A Cost Model for Estimating the Performance of Spatial Joins Using R-trees *

Yun-Wu Huang†
IBM T. J. Watson Research Center
ywh@watson.ibm.com

Ning Jing‡
Changsha Institute of Technology
ningjing@pdns.nudt.edu.cn

Elke A. Rundensteiner§
Worcester Polytechnic Institute
rundenst@cs.wpi.edu

Abstract

The development of a cost model for predicting the performance of spatial joins has been identified in the literature as an important and difficult problem. In this paper, we present the first cost model that can predict the performance of spatial joins using R-trees. Based on two existing R-trees (join targets), our model first estimates the number of expected I/Os for the join process by assuming a zero buffer size. Our method for this estimation extends the cost model for R-tree window queries (developed by Kamel and Faloutsos and by Pagel et al.) to also handle spatial joins (which are more complex). In the context of spatial join processing, this number of zero-buffer expected I/Os is not practical for performance prediction in a buffered environment. To model the buffer impact, we use an (exponential) distribution function to measure the probability that a bufferless I/O would cause a page fault in a buffered environment. Based on this probability and the zero-buffer expected I/O cost, the estimated number of I/Os for an R-tree join can then be computed. The comparisons between the predictions from our cost model and the actual results from our experiments based on real GIS maps show that the average relative error ratio is about 10% with a maximum of about 20% for a wide range of buffer sizes. Therefore, our model is a useful tool for the query optimization of spatial join queries.

* This work was supported in part by the University of Michigan ITS Research Center of Excellence grant (DTFH61-93-X-00017-Sub) sponsored by the U.S. Dept. of Transportation and by the Michigan Dept. of Transportation. N. Jing was supported in part by the State Education Commission of P.R. China.

† This work was performed while the author was at the University of Michigan.

‡ This work was performed while the author was a visitor at the University of Michigan.

§ This work was performed while the author was a faculty member of the University of Michigan.

1 Introduction

Recently, the ability to manage spatial data has become more and more crucial in a wide range of applications such as Geographic Information Systems (GIS), Cartography, image processing, VLSI, and CAD/CAM. For this reason, spatial databases have experienced an ever-increasing market demand. One of the essential functions required by many spatial database applications are spatial joins which combine objects from two data sets based on a spatial predicate such as *intersect* or *contain* [3, 10]. For example, a query such as "**Q1**: *Find all counties that intersect flooded areas with water-level* $>$ *one foot*" is a spatial join query combining counties with areas based on an *intersect* spatial predicate.

Current strategies for spatial join processing typically execute in two steps:

- The filter step uses a simple data structure, such as the MBR (Minimum Bounding Rectangle), to approximate each object. It then computes a set of object-pairs, one from each target data set, such that the MBRs of the two objects satisfy the spatial predicate. This set of object-pairs is called a *candidate set*.
- The refinement step takes the *candidate set* produced by the filter step as input, and retrieves the full spatial data representation of the two objects in each object-pair of the candidate set. It then tests to see if they indeed satisfy the spatial predicate by using an accurate spatial function applied to the two spatial polygon objects instead of only their MBRs.

Because processing the refinement step is very expensive in terms of overall processing time [20], a possible query optimization strategy is to push it as late as possible until all other constraints are resolved. Take query **Q1** for example, one could first select all flooded areas satisfying the constraint "*water-level* $>$ one *foot*", and then process the filter step to create a *candidate set* between the counties and the selected subset of flooded areas. Alternatively, the query processor may process the filter step between all counties and all flooded areas first, then conduct selection to reduce the size of the *candidate set*. In either way, the refinement process is best pushed to the very end.

In order for a query processor to properly choose among these two or other possible processing strategies, we must be able to predict their costs with acceptable confidence. A cost model that effectively predicts the cost of spatial joins is not only important for simple spatial join processing, but also valuable for more complex classes of queries such as spatial path queries [10]. Spatial path queries could be optimized by pre-processing spatial joins before conducting path search, or vice versa, by first conducting path search on the full map and then performing spatial join only for the subset of links on the selected path. Due to the lack of analytical models, in [10], we address this problem of evaluating alternative processing strategies for spatial path queries by performing experimental studies. The development of cost models for spatial joins, which is the topic of this paper, will thus be a valuable step towards being able to effectively select among possible query processing plans for such complex queries.

