An Incremental Refining Spatial Join Algorithm for Estimating Query Results in GIS

Wan D. Bae, Shayma Alkobaisi, Scott T. Leutenegger

Department of Computer Science, University of Denver
2360 South Gaylord Street, Denver, CO 80206, U.S.A.
{wbae, salkobai, leut}@cs.du.edu

Abstract. Geographic information systems (GIS) must support large georeferenced data sets. Due to the size of these data sets finding exact answers to spatial queries can be very time consuming. We present an incremental refining spatial join algorithm that can be used to report query result estimates while simultaneously provide incrementally refined confidence intervals for these estimates. Our approach allows for more interactive data exploration. While similar work has been done in relational databases, to the best of our knowledge this is the first work using this approach in GIS. We investigate different sampling methodologies and evaluate them through extensive experimental performance comparisons. Experiments on real and synthetic data show an order of magnitude response time improvement relative to the exact answer obtained when using the R-tree join.

1 Introduction

Geographic Information Systems (GIS) are used in many fields and applications for exploring large data sets in order to obtain intuition and insight into the stored information. GIS queries often compute exact numerical query answers. However, computing the exact answers to queries in large databases can be time consuming. Often an approximate answer is sufficient and can prevent wasted computation time, thus allowing a more interactive exploration of the data. Our goal is to speed up the exploratory process of GIS data while providing a statistical confidence of preliminary results through an incremental refining process. For example, instead of waiting a long time for an exact answer, we compute an approximate answer, say bounded by 5% of the exact answer with 95% probability (confidence level), in $\frac{1}{20}$ of the time needed for an exact answer. To make the system more interactive, the user is given the ability to stop queries whenever the answer is "good enough". This approach allows the user to quickly obtain an idea of whether or not the query is useful and can therefore modify it or stop it accordingly. While GIS and spatial databases offer many sophisticated algorithms for computing exact query answers for spatial data, many of these techniques are time prohibitive for the exploration of large spatial data sets. Thus, many GIS applications can benefit from our approximate and fast approach.

In this paper we present the *Incremental Refining Spatial Join (IRSJ)* algorithm. The *IRSJ* algorithm provides an order of magnitude improvement in response time relative to a full R-tree join. The algorithm achieves its reduced execution time by providing query estimates obtained from a subset of the full join directed by random sampling. We experimentally compare two versions of the *IRSJ* algorithm against each other as well as against the time needed to obtain an exact answer using a full R-tree join algorithm.

2 Background

2.1 Overview of GIS and Motivation

GIS data are used to describe the geometry and location of various types of geographic phenomena [12]. Geographic or spatial queries are performed on a spatially indexed database in order to obtain answers that depend on spatial relationships between data items [13]. Examples of spatial relationships include intersection, containment, and adjacency. While our approach can be used for any of these, in this paper we focus on finding an estimate of the number of intersections. An example where our method would be useful is "How many mineral plants intersect radiometric aged areas in the US?", where this query returns the number of areas that can be of interest to geologists to estimate the earth's age. Such a query would likely take a long time to find the exact number of intersections. Our approach allows the time to be dramatically reduced if the user is willing to accept an estimated value within a bounded confidence interval.

2.2 Related Work

Our work follows the lead of online query processing work done at the University of California-Berkeley [1, 3, 4]. The authors proposed an interface for online relational aggregation to provide the user with the ability to control the query execution process. The goal was to minimize time by obtaining an approximate query answer instead of computing the exact answer. They proposed the "ripple join" as part of this paradigm. We follow their general framework but devise a new approximate sampling based join algorithm to work for spatial data.

Spatial relations are usually indexed using R-trees [8]. 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 query, 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 data sets. Many spatial join algorithms based on R-trees have been proposed with perhaps two of the most common being found in [9, 10]. Our algorithm follows a similar approach to that found in [10]. A data item from one data set is joined with the other data set by executing a window query. For example, if we join "cities" and "rivers", then the MBR of a city provides the query MBR to be executed against the river data set. Our work differs from the past work in that we only execute intersection queries for a subset of the "outer" data set through sampling. We incrementally refine the query answer until we get the desired confidence interval accuracy.

