. The second step is a refinement step where the objects resulting from the first step are read from disk and have their geometries processed. On the other hand, [26] proposes a Multi-Step Query Processor (MSQP) including another step between the first and the second step presented previously. In the proposed step the output resulting from the first step is processed against a geometric filter that uses a compact and approximate representation of the object, such as Convex Hull, 5C, RMBR and others found in [27]. The goal is to reduce the number of objects that will have their exact geometry processed in the last step. However, in both approaches (processing the spatial join in two or three steps) it is necessary to process the exact geometries of the objects, the most expensive step that consumes more CPU and I/O resources. To be the best of our knowledge, there is no approach that does not execute the last step, returning to the user an approximate answer along with a confidence interval, processing the join predicate on small approximations of data and not reading the real objects from the disk.

This work is concerned with approximate query processing in spatial databases. We extended the approach presented in [17] in which the use of Four-Colours Raster Signature [6] for approximate spatial query processing was introduced. We propose a new algorithm to compute the approximate intersection area of polygon $\times$ polygon, processing the query on 4CRS raster approximation, along with a confidence interval that is returned to the user allowing him to decide if the accuracy of the response is sufficient. Besides, we also present experimental results in order to show the effectiveness of our approach. One application that could benefit from our approach is the agriculture production estimation. According to the estimated values of agriculture production, several decisions must be taken, for example number and size of warehouses that will store the harvest, number of transports that must be available, roads and railroads that must be (re)constructed, etc. Several spatial joins involving the overlay of thematic planes such as soil, rural areas, rainfall indicators, pollution, areas that are open to pest attacks, etc., must be evaluated to estimate the agriculture production, something that can take a lot of time. On the other hand, a fast approximate answer could be enough for the agriculture production estimation.

The work has been divided in sections, as follows. Section 1 is the introduction. Section 2 presents the most important characteristics of Four-Colours Raster Signature for this work and our proposal of using Four-Colours Raster Signature for estimating the overlapping area of polygon join. Section 3 is dedicated to present the experimental results. Finally, Section 4 shows the conclusions and the future developments of this work.

2 Four-Colours Raster Signature and Estimating the Overlapping Area of Polygon Join

2.1 Four-Colours Raster Signature

The Four-Colours Raster Signature (4CRS) was introduced by [6] to be used as a polygon approximation in spatial join processing. The characteristics of 4CRS and its advantages over other methods motivated its use in approximate query processing area as well. The target of this new approach is to reduce the time required to process a query by avoiding accessing the real datasets which can lead to large amount of time, and processing an approximate query through the execution of a fast algorithm on approximate data, much smaller than the real one. On the other hand, the answer will be estimated and not exact. So, it is also necessary to return a confidence interval in order to have a precision measure of the approximate answer. In general, it is enough for the user to have an approximate answer to make his decision since it has a short execution time and the desired accuracy.

The 4CRS of one polygon is a raster approximation represented by a small four-colour bitmap upon a grid of cells. Each cell of the grid has a colour representing the percentage of the polygon's area within the cell, as shown in Table 1. In Figure 1, an example of 4CRS is presented. The grid can have its scale changed in order to obtain a more accurate representation (higher resolution) or a more compact one (lower resolution). Further details of 4CRS signature can be found in [6] and [17].

Table 1. Types of 4CRS cell

Bit value	Cell type	Description
00	Empty	The cell is not intersected by the polygon
01	Weak	The cell contains an intersection of 50% or less with the polygon
10	Strong	The cell contains an intersection of more than 50% with the polygon and less than 100%
11	Full	The cell is fully occupied by the polygon

Figure 2 presents two examples of grid of cells of the same size. It is easy to notice that it is harder to figure out a simple algorithm that executes on grids like the one presented in Figure 2.a than to figure out a simple algorithm that executes on perfectly overlapped grid, as shown in Figure 2.b. [6] presents an approach for computing the grid of raster approximations where the space is divided into cells independently of the object position through a universal grid so that the coordinate system determines the grid. By doing so, it is assured that if two cells overlap each other then their sides are perfectly superimposed (Figure 2.b). Also, the length of each cell side is always a power of two. So, if two 4CRS signatures have different lengths of cell side and they overlap each other, it is ensured that a small cell is entirely within a great one. This approach was employed in this work, and more details about it can be found in [6].

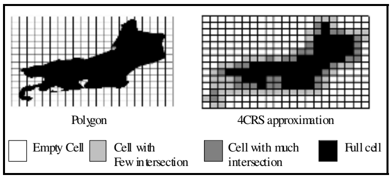

Fig. 1. Example of 4CRS signatures

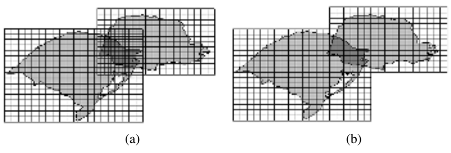

Fig. 2. Grids of cells with same size (a) not overlapping perfectly and (b) overlapping perfectly

When executing query processing on two 4CRS approximations, it is essential that both of them have the same cell size. If that does not apply, it is imperative to perform a change of scale. This is accomplished through the grouping of cells of the approximation with smaller cell size. The algorithm to change the scale evaluates the average of the sum of numerical values assigned to each type of cell, which represents the percentage of the polygon's area within the cell. For *Empty* and *Full* cells the numerical values are 0% and 100%, respectively, since these values represent the exact percentage of intersection area of the cell and the polygon. Due to the fact that in approximate query processing an exact answer is not required, but a close approximate one, in this work we propose to use the average percentage of polygon's area inside the cell as the numerical values for *Weak* and *Strong* cells, which are of 25% and 75%, respectively. These values can be used because the grid and the polygon are independent from each other, and it is expected that the distribution of the percentage of the polygon's area within the cell is very close to the uniform distribution. In fact, we computed the distribution of the polygon area within the cell for the township dataset of Iowa (US) in intervals of 1%, and the result suggests that the uniform distribution assumption holds. Moreover, as shown in [17], the measure used for computing the confidence interval is the variance. Assuming the uniform distribution, the variance of area of weak cells in percentage is $(0.5\text{-}0)^2/12 = 1/48 = 0.020833$, since weak cells have distribution between (0, 0.50]. The strong cell has the same variance. In our test over township dataset of Iowa (US), the computed variances were 0.021978 and 0.021952 for weak and strong cells respectively, whose values are very close to the variance assuming the uniform distribution.