It is stated in [3] that developing an analytical cost model for proposed spatial join processing using R-trees is difficult, if not impossible. This difficulty is partially based on the following two factors: one, an R-tree join involves interaction between two R-trees, and two, any tree node may be visited more than once, thus modeling of buffer management must be properly incorporated into the analytic cost model. Therefore, while there has been a plethora of recent research efforts that focused on optimizing the spatial join process (in particular the filter step) [3, 9, 11, 15, 20], no one has succeeded to date to develop an analytical cost model. In other words, current research in spatial joins does not provide the query manager a way to predict the processing cost of the spatial join.

In this paper, we fill this gap by presenting the first cost model that can analytically measure the performance of the filter step of a spatial join process. In particular, we focus on spatial joins based on existing R-tree indexes for both input data sets [3, 9]. We refer to this type of join as R-tree join in this paper. R-tree joins are very important because R-trees (and their variants) have become the most popular spatial access methods used by many spatial DBMS products [12, 24, 27]. Furthermore, the performance of R-tree joins is one of the best among all spatial join strategies [20].

Our cost model first estimates the number of expected I/Os incurred during the filter step without considering the buffer effect (we assume a zero buffer size). We call it ZEIO for Zero-buffer Expected I/Os. Our estimation of ZEIO for an R-tree join is based on the spatial density concept of the R-tree cost model presented in [13, 19] for window queries. However, we extend the model beyond window queries to also capture the I/O cost for the more complex spatial join processing. Unlike for window query processing where each tree page is accessed at most once, each R-tree page is typically accessed multiple times during an R-tree join. The estimated ZEIO, based on our zero-buffer assumption, therefore cannot be directly used to measure the R-tree join performance with a non-zero buffer. Given that in practice buffer sizes play an important role in the performance of a spatial join query, our cost model must account for the sizes of the buffer.

For a zero-buffer, ZEIO is the expected number of page accesses where each access has a probability of 1 in causing a page fault. For a non-zero buffer, the probability of each access causing a page fault is generally less than 1. We propose to simulate the R-tree join paging behavior under a non-zero buffer using the exponential distribution function to capture the locality of page accesses during an R-tree join (for further discussion of this see Section 3.2.2). Given a known buffer size, we then use this distribution to derive the probability of a page fault for each page access. Lastly, based on the derived probability and the ZEIO computed from our cost model, we can measure the expected number of I/Os (EIO) which takes buffer sizes into account.

In this paper, we also present actual R-tree join experiments conducted based on real GIS maps (with roughly evenly distributed objects) extracted from [4, 23]. The comparison between the prediction from our analytical model and the results from the experiments show that the average error ratio is about 10%, with a maximum error of about 20%. Our cost model therefore provides the query manager with a powerful tool for predicting the R-tree join performance, thereby contributing to more efficient processing of spatial join queries.

The outline of the remainder of this paper is as follows: In Section 2, we describe the background and related work on spatial join processing and R-tree modeling. Our proposed R-tree join cost model is introduced in Section 3, followed by the performance evaluation in Section 4. Section 5 concludes this paper.

2 Background

2.1 R-tree Structure

R-trees [8] are an extension of B-trees [1] that stores multi-dimensional data. Like B-trees, R-trees are balanced and dynamically adjustable. Unlike B-trees, a non-leaf node in an R-tree contains entries of the form $< addr, mbr >$ where $addr$ is the address of a child node and mbr is the Minimum Bounding Rectangle (MBR) that encloses MBRs of all entries in that child node. A leaf node contains entries of the form $< oid, mbr >$ where oid refers to a spatial object stored in the database and mbr is the MBR of that spatial object.

In most R-tree variants, MBRs stored in the tree nodes are allowed to overlap one another [2, 8, 13]. This means that, unlike B-trees, there may be more than one search path to identify a desired object. To improve this weakness, recently proposed R-tree variants try to minimize the overlap between the MBRs [2, 13] in their tree nodes. Among them, R*-tree [2] introduces more sophisticated heuristics than the original R-tree [8] which have been shown to improve the query performance. In [13], R-trees are constructed in a bottom-up approach called the packed R-tree. By packing bottom-up, 100% node occupancy is achieved. A Hilbert-curve [13] transformation is also deployed to pre-sort the MBRs in order to achieve minimal MBR overlap. In this paper, we present experimental studies based on packed R-trees.