Another approach for obtaining join selectivity estimates is to use histogram-based methods [11]. Such methods offer the promise of even faster query estimates than our method, but still have disadvantages. Although the experimental results in [11] show low error, the method is not bounded whereas our method provides error confidence intervals and the ability to incrementally tighten the intervals. Also the histogram methods provide an estimate of the number of joins, but they do not provide actual join results. Our method can be used to produce a subset of the actual join results rather than just an estimate of the number of tuples that will join.

3 The Incremental Refining Spatial Join in GIS

Our incremental refining spatial join algorithm consists of three steps: sampling, spatial joining, and refining the estimation of the query result.

3.1 Random Data Sampling

Sampling chooses a subset of the data to obtain query estimates. Chosen samples should accurately represent the entire data set, and a confidence interval is used to reflect the accuracy of the estimated value [14, 15]. We consider two common database sampling methods: tuple-level and page-level [5]. In tuple-level sampling, a number of tuples are chosen as samples, each with the same probability. Tuple-level sampling obtains random samples regardless of data clustering, however, its performance is poor if no index is available. If an index is available, then performance is improved since the sampling predicate is applied inside the index of leaf pages. In page-level sampling, pages are chosen as samples instead of tuples. If a page is chosen at random, all tuples in that page are processed to calculate the number of intersections for that page. Page-level sampling has better performance than tuple-level in terms of I/Os [5]. However, aggregate estimate accuracy can be worse when using page-level sampling due to the correlation of data within a page. The query estimations and the confidence intervals are statistically meaningful only if samples are retrieved in random order. [6] presented techniques for random sampling from various indices to produce meaningful confidence intervals.

In our experiments we used one of the weighted random sampling methods, *Acceptance/Rejection* in [6, 7], in which the inclusion probability is proportional to some parameter of the item sampled. We investigated tuple-level and page-level sampling with varying data sets and buffer sizes. We do sampling incrementally without replacement.

Algorithm 1 $IRSJ_t(R, S, C_f, n)$

```
 1: $C \leftarrow 0$; $C_I \leftarrow 0$ { count, confidence interval}
 2: repeat
 3:   for $i = 0$ to $k$ do
 4:     $L \leftarrow$ Choose leaf from $R$ at random
 5:     $M \leftarrow$ MBR of a randomly chosen tuple within $L$
 6:     $I \leftarrow$ number of intersections of a Window Query($M, S$)
 7:     $C \leftarrow C + I$
 8:   end for
 9:   $C_I \leftarrow$ Compute confidence interval using $C$
10:   $EV \leftarrow$ Compute estimated value using $C$
11: until The desired confidence interval $C_f$ is attained
```

3.2 The Incremental Refining Spatial Join Algorithm (*IRSJ*)

We have developed and compared two $IRSJ$ versions: $IRSJ_t$ and $IRSJ_p$ for tuple-based and page-based sampling, respectively. Assume we have two data sets, denoted R and S, that we wish to join. Let R be the outer data set and S be the inner data set. We assume R and S are both indexed by R-trees.

In $IRSJ_t$ we select a page of R at random and choose one tuple within this page at random. We use the MBR of this tuple as the MBR of an intersection query to query data set S using its R-tree. The number of intersections is reported and used in calculating a running estimated value and a confidence interval. Algorithm 1 describes $IRSJ_t$, where C_f is the desired confidence interval and k (updating rate) is the number of tuples in each sampling step. A tradeoff exists between the rate at which the confidence intervals are updated and the time to which the interval length decreases at each update. In our experiments we used $k = 30$. Since pages may contain a different number of tuples, especially if R is indexed by an R-tree, it is necessary to choose pages with a probability relative to the number of tuples within the page.

In $IRSJ_p$ we sample a page of R at random for each update. The difference between $IRSJ_t$ and $IRSJ_p$ is that in $IRSJ_p$ we compute an intersection query for each tuple within a sampled page, then use the obtained average as a single sample. Due to likely correlation of tuples within a page, it is necessary to treat the average of the page as a single value when calculating the confidence interval.