2.2 Expected Area

In this section, the calculus of the expected areas corresponding to the overlapping of two different types of cells with the same size is presented. These expected areas are employed by the algorithm for estimating the overlapping area of polygon join, which is presented in Sub-Section 2.3.

It is easy to notice that the expected area corresponding to a combination of an *Empty* cell with any other type of cell results in an expected area of 0% (zero percent). In the same way when two *Full* cells overlap, the expected area is 100%. Thus, we compute the expected intersection areas for the overlapping of the other type of cells. They were estimated as the mean value of the possible percentage occurrences of the intersection area between two types of cells.

As the datasets are reasonably independent (for example, there is no rule that all township boundaries must be defined by courses of rivers), we can assume that the expected area corresponding to the intersection of two cells with areas x_1 and x_2 is $x_1 \times x_2$. For instance, the expected area corresponding to the overlapping of two *Weak* cells with 10% and 15% of the area of the polygon within them is 1.5% (0.01×0.15). Besides, even though the area is a continuous value, in order to make easy the demonstration of the calculus, we are assuming that the cell area is computed as discrete values, in steps of size of $1/n$ for a large n ($n \rightarrow \infty$). Also, all the values are shown in percentage.

Let *X* be a random variable representing the computed intersection area of one cell of the grid against the polygon; $G(x_1, x_2)$ a function that gives the intersection area between two types of cells x_1 and x_2; and $p(x_1, x_2)$ the join probability function of two variables X_1 and X_2. The definition of mean (or expected value *E*) of two variables is presented in Equation 1 [7].

$$\mathrm{E}[G(X_1, X_2)] = \sum_{x_1}\sum_{x_2} G(x_1, x_2) \times p(x_1, x_2) \; . \tag{1}$$

Since the intersection area between a cell and a polygon is independent of the intersection area of another cell and the polygon, X_1 and X_2 are linearly independent and the joint probability function $p(x_1, x_2)$ can be expressed as $p(x_1, x_2)= p(x_1) \times p(x_2)$. In addition, let *n* be the possible observed values of the percentage of the area of the polygon within the cell. Thus $p(x_1)$ and $p(x_2)$ are equal to $1/n$, since that each value for the intersection area has the same probability of occurrence. Besides, $G(x_1, x_2)$ can be expressed as the multiplication of the intersection areas of the cells within the polygon. Therefore for *n* different kinds of cell intersections $\mathrm{E}[G(x_1,x_2)]$ can be approximately given by Equation 2.

$$\mathrm{E}[G(X_1, X_2)] = \mu = \sum_{i=1}^{n}\sum_{j=1}^{n} \delta(x_1) \times \delta(x_2) \times p(x_1) \times p(x_2) \; . \tag{2}$$

Where $\delta(x)$ is a function that returns the percentages of the area of the polygon within the cell. This function can be expressed as equations 3 and 4.

$$\delta(k) = \frac{k}{2n}, \; 1 \le \mathrm{k} \le \mathrm{n}, \text{ for } \textit{Weak} \text{ cell} \; . \tag{3}$$

$$\delta(k) = \left(\frac{k}{n} + \frac{1}{2}\right), \; 1 \le \mathrm{k} \le \mathrm{n}, \text{ for } \textit{Strong} \text{ cell} \; . \tag{4}$$

In the case of *Weak* and *Strong* cells the percentages vary in the intervals (0, 50%] and (50%, 100%), respectively. While the percentages for *Empty* cell is 0% and for *Full* cells is 100%.

From equations 2, 3 and 4 the expected area of the overlapping of two *Weak* cells employed by the algorithm for computing the approximate intersection area of two polygons can be calculated as follows.

- *Weak* x *Weak* cells

$$\begin{gathered} \mathrm{E}[G(Weak_1, Weak_2)] = \mu = \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_i(weak_1) \times \delta_j(weak_2) \times p(weak_1) \times p(weak_2) \\ \mu = \lim_{n\to\infty} \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{i}{2n}\frac{j}{2n}\frac{1}{n}\frac{1}{n} = \lim_{n\to\infty} \frac{1}{4n^4} \sum_{i=1}^{n} i \sum_{j=1}^{n} j \; . \end{gathered} \tag{5}$$

Since the sum of the sequence $\sum_{k=1}^{n} k$ can be expressed as $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$, and using the L'Hôpital rule, Equation 5 can be rewritten as Equation 6.

$$\mu = \lim_{n\to\infty} \frac{1}{4n^4}\sum_{i=1}^{n} i \sum_{j=1}^{n} j = \lim_{n\to\infty}\frac{1}{4n^4}\frac{n(n+1)}{2}\frac{n(n+1)}{2} = \frac{1}{16}\lim_{n\to\infty}\frac{(n^2+n)(n^2+n)}{n^4} = \frac{1}{16}\ . \tag{6}$$

Following the same reasoning the expected area of the intersection of *Weak* x *Strong* cells, *Strong* × *Strong* cells, *Weak* × *Full* cells, and *Strong* × *Full* cells have the values 3/16, 9/16, 1/4 and 3/4, respectively. Table 2 presents the expected overlapping areas of different types of cells.