2.2 Spatial Joins

While there are many recent research efforts reported in the literature that focus on spatial join processing, we review only a subset of these efforts below due to space limitations. In [18], the z-ordering technique is used to transform multi-dimensional data into the 1-dimensional domain. Spatial join is then conducted on the B$^+$-tree structures that store z-ordering values of the spatial data. In [21], spatial join indexes are computed over spatial data indexed by Grid files [17]. In [16], spatial join is conducted to create the distance-associated indexes for spatial range queries. In [7], a model of the generalization tree is proposed to compare the tree-based spatial joins with the alternative approaches using cost estimation.

Spatial join based on depth-first traversal of R-trees was proposed in [3]. Their technique exploits the R-tree hierarchy by synchronously traversing both R-trees in depth-first order. Subtrees from both R-trees are only traversed if the MBRs of the root nodes of the subtrees overlap. A variety of CPU and I/O optimizations are also presented in [3]. To this date, this R-tree join [3] has become the state-of-the-art approach for spatial joins when R-tree indexes exist for both spatial data sets. Its performance has also become the yardstick used by other researchers to measure the performance of their proposed non-index based spatial join methods [14, 15, 20]. In [9], we introduce an alternative R-tree join technique based on breadth-first traversal that we show to be an improvement over the depth-first approach.

2.3 Cost Models for R-trees

A cost model for measuring R-tree performance was first presented by Faloutsos et al [5]. Later, Kamel and Faloutsos [13] and Pagel et al. [19] presented an R-tree cost formula that can estimate the average number of disk accesses for a fixed-sized query window. Their work [13, 19] assumes uniform distribution of data, an assumption which we have also adopted for this work. Because our proposed cost formula for the estimated number of potential disk accesses for an R-tree join is based on the concepts presented in [13, 19], we shall describe this model in more detail in Section 2.4.

In [6], Faloutsos and Kamel presented a new R-tree cost formula for point data sets using a data property called fractal dimension that no longer requires the assumption that the data is uniformly distributed. This model is not applicable to our spatial join analysis because our input data corresponds to regions rather than points. In [26], Theodoridis and Sellis presented a model that can predict the R-tree performance based on certain properties of data sets, meaning that the performance of R-trees is estimated even before they are built.

To the best of our knowledge, no analytical cost model for spatial joins has been presented in the literature.

2.4 An R-tree Window Query Cost Model

We now describe the R-tree cost model proposed by [13, 19] which is used as the basis for the R-tree join cost model presented in this paper. As required by the model, we assume the data sets contain two-dimensional data, although extending the model for multi-dimensional data is straightforward. Formal proofs are omitted below, but most can be found in the original papers [13, 19].

For the remainder of the discussion, we assume that the entire data space is a window of 1×1.

Lemma 1 *Let w be a window of x $\times$ y over the entire data space* 1×1. *Then the probability of a point being in w is x $\times$ y.*

Lemma 2 *Let* w_1 *be a window of* $x_1 \times y_1$ *and* w_2 *be a window of* $x_2 \times y_2$ *over a region of 1 $\times$ 1. Then the probability of* w_1 *overlapping* w_2 *is* $(x_1 + x_2) \times (y_1 + y_2)$.

The above is based on the observation that if we represent w_1 with a point i (e.g., lower left corner) in w_1, then the probability of w_1 overlapping w_2 is equal to the probability of i being in a window of $(x_1 + x_2) \times (y_1 + y_2)$.

Theorem 1 *Given an R-tree, let h be the height,* N^l *be the number of nodes at level l, and* n_i^l *be the i'th node of the R-tree at level l. Let* $x_i^l \times y_i^l$ *be the MBR of* n_i^l. *For a window query of size x $\times$ y, the expected number of disk accesses is given by:*

$$DA = \sum_{l=1}^{h} \sum_{i=1}^{N^l} ((x_i^l + x) \times (y_i^l + y)). \qquad (1)$$

Theorem 1 is the application of Lemma 2 to every node in the R-tree. Based on Equation 1, the performance of a window query with a size of $x \times y$ on a given R-tree can then be estimated. Our cost formula in estimating the performance of an R-tree join is an extension of this equation.

3 A Cost Model for R-tree Joins

We now propose an R-tree join cost model as a two-step process. First, we measure the join performance without considering the buffer (i.e., the buffer size is assumed to be *zero*). The result of this process will generate the total number of zero-buffer expected I/Os (ZEIO). This corresponds to a model of processing where each access of a tree page, redundant or not, causes a page fault. Next, our model incorporates the buffer size variable by modeling the R-tree join paging patterns using a probability distribution function.