4 Confidence Interval Calculation

To provide bounds on the accuracy of our incremental result, we concurrently calculate and return to the user the current estimated value and confidence interval. We give an overview of the statistical method used in $IRSJ$ and present the confidence interval of a population proportion based on the Central Limit Theorem (CLT) [2, 14, 15]. CLT states that the sampling distribution of the sample mean approximates a normal distribution for a specified number of samples from any population. The approximation improves with more samples. For the confidence interval we use the binomial probability distribution. The outcome of each trial (join) is either “intersect” or “does not intersect”. The binomial distribution is determined by the number of trials n and the probability p of success in a single trial. The probability of a success remains the same from one trial to the next. We assume that the normal curve is a good approximation to the binomial distribution. Empirical studies have shown that these methods are quite good when both $np > 5$ and $nq > 5$, where $q = 1 - p$ [15].

We describe the way to obtain the confidence intervals of a population proportion. Let r be the number of successes out of n trials in a binomial experiment. We take the sample proportion of successes $\hat{p} = r/n$ as our point estimate for p, the population proportion of successes, and point estimate for q is $\hat{q} = 1-\hat{p}$. The difference between the actual value of p and the estimate $\hat{p}$ is called the error estimate for using $\hat{p}$ as a point estimate for p. For large samples the distribution of $\hat{p}$ is well approximated by a normal curve with mean $\mu = p$ and standard error $\sigma = \sqrt{\frac{pq}{n}}$. Since the distribution of $\hat{p}$ is approximately normal, we use features of the standard normal distribution to find the difference $\hat{p} - p$. An interval that estimates a population parameter within a range of possible values at a specified probability is called confidence level c. Let z_c be the number such that an area equal to c under the standard normal curve falls between $-z_c$ and z_c. Then we have $P(-z_c\sqrt{\frac{pq}{n}} < \hat{p} - p < z_c\sqrt{\frac{pq}{n}}) = c$.

We call E the maximal error tolerance of the error of estimate $|\hat{p} - p|$ for a confidence level. To find the confidence interval for p, we have $P(\hat{p}-E < p < \hat{p}+E) = c$. The difficulty is that we may not know the actual values of p or q in most situations, so we use our point estimates $p \approx \hat{p}$ and $q = 1-p \approx 1-\hat{p}$ to estimate E. These estimates are safe for most practical purposes since we are dealing with large-sample theory. Then the confidence interval for p is $\hat{p} - E < p < \hat{p} + E$, where $\hat{p} = \frac{r}{n}$: we have $E = z_c\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} = z_c\sqrt{\frac{\hat{p}\hat{q}}{n}}$, where z_c= critical value for confidence level c taken from a normal distribution.

In $IRSJ_t$, $n = n_t$, the number of tuples processed and in $IRSJ_p$, $n = n_p$, the number of pages processed. Note that $n_p \leq n_t$. Different sampling methods generalize the standard CLT of confidence interval formulas [14]. We use the following formula for the confidence interval of $IRSJ_t$:

$$E = z_c\sqrt{\frac{\hat{p}(1-\hat{p})}{n-1} \cdot \frac{N-n}{N}}$$

5 Experiments

In this section we present experimental results of the $IRSJ_t$ and $IRSJ_p$ algorithms on both synthetic and real GIS data sets. We compare the algorithms to each other as well as to obtain an exact answer using a full R-tree join algorithm [9]. We show the estimated values and the corresponding confidence intervals returned to the user with a 95% confidence level. We present the number of I/Os with varied buffer sizes as well as the number of node (page) accesses. All code is implemented in Java.