Table 2. Expected areas of the overlapping of different types of cells

Cell types	Empty	Weak	Strong	Full
Empty	0	0	0	0
Weak	0	0.0625	0.1875	0.25
Strong	0	0.1875	0.5625	0.75
Full	0	0.25	0.75	1

2.3 Algorithm for Estimating the Overlapping Area of Polygon Join

The algorithm for estimating the overlapping area of polygon join computes the sum of the expected area of their 4CRS signatures' cells that overlap each other, and multiplies the resulting value by the cell's area. Since there are four different types of cells, the superimposing possibilities are sixteen (Table 2), and the algorithm employs a matrix to store the expected areas. It is only necessary to consider the cells that are inside the intersection MBR of the two 4CRS signatures. The algorithm in C-like language is presented in Figure 3, and it handles 4CRS signatures with different or the same length of cell side. It is ensured that when two cells intersect, their sides overlap exactly, and when the lengths of cell sides are different it is always ensured that the smaller cell is whole contained by greater one, according to the approach used to compute the grid of cells presented in Sub-Section 2.1.

```
void approxIntersectionArea(signat4CRS1, signat4CRS2)
  approximateArea = 0;
  interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
  if (signat4CRS1.lengthOfCellSide ==
      signat4CRS2.lengthOfCellSide) then
    s4CRS = signat4CRS1;
    b4CRS = signat4CRS2;
  else
    s4CRS = smallerCellSide(signat4CRS1, signat4CRS2);
    b4CRS = biggerCellSide (signat4CRS1, signat4CRS2);
  approximateArea = 0;
  For each b4CRS cell b that is inside interMBR Do
     For each s4CRS cell s that is inside cell b Do
       approximateArea += expectedArea[s.type,b.type];
  cellArea = s4CRS.lengthOfCellSide *
             s4CRS.lengthOfCellSide;
  return approximateArea * cellArea;
```

Fig. 3. Algorithm for computing the approximate intersection area of polygon × polygon

2.4 Confidence Interval Calculus

When executing a query whose result is an approximate answer, it is important to show to the user a confidence interval of the query's answer, so that the user can decide if the precision of the approximate answer is enough. The precision measure used in this work is based on the Central Limit Theorem [22], which holds almost regardless of the form of the density function. The Central Limit Theorem states that if a population has a mean $\boldsymbol{\mu}$ and a variance $\boldsymbol{\sigma}^2$, then the distribution of sample means derived from this distribution approaches the normal distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2/\boldsymbol{n}$ as the sample size $\boldsymbol{n}$ increases. Thus, at some stage, means for large enough sample sizes, whether the random variable is discrete or continuous, will be approximately normally distributed. Clearly, the form of the parent density function will have some effect on the sample size required, and an asymmetric distribution will generally call for a large $\boldsymbol{n}$ than a symmetric one. However, a sample size of 30 is sufficiently large for many distributions. The confidence interval for approximate processing is computed as the sum of the confidence intervals of each combination of pair of cells. Consulting a statistical table of normal distribution, for a 95% confidence interval we have a range of $(\boldsymbol{\mu}\pm 1.96\times(\boldsymbol{\sigma}^2/\boldsymbol{n})^{1/2})$, and for a 99% confidence interval we have $(\boldsymbol{\mu}\pm 2.576\times(\boldsymbol{\sigma}^2/\boldsymbol{n})^{1/2})$. Equation 7 was used for computing the confidence interval of our experiments.

$$\sum_c n_c \times \left(\mu_c \pm p \times \sqrt{\frac{\sigma_c^{\ 2}}{n_c}} \right) \cdot \tag{7}$$

- $\boldsymbol{\mu_c}$ and $\boldsymbol{\sigma_c}^2$ correspond to the mean and the variance of a combination of cells *c* in the set {*Empty* × *Empty*, *Empty* × *Weak*, …, *Weak* × *Weak*, …, *Full* × *Full*};
- *p* is the value corresponding to the confidence interval chosen, i.e., 1.96 for a 95% confidence interval;
- n_c is the number of cells for the combination *c*.

In order to get the result in area units it is necessary to multiply the result by the cell's area.

For the confidence interval calculus it is necessary to have computed the mean and variance values of the expected areas corresponding to the overlapping of two different types of cells with the same size. Mean values are presented in Table 2 (Sub-Section 2.2) and the calculus of the variance for each combination is presented as follows.

The expected area corresponding to a combination of an *Empty* cell with any other type of cell results in an expected area of 0% (zero percent), because of the intersection area of such kinds of cells is zero. Consequently, the variance of the expected area is zero. In the same way, when two *Full* cells overlap, the expected area is always 100%, and the variance is also zero. Thus, we only need to compute the variances of the expected intersection areas for the overlapping of the other types of cells. We use the same assumptions that were used to calculate the expected areas corresponding to the overlapping of two different types of cells with the same size (Sub-Section 2.2).