3.1 Measuring the Zero-Buffer Expected I/Os (ZEIO)

We now present the cost formula that estimates ZEIO for an R-tree join. This cost formula (Equation 2) is an enhancement of the window query estimation in Equation 1 with the purpose of modeling the cost of R-tree joins. To explain the derivation of Equation 2 from Equation 1, we note that an R-tree join can be viewed as a series of window queries on one R-tree such that while searching over the MBRs at level l from one R-tree, the target windows are the MBRs at the same level from the other R-tree. We first assume that the two input R-trees for the joins are of the same height h and will relax this restriction later.

Theorem 2 *Let R and $\tilde{R}$ be two input R-trees for an R-tree join with their tree components as described in Theorem 1. The number of zero-buffer expected I/Os of an R-tree join between R and $\tilde{R}$ is:*

$$ZEIO_h = 2 + 2 \times \sum_{l=1}^{h-1} \sum_{i=1}^{N^l} \sum_{j=1}^{\tilde{N}^l} (x_i^l + \tilde{x}_j^l) \times (y_i^l + \tilde{y}_j^l). \quad (2)$$

The basic idea of performing R-tree joins is to synchronously test overlaps between MBRs while traversing down the two trees [3]. Beginning from the root nodes of both R-trees, if their MBRs overlap then their children nodes are tested pair-wise for overlapping. This process continues downwards iteratively until either the tested MBRs do not overlap or the parent of leaf nodes is reached. In Equation 2, the first term (the constant 2) is the cost of retrieving the root nodes from the two trees. The constant in front of the outer loop (the factor of 2) sums the I/O costs for both trees. The loop with variable l represents the tree levels. While the loop with variable i iterates over nodes at the same level, its inner loop (with variable j) computes the expected number of children nodes one needs to traverse for a given tree node.

Note that we do not need to retrieve information from the leaf nodes for conducting such a computation (the upper bound of the loop over levels in Equation 2 is $h-1$). This is because the filter step stops at the leaf level and the MBRs of the leaf nodes are stored at their parent nodes. Therefore, the computation of $ZEIO$ requires less I/O costs than the actual computation of an R-tree join by a factor of o, where o is the average number of MBRs stored in each R-tree node. Typically, $o > 100$, therefore the computation of $ZEIO$ can be efficiently accomplished by traversing the upper levels of the two R-trees.

We now extend the above result to the case when the two trees have different heights.

Theorem 3 *Let R and $\tilde{R}$ be two input R-trees for an R-tree join with their respective tree components described in Theorem 1. Let h and $\tilde{h}$ ($h < \tilde{h}$) be the heights of R and $\tilde{R}$, respectively. The number of zero-buffer expected I/Os (ZEIO) of an R-tree join between R and $\tilde{R}$ is:*

$$ZEIO_{\tilde{h}} = ZEIO_h + 2 \times \sum_{l=h}^{\tilde{h}-1} \sum_{i=1}^{N^h} \sum_{j=1}^{\tilde{N}^l} (x_i^{h-1} + \tilde{x}_j^l) \times (y_i^{h-1} + \tilde{y}_j^l) \quad (3)$$

When the trees are of different height, the synchronized downward traversal process continues until the parent-of-leaf level is reached for the shorter tree (the cost is $ZEIO_h$ in Equation 2). From then on, the traversal stays at the same level for the shorter tree while it continues downward for the taller one. The join process terminates when both trees reach their parent-of-leaf levels.

ZEIO is the expected number of I/O (EIO) only when the buffer size is zero. However, multiple accesses of the same tree pages are likely to occur during R-tree joins. Depending on the buffer size, many of the potential disk accesses may find the pages in the system buffer without causing a disk access. Therefore, ZEIO = EIO *iff* $b = 0$, where b is the size of the system buffer. Typically $b > 0$, and we have ZEIO $\geq$ EIO, i.e., ZEIO is the I/O upper bound of an R-tree join. Therefore ZEIO is only a worst-case estimated cost of R-tree joins. We now propose to use the derived ZEIO and other information such as b as inputs to conduct a probabilistic analysis in better approximating EIO.

3.2 Approximating the Expected Number of I/Os (EIO)

To approximate EIO, we incorporate the buffer impact by using the following probabilistic analysis.