5.1 Datasets and Experimental Methodology

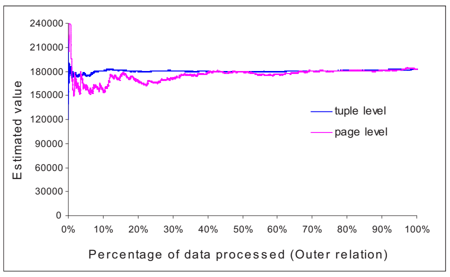
We consider both synthetic and real data sets in our studies. Our synthetic data sets consist of (i) uniform (random) and (ii) skewed (hyper-exponential) distributions. For the uniform data set, (x,y) locations are distributed uniformly and independently between 0 and 1. The (x,y) locations for the skewed data sets are independently drawn from a hyper-exponential distribution with mean 0.3 and variance 0.25. Let U and S denote uniform and skewed distributed data sets, respectively. We considered all join combinations: $S \bowtie S$, $U \bowtie S$, $S \bowtie U$, $U \bowtie U$. We vary the number of tuples of each data set between 100,000 and 600,000 tuples. We only present the most illustrative subset of our results due to space limitations. Results not shown also resulted in up to an order of magnitude improvement in performance. Our real data sets are from the U.S. Geological Survey in 2001 and 2005 [16]: (i) Mineral Resources in the US (outer relation R): U.S. Geological Survey, 2005. (ii) Geochemistry of unconsolidated sediments in the US (inner relation S): U.S. Geological Survey, 2001. The two data sets have 300,434 and 199,850 tuples (MBRs), respectively.

Our experiments are conducted using the following parameters: R-tree page size of 4Kbytes with fan-out size of 100 for leaf and non-leaf nodes and minimum capacity of 40. Since R-trees are disk-based index structures, the natural performance metric is the number of page I/Os required for a given buffer size. We assume an LRU buffer and vary buffer size between 300 and 1800 pages resulting in the buffer holding between 3% and 60% of the inner R-tree.

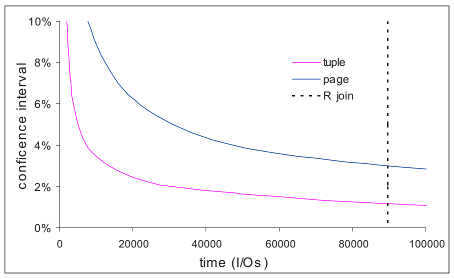
5.2 Synthetic Data Results

We first present results for the synthetic data sets. In Figure 1 (a) and (c) we plot the running estimated query result versus the numbers of tuples processed. The smoother line is for $IRSJ_t$ while the jagged line is for $IRSJ_p$. In Figure 1 (a) the outer data set is uniform while the inner data set is skewed, and in Figure 1 (c) the outer data set is skewed while the inner data set is uniform. In Figure 1 (b) and (d) we plot the confidence interval versus time measured as the number of buffer misses. Figure (b) corresponds to (a) and figure (d) to (c). The time to process a page of tuples in $IRSJ_p$ is greater than the time to process a tuple in $IRSJ_t$, thus, the most fair comparison is time which is directly proportional to the number of buffer misses. The top curve is for $IRSJ_p$ and the lower curve is for $IRSJ_t$. The vertical line on the right is the execution time needed for a full R-tree join algorithm. The estimate accuracy is significantly worse under $IRSJ_p$ than $IRSJ_t$. Put another way, the $IRSJ_t$ algorithm takes less time to obtain the same accuracy as $IRSJ_p$. Note that the time to obtain an exact answer is more than an order of magnitude greater than that needed to get an accuracy within 5% using $IRSJ_t$.

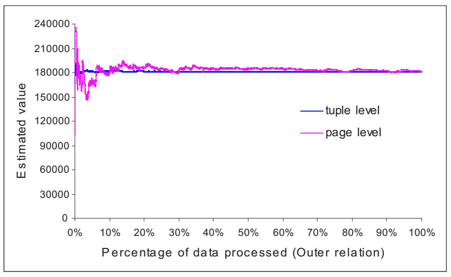
In Figure 2 we show experimental results when we vary the size of the data sets and also vary buffer size to keep the buffer size fixed at 10% of the inner R-tree size. As can be seen, the full R-tree join requires 4-16 times more disk access than $IRSJ_t$ with a 5% half confidence interval. As the data set size increases