Let X be a random variable representing the computed intersection area of one cell of the grid against the polygon; $G(x_1, x_2)$ a function that gives the intersection area between two types of cells x_1 and x_2; and $p(x_1, x_2)$ the join probability function of two variables X_1 and X_2, the variance of the intersection area of two different types of cells can be expressed as Equation 8.

$$\sigma^2 = \sum_{x_1}\sum_{x_2}\left(G(x_1,x_2)-\mu\right)^2 \times p(x_1,x_2)\ . \tag{8}$$

In the same way as presented in Sub-Section 2.2, we assume that X_1 and X_2 are linearly independent and the joint probability function $p(x_1, x_2)$ can be expressed as $p(x_1, x_2)= p(x_1) \times p(x_2)$; $p(x_1)$ and $p(x_2)$ can be expressed as $p(x_1) = p(x_2) = 1/n$; and, $G(x_1, x_2)$ is the multiplication of the intersection areas of the cells within the polygon. By doing so, Equation 8 can be rewritten as Equation 9.

$$\sigma^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\delta(x_1)\times\delta(x_2)-\mu\right)^2 \times p(x_1)\times p(x_2)\ . \tag{9}$$

Where $\delta(x)$ is a function that returns the percentages of the area of the polygon within the cell. This function can be expressed as equations 3 and 4 (Sub-Section 2.2). Thus, from equations 3, 4 and 9 the variance of the percentage of the intersection area between two *Weak* cells can be calculated as follows (Equation 10).

- *Weak* × *Weak*

$$\begin{aligned}
\sigma^2 &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\delta(weak_1)\times\delta(weak_2)-\mu_{weak\times weak}\right)^2 \times p(weak_1)\times p(weak_2)\\
\sigma^2 &= \lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{i}{2n}\times\frac{j}{2n}-\mu_{weak\times weak}\right)^2 \times\frac{1}{n}\times\frac{1}{n}\\
\sigma^2 &= \lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{i^2j^2}{16n^4}-\frac{ij\mu_{weak\times weak}}{2n^2}+\mu_{weak\times weak}{}^2\right)\times\frac{1}{n^2}\\
\sigma^2 &= \lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{i^2j^2}{16n^6}-\lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{ij\mu_{weak\times weak}}{2n^4}+\lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\mu_{weak\times weak}{}^2}{n^2}\ .
\end{aligned} \tag{10}$$

Since the sum of the sequences $\sum_{k=1}^{n}k$ and $\sum_{k=1}^{n}k^2$ can be expressed as $\sum_{k=1}^{n}k=\frac{n(n+1)}{2}$ and $\sum_{k=1}^{n}k^2=\frac{2n^3+3n^2+n}{6}$, and using the L'Hôpital rule, the three limits of Equation 10 can be solved as equations 11, 12 and 13.

$$\lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{i^2j^2}{16n^6}=\frac{1}{16}\lim_{n\to\infty}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}i^2j^2}{n^6}=\frac{1}{16}\lim_{n\to\infty}\frac{\frac{2n^3+3n^2+n}{6}\times\frac{2n^3+3n^2+n}{6}}{n^6}=\frac{1}{144}\ . \tag{11}$$

$$\lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{ij\mu_{weak\times weak}}{2n^4}=\frac{\mu_{weak\times weak}}{2}\lim_{n\to\infty}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}ij}{n^4}=\frac{\mu_{weak\times weak}}{2}\lim_{n\to\infty}\frac{\sum_{i=1}^{n}i\frac{n(n+1)}{2}}{n^4}$$
$$=\frac{\mu_{weak\times weak}}{2}\lim_{n\to\infty}\frac{\frac{n(n+1)}{2}\times\frac{n(n+1)}{2}}{n^4}=\frac{\mu_{weak\times weak}}{8}=\frac{1}{16}\times\frac{1}{8}=\frac{1}{128}\cdot \tag{12}$$

$$\lim_{n\to\infty}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\mu_{weak\times weak}{}^2}{n^2}=\mu_{weak\times weak}{}^2\lim_{n\to\infty}\frac{n\times n}{n^2}=\mu_{weak\times weak}{}^2=\left(\frac{1}{16}\right)^2=\frac{1}{256} \tag{13}$$

Applying equations 11, 12 and 13 in Equation 10, the variance of the percentage of the intersection area between two *Weak* cells are presented in Equation 14.

$$\sigma^2=\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\delta(weak_1)\times\delta(weak_2)-\mu_{weak\times weak}\right)^2\times p(weak_1)\times p(weak_2)$$
$$\sigma^2=\frac{1}{144}-\frac{1}{128}+\frac{1}{256}=0.003038194\cdot \tag{14}$$

The variances of the expected areas of the intersection of other types of cells can be calculated following the same reasoning. They are presented in Table 3, and we do not present their calculus due to space limitations.

Table 3. Variance of the expected areas of the overlapping of different types of cells

Cell types	Empty	Weak	Strong	Full
Empty	0	0	0	0
Weak	0	0.003038194	0.013454861	0.020833333
Strong	0	0.013454861	0.023871528	0.020833333
Full	0	0.020833333	0.020833333	0

Therefore it is possible to return to the user a confidence interval for the approximate query processing. For instance, let a query to produce the following pair of cells 100 *Weak* × *Weak* cells, 40 *Weak* × *Strong* cells, 70 *Weak* × *Full* cells, 60 *Strong* × *Strong* cells and 200 *Full* × *Full* cells we compute the 95% confidence interval as presented in Figure 4 (for simplicity we assume that each cell has the same area, equals to 1).

- W×W:100 × (0.0625 ± 1.96 × $(0.0030382/100)^{1/2}$) = 6.25 ± 1.0803
- W×S: 40 × (0.1875 ± 1.96 × $(0.013454/40)^{1/2}$) = 7.50 ± 1.4378
- W×F: 70 × (0.2500 ± 1.96 × $(0.020833/70)^{1/2}$) = 17.50 ± 2.3669
- S×S: 60 × (0.5625 ± 1.96 × $(0.023872/60)^{1/2}$)= 33.75 ± 2.3457
- F×F: 200 × 1 = 200 (full cells have the exact area!)
- Total: 265 ± 7.2308.

Fig. 4. Example of 95% confidence interval calculus

So, the confidence interval has a range of ±7.2308 that is 95% of the approximate answers with these numbers of cell combinations will have an error of at most ±2.7286%, a result with enough precision for most applications. For a 99% confidence interval, it is necessary to replace 1.96 to 2.576 in the calculus presented in Figure 4. In this case, the computed value is 265 ± 9.5034. The confidence interval has a range of ±9.5034, which means an error of at most ±3.5862% in 99% of the cases.

3 Experimental Results

This section is dedicated to present the experimental results found by using 4CRS signature for estimating the overlapping area of polygon join. In order to evaluate the effectiveness of our approach we compared the approximate processing against the exact processing according to the following metrics: response time (the time to provide an approximate answer for a query); accuracy (the precision of the answers, along with a confidence interval); and footprint (the storage requirements for the approximations).

3.1 Test Environment, Experimental Data Sets, 4CRS Signatures and R*-Tree Characteristics

Tests were executed on a PC Pentium IV 1.8 GHz with 512 MB of RAM. A page size of 2,048 bytes for I/O operations was defined. The polygon real data sets used in the experiments consist of township boundaries, census block-group, geologic map and hydrographic map from Iowa (US), available online at "http://www.igsb.uiowa.edu/rgis/gishome.htm", and Brazilian municipalities [10]. In order to simulate large datasets, the Iowa datasets were replicated six times, in the same way as suggested by [26]. The original polygons were shifted by random displacements of *x* and *y* coordinates. In the case of Brazilian municipalities, we performed only one replication (named Brazilian municipalities'), so that we could execute the test of Brazilian municipalities against Brazilian municipalities'. Some data characteristics are presented in Table 4.

Table 4. Test data sets characteristics

	Datasets	size (KB)	# pol.	# seg.	Avg # seg.
Iowa	Census block-group	38,824	17,844	1,764,588	98
	Topography	60,748	40,140	7,561,104	188
	Hydrologic map	6,904	2,670	475,812	178
	Township boundaries	25,288	12,216	1,059,438	86
	Geological map	21,856	9,984	640,428	64
Brazil	Municipalities	9,840	4,645	399,002	85
	Municipalities'	9,840	4,645	399,002	85
	Average	24,757	13,163	1,757,053	112

In order to generate the 4CRS signatures, we have to choose the maximum number of cells of the grid [6]. Intuitively, the larger the number of cells, the closer is the approximation to the original polygon. However, processing 4CRS signatures that have large sizes could produce high I/O and CPU costs. To evaluate the effects of different choices, we executed experimental tests with maximum number of cells of 250, 500, 1000, 1500 and 2000. We evaluated the approximate processing against the exact processing computing the intersection area of dataset 1 × dataset 2 presented in Table 7. Signatures with maximum number of cells equal to 250 have smaller storage requirements, but the precision of the approximate answers is not good enough. On the other hand, the answers are better estimated when the maximum number of cells was 2000; however the I/O and CPU costs are higher as well, because of the higher signature sizes. Figure 5 summarizes these experimental results showing: storage requirements (percentage of 4CRS signatures' sizes related to the datasets' sizes); error of the approximate answer (the percentage corresponding to the difference between the approximate value and the exact value related to exact value); percentage of the time required to execute the approximate processing related to the exact processing; and the percentage corresponding to the number of disk accesses needed to execute the approximate processing related to the exact one. We present in details in Sub-Section 3.2 the experimental results when 500 was used as the maximum number of the grid cells, which produced approximate answers with acceptable average error and confidence interval.

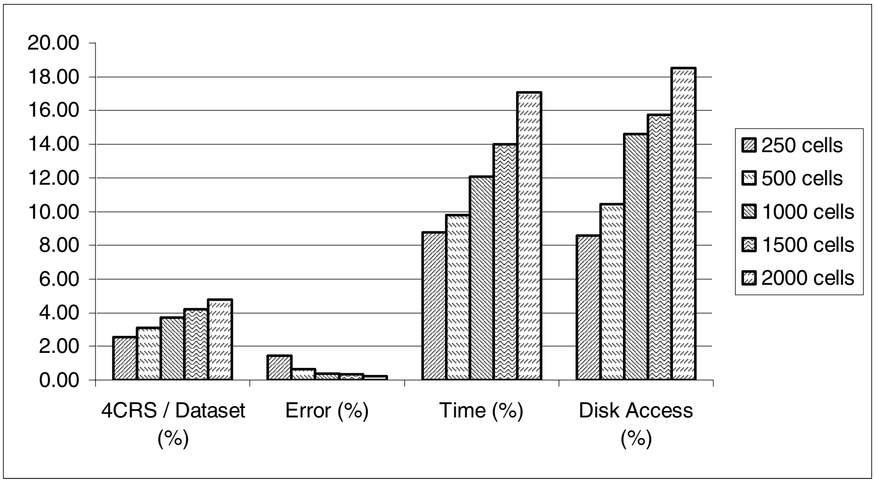

Fig. 5. Storage requirements, accuracy and number of disk access for maximum number of cells of the grid equal to 250, 500, 1000, 1500 and 2000

The 4CRS signature generation time was not shown because [6] evaluated its efficiency and presented good results. Table 5 presents the 4CRS signatures characteristics for the maximum number of cells equals to 500. We can notice that, in order to store 4CRS signatures of maximum number of cells equal to 500 it is only

Table 5. 4CRS signatures' characteristics with maximum number of cells equal to 500

Datasets		Dataset size (KB)	4CRS size (KB)	4CRS / Dataset size (%)
Iowa	Census block-group	38,824	1,603	4.13
	Hydrologic map	6,904	177	2.56
	Township boundaries	25,288	838	3.31
	Geological map	21,856	676	3.09
Brazil	Municipalities	9,840	329	3.34
	Municipalities'	9,840	329	3.34
Average		18,759	659	3.30

needed, on average, 3.30% of the space needed to store the real datasets. In other words, it is necessary approximately 30 times more space to store the real datasets than to store the 4CRS signatures.