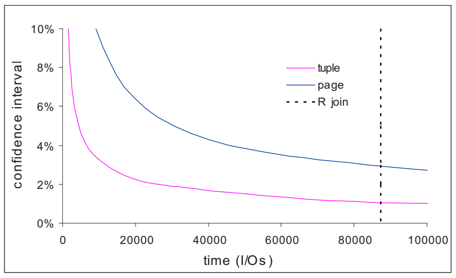
(a) Estimated Value: U-600K $\bowtie$ S-400K



(b) Confidence Interval: U-600K $\bowtie$ S-400K



(c) Estimated Value: S-600K $\bowtie$ U-400K



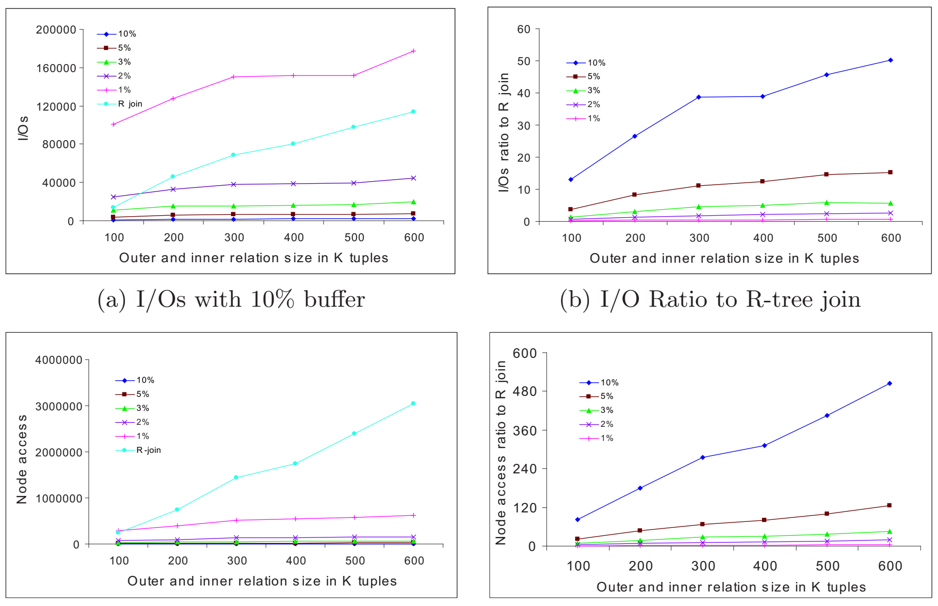
(d) Confidence Interval: S-600K $\bowtie$ U-400K

Fig. 1. Estimated Value and Confidence Interval for synthetic datasets 600K $\bowtie$ 400K

the advantage of $IRSJ_t$ over the full R-tree join increases. Thus, for larger data sets, as expected in a real GIS, the benefit of our proposed approach will be even greater. Figure 3 shows the results for an experiment where the size of data sets is varied while buffer size is fixed at 1200 pages. Experiment results show that $IRSJ_t$ provides an I/O performance improvement of more than an order of magnitude relative to the R-tree join.

5.3 GIS Real Data Result

We now present results for the U.S.G.S. data sets. Our query is to join mineral resources with geochemical unconsolidated sediments in the US. The query returns the number of intersections of mineral resources and geochemical unconsolidated sediments. In Figure 4 we plot the accuracy and confidence intervals for $IRSJ_t$ and $IRSJ_p$. Again, the smoother line is for $IRSJ_t$ while the jagged line is for $IRSJ_p$. As can be seen, $IRSJ_t$ takes less I/Os to get the same accuracy as $IRSJ_p$. In Table 1 on page 10 we present the number of I/Os and the number of node accesses with buffer size 5% and 10% of the inner R-tree. For a buffer size of 10%, the R-tree join requires 74 times more I/Os than the $IRSJ_t$ with a 5% half confidence interval. In Figure 5 we present results of our experiments when we vary buffer size from 300 to 1800 pages. The experiment results again show that $IRSJ_t$ provides a good estimation in very early query processing stages.