To perform the join, the R*-tree [19] was chosen as a spatial access method in order to reduce the search space. In other words, the R*-Tree was used to take account only the objects that have at least MBR intersection and not all of them. That choice was due to the wide use of R*-Tree, as well as, to the successful results found in the literature. The access methods traditionally used employ the object's Minimum Bounding Rectangle (MBR), and the access methods execution returns what is called a set of candidates, since it contains all the pairs of polygons that belong to the answer plus other pairs that have only MBR intersection. In the same way as [26] and [6] do, for our tests we generated R*-Trees that store the 4CRS signatures as part of the polygons' keys, and this means that they were stored in the leaf nodes in the R*-Tree index. It is a reasonable approach since in this way we have to compute the 4CRS just once.

Our tests can be described according to the concepts of Multi-Step Query Processor (MSQP) proposed by [26], presented in Section 1. In the approximate query processing, only the first two steps of the MSQP (SAM + Filter steps) were executed. Since it is not necessary to access the real objects when computing an approximate answer, the last step of MSQP was not executed. On the other hand, in the exact query processing, we executed the first and last step of MSQP (SAM + Refinement steps). In other words, after finding the objects that have MBR intersection, the exact representation of the objects was processed, and exact answers returned. To perform a fair test we generated R*-Trees without storing the 4CRS signatures on their leaf nodes to be used in the exact query processing. By doing so, the sizes of the R*-Trees without storing signatures are smaller than the sizes of the R*-Trees that store them, consequently the number of disk accesses in the first step is smaller as well. The R*-Trees characteristics are presented in Table 6. The column "R*-Tree type" shows that the characteristics presented are of R*-Tree that stores 4CRS signatures or R*-tree that do not store signatures.

In order to evaluate the 4CRS effectiveness in the approximate processing area, besides the storage requirements, we evaluated the approximate processing against the exact processing testing the accuracy of the approximate answer, execution time and disk accesses. The approximate query processing was done executing the algorithm proposed in the Sub-Section 2.3, while the exact query processing was performed using the General Polygon Clipping library that is available on the web at http://www.cs.man.ac.uk/aig/staff/alan/ software/#gpc.

Table 6. R*-Trees' characteristics

	Datasets	**R*-Tree type**	**R*-Tree size (KB)**	**Time (sec)**	**Leaf node average use (%)**	**Height**	**# leafs**
Iowa	Census block-group	4CRS	2,124	19.04	69.98	3	1045
		-	1,160	17.93	69.81	3	570
	Hydrologic map	4CRS	334	2.24	68.33	3	162
		-	162	2.14	75.35	2	79
	Township boundaries	4CRS	1,546	12.95	68.70	3	760
		-	800	11.97	69.50	3	392
	Geological map	4CRS	1,258	9.55	68.41	3	617
		-	644	9.32	70.46	3	316
Brazil	Municipalities	4CRS	586	4.66	71.15	3	286
		-	284	4.07	75.05	3	138
	Municipalities'	4CRS	582	4.92	71.63	3	284
		-	284	4.11	75.05	3	138
	Average	4CRS	1,289	8.89	69.70	3	525
		-	663	8.26	72.54	3	272

3.2 Results of Approximate Query Processing

This sub-section is dedicated to presenting, in detail, the experimental results when the maximum number of cells of the grid was 500. The results correspond to: precision of the approximate answer, including confidence intervals; processing time; and number of disk accesses. Storage requirements of 4CRS signatures were presented in Sub-Section 3.1 (Table 5). We executed queries computing the intersection area of dataset 1 against dataset 2 (presented in Table 7) comparing the approximate processing against the exact one. Each query was executed 20 times, and for each time we generated a random window so that only the considered pairs of objects were inside the window. In order to evaluate the effect of the number of objects returned by each query, we executed two different tests. In one test the random windows were generated with size of 4% of the size of the whole space of the datasets, and in the other test the windows were generated with size of 12.25%. The results are presented in Table 8 and Table 9. Since the values of both tests are quite similar, we will only analyze in more details the results corresponding to the second test (Table 9). The most relevant difference between the tests is that in the second test each window is intercepted by more objects than the number of intersections in the first test. As a result, the number of cells considered to compute the confidence interval is bigger and its value is closer to the computed approximate answer.

Experimental results show the effectiveness of the use of 4CRS signature in the approximate processing area due to the quite small error of the approximate answers, the short time of the approximate processing and the small number of disk accesses. The average error of the approximate answers is 0.59%, while the confidence intervals of 95% and 99% have average values of 0.97% and 1.28%, respectively (Table 9, column "Error and confidence interval"). In other words, the approximate answers have on average a difference of only 0.59% related to the exact ones. Besides, in order to show the accuracy of the approximate answers, a confidence interval is also returned to the user which means that for a precision of 95% the error is at most ±0.97%, while for a precision of 99% the error is at most ±1.28%.

The approximate query processing is on average approximately 9 times faster than the exact query processing, since it needs only approximately 11% of the time of the exact processing to execute the approximate one. Table 9 (columns "Processing Time") presents the processing time in seconds and the percentages corresponding to the approximate query processing related to exact one.

Table 7. Tests

Labels	Dataset 1	Dataset 2
Query-1	Brazilian municipalities	Brazilian municipalities'
Query-2	Township boundaries	Census block
Query-3	Township boundaries	Geological map
Query-4	Township boundaries	Hydrologic map
Query-5	Census Block	Hydrologic map
Query-6	Hydrologic map	Geological map

Table 8. Experimental results corresponding to the 20 executions of the intersection area of dataset 1 × dataset 2 × random window with size of 4% of the size of the whole space

	Error and Confidence Interval			Processing Time			Number of Disk Accesses			Average of objects per window
Queries	**Error (%)**	**C. I. 95%**	**C. I. 99%**	**Approx. Proc.**	**Exact Proc.**	**%**	**Approx. Proc.**	**Exact Proc.**	**%**	
Query-1	0.779	2.973	3.907	6.279	73.025	8.60	2138	26826	7.97	1813
Query-2	0.304	0.534	0.702	14.621	93.204	15.69	8691	50979	17.05	2590
Query-3	0.831	1.386	1.822	12.478	109.337	11.41	5440	44289	12.28	2551
Query-4	0.255	1.231	1.617	8.212	75.098	10.94	5747	27064	21.23	1591
Query-5	0.438	1.292	1.699	17.375	110.85	15.67	4470	33862	13.20	1935
Query-6	0.847	1.243	1.634	20.66	95.217	21.70	2736	22419	12.20	1267
Average	0.58	1.44	1.90	13.27	92.79	14.00	4870	34240	13.99	1958

The total execution time is not a good measure of performance gain as it is totally dependent on the algorithm used to execute the exact processing. Besides, the cache of the Operating System can influence processing time. Instead, the total number of disk accesses is a reliable performance gain measure, as the objects to be processed have to be, at least, read from disk. Table 9 (columns "Number of Disk Accesses") presents the gains of the approximate processing related to the exact one. The former needs in average only 16% of the number of disk access of the latter. In other words, the exact processing requires in average 6 times more disk accesses than the approximate processing.

Table 9. Experimental results corresponding to 20 executions of the intersection area of dataset 1 × dataset 2 × random window with size of 12.25% of the size of the whole space

	Error and Confidence Interval			**Processing Time**			**Number of Disk Accesses**			**Average of objects per window**
Queries	**Error (%)**	**C. I. 95%**	**C. I. 99%**	**Approx. Proc.**	**Exact Proc.**	**%**	**Approx. Proc.**	**Exact Proc.**	**%**	
Query-1	1.05	2.48	3.26	18.00	272	6.62	6166	62353	9.89	4907
Query-2	0.30	0.31	0.40	40.64	398	10.22	24512	133509	18.36	7394
Query-3	0.79	0.82	1.08	33.68	392	8.60	15737	108508	14.50	6940
Query-4	0.18	0.75	0.99	21.84	250	8.74	13378	65645	20.38	4305
Query-5	0.45	0.74	0.97	46.14	390	11.84	13055	87123	14.98	5581
Query-6	0.74	0.75	0.98	86.01	405	21.22	9504	59656	15.93	3886
Average	0.59	0.97	1.28	41.05	351	11.21	13725	86132	15.67	5502

4 Conclusions

This work proposes, implements, and evaluates a new approach for estimating the overlapping area of polygon join queries. The target is to provide an estimated result in orders of magnitude less time than the time to compute an exact answer, along with a confidence interval for the answer. We propose to compute the intersection area of pairs of polygons over 4CRS signatures of the polygons, processing compact and approximate representations of the objects, and avoiding accessing the whole data. By doing so, the exact geometries of the objects are not processed during the join execution, which is the most costly part of the spatial join since it requires the search and transfer of large objects from the disk to the main storage ([26] and [18]). Also, the exact processing algorithm needs to use complex CPU-time intensive algorithms for deciding whether the objects match the query condition [27]. There are many scenarios and applications where a slow exact answer can be replaced by a fast approximate one, provided that it has the desired accuracy, as presented in Section 1.

We evaluate our approach comparing the approximate processing against the exact processing according to storage requirements, accuracy, response time, and number of disk accesses. The results achieved were quite good, and demonstrated the effectiveness of our approach. The 4CRS signature has low storage requirements; the approximate answers have a quite small error; and, the processing time and the number of disk accesses required to execute the approximate processing are much smaller than the time and number of disk accesses of the exact processing. In Sub-Section 3.2, we presented details of the experimental results for small size signatures. These tests showed that an average of 30 times less space to store 4CRS signatures is needed than to store the real datasets. The approximate answers have an average error of 0.6%, while the confidence intervals of 95% and 99% have average values of 0.97% and 1.28% respectively, which is enough precision for most applications. Besides, the approximate processing varies from 5 to 15 times faster than the exact processing in response time and from 5 to 10 times relate to number of disk accesses.

As future work we plan to evaluate the use of more colours in the raster approximations, for example, eight colours. We believe that it can provide a better precision, and confidence intervals closer to the approximate answer. Besides, although this will have the extra cost of storing more bits for colour representation, storage requirements can be kept small since we apply compression methods on the 4CRS signatures. We also plan to investigate an algorithm to compute the number of cells that leads to a 4CRS signature that better represents the polygon, based on the complexity of the polygon [28]. A straightforward approach is to compute the 4CRS signature starting with the maximum number of cells equal to one, and then increase this number until the proportion between approximate area (*strong* and *weak* cells) and the exact area reaches a pre-defined threshold. Besides, the algorithm proposed in this work can be extended or new algorithms can be developed in order to process other kinds of operations.