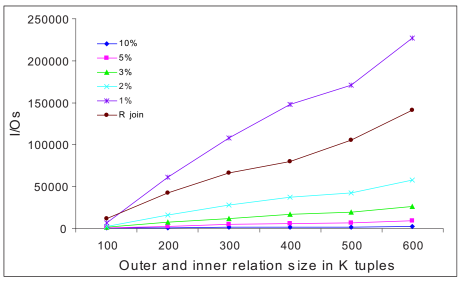
(a) I/Os with 10% buffer (b) I/O Ratio to R-tree join

(c) Node Accesses with 10% buffer (d) Node Access Ratio to R-tree join

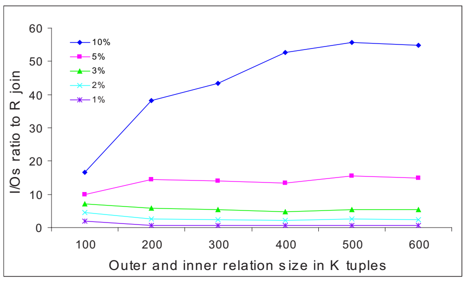
Fig. 2. I/Os and Node Accesses of $IRSJ_t$ for varying size of synthetic datasets with 10% buffer size and Ratio to a full R-tree join: R(uniform) $\bowtie$ S(skewed)

6 Conclusions

In this paper we proposed the Incremental Refining Spatial Join ($IRSJ$) algorithm to efficiently estimate the results to spatial queries. We implemented two versions of $IRSJ$: one based on tuple-level sampling ($IRSJ_t$) and the other based on page-level sampling ($IRSJ_p$). Our experiments show that the time (I/Os) required to obtain a reasonably accurate estimate was order of magnitude smaller than the time needed for an exact answer obtained using a full R-tree join algorithm using both real and synthetic data sets. We also observed that as the size of data sets increases, the improvement of $IRSJ_t$ over the full R-tree join also increases. Thus, the benefit of our approach will be even greater for larger GIS data sets. Perhaps surprisingly, our tuple-level sampling algorithm performed better than our page-level sampling algorithm. This is a result of only being able to use one datum in confidence interval calculation due to data set clustering which violates the needed independence. In our future work we plan to investigate different sampling methods to improve the performance of $IRSJ$ and develop a mathematical model to determine the optimal sampling strategy. Further, we plan to explore the utility of $IRSJ$ for answering multiway joins.

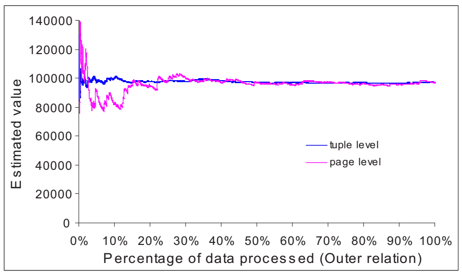


(a) I/Os

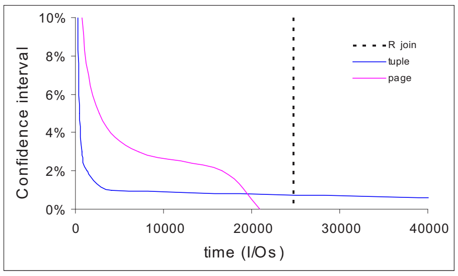


(b) I/O Ratio to R-tree join

Fig. 3. I/Os and Ratio of $IRSJ_t$ to a full R-tree join for varying size of synthetic datasets with a fixed buffer size 1200