References

1. Das, A., Gehrke, J., Riedwald, M.: Approximation Techniques for Spatial Data. In Proc. of ACM-SIGMOD Conference, Paris, France (2004) 695-706.
2. Dobra, A., Garofalakis, M., Gehrke, J. E., Rastogi, R.: Processing complex aggregate queries over data streams. In Proc. of SIGMOD (2002) 61-72.
3. Faloutsos, C., Jagadish, H. V., Sidiropoulos, N. D.: Recovering information from summary data. In Proc. of 23rd Int. Conf. on Very Large Data Bases, Athens, Greece (1997) 36-45.
4. Barbara, D., DuMouchel, W., Faloutsos, C., Hass, P., Hellerstein, J. M., Ioannidis, Y., Jagadish, H., Johnson, T., Ng, R., Poosala, V., Ross, K., Sevcik, K.: The New Jersey data reduction report. Bulletin of the Technical Committee on Data Engineering, IEEE Data Engineering Bulletin (1997) 20(4):3-45.
5. Heuvelink, G.: Error Propagation in Environmental Modeling with GIS, Taylor & Francis, London, UK (1998).
6. Zimbrao, G., Souza, J. M.: A Raster Approximation for Processing of Spatial Joins. In Proc. of the 24th VLDB Conference, New York City, New York (1998) 558-569.
7. Larson, H. J.: Introduction to probability theory and statistical inference. John Wiley & Sons (1982).
8. Jagadish, H. V., Mumick, I. S., Silberschatz, A.: View maintenance issues in the chronicle data model. In Proc. ACM PODS, San Jose, CA (1995) 113-124.
9. Zhu, H., Su, J., Ibarra, O. H.: Toward Spatial Joins for Polygons. In Proc. Int. Conf. of SSDBM, Berlin, Germany (2000) 2431-246.
10. IBGE (Brazilian Institute of Geography and Statistics): Malha Municipal Digital do Brasil, Rio de Janeiro, Brasil (1994).

11. Orenstein, J. A.: Spatial query processing in an object-oriented database system. In Proc. of ACM SIGMOD Int. Conf. on Management of Data, Washington, DC (1986) 326-336.
12. Han, J., Kamber, M.: Data Mining: concepts and techniques. Academic Press (2001).
13. Hellerstein, J. M., Haas, P. J., Wang, H. J.: Online aggregation. In Proc. of ACM SIGMOD Int. Conf on Management of Data, Tucson, Arizona (1997) 171-182.
14. Costa, J. P., Furtado, P.: Time-Stratified Sampling for Approximate Answers to Aggregate Queries. In Proc. of Int. Conf. on Database Systems for Advanced Applications, Kyoto, Japan (2003) 215-222.
15. Roddick, J., Egenhofer, M., Hoel, E., Papadias, D., Salzberg, B.: Spatial, Temporal and Spatiotemporal Databases Hot Issues and Directions for PhD Research. SIGMOD Record, (2004) 33(2):126-131.
16. Zhang, J., Goodchild, M.: Uncertainty in Geographical Information System. Taylor & Francis, Erewhon, NC (2002).
17. Azevedo, L. G., Monteiro, R. S., Zimbrao, G., Souza, J. M.: Approximate Spatial Query Processing Using Raster Signatures. In Proc. of VI Brazilian Symposium on GeoInformatics, Campos do Jordao, Brazil (2004).
18. Lo, M. L., Ravishankar, C. V.: Spatial Hash-Joins. In Proc. of the ACM-SIGMOD Conference, Montreal, Canada (1996) 247-258.
19. Beckmann, N., Kriegel, H. P., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In Proc. of ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ (1990) 322-331.
20. Gibbons, P. B., Matias, Y., Poosala, V.: Aqua project white paper. Technical report, Bell Laboratories, Murray Hill, NJ (1997).
21. Furtado, P., Costa, J. P.: Time-Interval Sampling for Improved Estimations in Data Warehouses. In Proc. of 4th Int. Conf. on Data Warehousing and Knowledge Discovery, Aix-en-Provence, France (2002) 327-338.
22. Steel, R. G. D., Torrie, J. H.: Introduction to statistics. McGraw-Hill Book Company (1976).
23. Güting, R. H., de Ridder, T., Schneider, M.: Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types. In Proc. of the 4th Int. Symposium on Large Spatial Databases, Portland, Maine (1995) 216-239.
24. Kothuri, R. K., Ravada, S.: Efficient Processing of Large Spatial Queries Using Interior Approximations. Proc. of the Int. Symposium on Spatial and Temporal Databases, Los Angeles, CA (2001) 404-424.
25. Madria, S. K., Mohania, M. K., Roddick, J. F.: A Query Processing Model for Mobile Computing using Concept Hierarchies and Summary Databases. Proc. of Int. Conference of Foundations of Data Organization, Kobe, Japan (1998) 147-157.
26. Brinkhoff, T., Kriegel, H. P., Schneider, R., Seeger, B.: Multi-step Processing of Spatial Joins. In Proc. of ACM-SIGMOD Int. Conference on Management of Data, Minneapolis, MN (1994) 197-208.
27. Brinkhoff, T., Kriegel, H. P., Schneider, R.: Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems. In Procs. of Int. Conf. on Data Engineering, Vienna, Austria, (1993) 40-49.
28. Brinkhoff, T., Kriegel, H. P., Schneider, R., Braun, A.: Measuring the Complexity of Polygonal Objects. In Proc. of ACM Int. Workshop on Advances in Geographic Information Systems, Baltimore, MD, (1995) 109-118.
29. Ioannidis, Y. E., Poosala, V.: Balancing histogram optimality and practicality for query result size estimation. ACM SIGMOD, (1995) 233-244.