(a) Estimated Value



(b) Confidence Interval

Fig. 4. Estimated Value and Confidence Interval for the real datasets

Acknowledgments

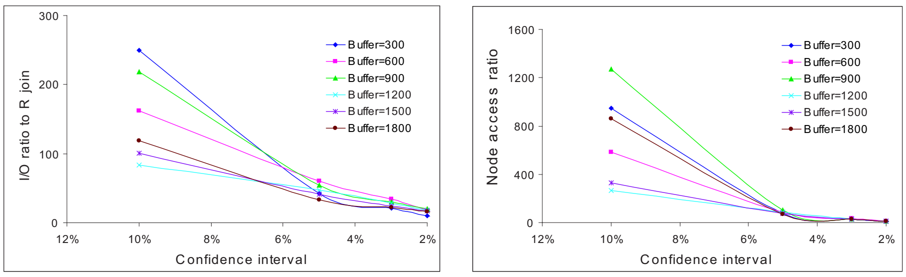
The following people provided helpful suggestions during early stages of this work: Mario Lopez, Sada Narayanappa, Brandon Haenlein, and Mohammed Al-bow. Help in statistics was graciously provided by Prof. Ray Boersema and Prof. Tom Obremski.

References

1. Hellerstein, J. M., Hass, P. J., Wang, H. J.: Online aggregation. In Proc. ACM SIGMOD. (1997) 171–182
2. Hass, P. J.: Large-sample and deterministic confidence intervals for online aggregation. In Proc. SSDM. (1997) 51–63
3. Hass, P. J., Hellerstein, J. M.: Ripple Joins for Online Aggregation. In Proc. ACM SIGMOD. (1999) 287–298
4. Hellerstein, J. M., Avnur, R., Raman, V.: Informix under CONTROL: Online Query Processing. Data Mining and Knowledge Discovery. Vol.12 (2000) 281–314

	Buffer Size	H.C.I.=10	H.C.I.=5	H.C.I.=3	H.C.I.=2	H.C.I.=1	R-join
I/Os	5%	233	403	770	1896	8159	27756
	10%	182	333	752	1164	3795	24756
Node Accesses	5%	780	2591	6933	18299	87233	262200
	10%	668	2136	9028	19671	85822	262200

Table 1. I/Os and Node Accesses of $IRSJ_t$ and a full R-tree join for the real datasets: H.C.I.=Half Confidence Interval, Buffer size=Percent of relation size



(a) I/O ratio to R-tree join (b) Node Access ratio to R-tree join

Fig. 5. I/Os and Node Access Ratio to a full R-tree join for the real datasets

5. Seshadri, S.: Probabilistic methods in Query Processing. Ph.D. Dissertation. University of Wisconsin. (1992)
6. Olken, F.: Random Sampling from Databases. Ph.D. Dissertation. University of California, Berkeley. (1993)
7. Rubinstein, R. Y.: Simulation and the Monte Carlo Method, John Wiley and Sons, Inc. (1981)
8. Guttman, A.: R-trees: a Dynamic Index Structure for Spatial Searching. In Proc. ACM SIGMOD. (1984) 45–57
9. Brinkhoff, T., Kriegel, H., Seeger, B.: Efficient Processing of Spatial Joins Using R-trees. In Proc. ACM SIGMOD. (1993) 237–246
10. Papadias, D., Mamoulis, N., Theodoridis, Y.: Processing and Optimization of Multiway Spatial Joins Using R-trees. In Proc. ACM PODS. (1999) 44–55
11. An, N., Yang, Z., Sivasubramaniam, A.: Selectivity estimation for spatial joins. In Proc. ICDE. (2001) 368–375
12. Medeiros, C. B., Pires, F.: Databases for GIS. ACM SIDMOD Record. Vol.23, No.1 (1994) 107–115
13. Larson, R. R.: Geographic Information Retrieval and Spatial Browsing. GIS and Libraries. University of Illinois. (1996) 81–127
14. Scheaffer, R. L., Mendenhall, W., Ott, R.L.: Elementary Survey Sampling. 5th edn. New York. Duxbury Press. (1995)
15. Serfling, R. J.: Basic Statistics for Business and Economics. McGraw-Hill. 4th edn. New York. (2002)
16. USGS Mineral Resources On-Line Spatial Data : http://tin.er.usgs.gov/. (2001, 2005)