3.2.1 A Probabilistic Analysis of Paging Pattern

Assume that during an R-tree join process, a tree node N is accessed s times and the size of the system (LRU) buffer is b. The only times when accessing N causes a page fault are when it is first read, and when there are b or more page faults occurring between two consecutive accesses of N. Let the number of page faults between two consecutive accesses of N be a random variable x (we call it the Inter-access Page Fault, or IPF). While the first access of N causes a page fault with a probability of 1, the probability of each of the subsequent accesses of N to cause a page fault is $P\{x \geq b\}$. The expected number of page faults $E(N)$ caused by an access of a tree node N therefore is defined as given next.

Definition 1 *Given a tree node N, then the expected number of page faults, denoted by $E(N)$, caused by an access of N is defined by:*

$$E(N) = \begin{cases} 1 & \textit{if first access of } N \\ P\{x \geq b\} & \textit{otherwise.} \end{cases}$$

3.2.2 Modeling the Distribution of the Inter-access Page Faults (IPF)

We assume that the paging behavior of an R-tree join exhibits a high locality such that the number of page faults occurring during two consecutive accesses of a page (IPF) is likely to be small. This assumption is reasonable because R-tree joins typically deploy some optimization techniques [3, 9] (such as *plane sweep*, sorting, pinning, etc.) to insure a good paging performance. Based on this, we treat IPF as a random variable that has a exponential distribution, which is widely used to model the inter-arrival time in queueing theory [25].

3.2.3 Applying the Exponential Distribution

The distribution of an exponential random variable x is:

$$f(x) = \lambda e^{-\lambda x}, \tag{4}$$

where $x \geq 0$ ($x < 0$ is not applicable in our model) and $\lambda > 0$. The cumulative distribution function for x is:

$$F(x) = 1 - e^{-\lambda x}, \tag{5}$$

for $x \geq 0$. The parameter $\lambda = \frac{1}{\mu}$ where μ is the mean of IPF. Intuitively, to assume IPFs are exponentially distributed is to assume that most of the IPFs for a given tree page are smaller than the mean μ while there exists a few of them that are larger, possibly much larger, than μ. In Section 4.2, we propose a formula that approximates the value of μ based on empirical analysis. We now apply the exponential distribution to our cost model:

$$P\{x \geq b\} = 1 - F(b). \tag{6}$$

By assuming that IPFs are exponentially distributed, we now have a way to compute $P\{x \geq b\}$. However, this distribution does not apply to the first access of any tree page since this process has a probability of 1 in causing an I/O. We therefore need to estimate the number of tree nodes traversed during an R-tree join and treat it as the number of compulsory I/Os required by the join process. We propose to use the sum of the number of tree pages from both R-trees to approximate the compulsory I/Os. This approximation is reasonable because, based on our and other empirical studies [3, 9], all tree pages from both trees will almost always be accessed at least once during an R-tree join over uniformly distributed spatial data.

3.2.4 Expected I/Os for An R-tree Join

Based on ZEIO derived by Equation 3, and $P\{x \geq b\}$ computed by Equations 5 and 6, we can now compute the expected I/O (EIO) for an R-tree join below:

$$EIO = n + m + (ZEIO - n - m) \times P\{x \geq b\} \tag{7}$$

where n and m are the numbers of pages for the two R-trees respectively. The first two terms $(n + m)$ represent the total number of first accesses for all tree pages traversed during the join process. The term $(ZEIO - n - m)$ represents the expected number of non-first accesses with the probability of one such access causing a page fault being $P\{x \geq b\}$.

4 Performance Evaluation

4.1 Experimental Data Sets

To evaluate the accuracy of our cost model, we experiment with three groups of real GIS map data sets. The first group (G1) contains a street map of 75,553 objects and a river and railway map of 77,215 objects in an area of California. The second group (G2) consists of a road map of 97,972 objects and a hydrography (rivers, lakes, etc.) map of 28,347 objects in an area of Wisconsin. Both groups of data were derived from the TIGER/Line files distributed by the US Census Bureau [4]. The third group of data (G3) is extracted from the Sequoia Storage Benchmark [23] which contains an island map of 50,316 objects and a stream map of 127,670 objects.

4.2 Experimental Setup

All experiments are conducted on a SUN Sparc-20 workstation running the Unix operating system. The R-tree join process is implemented using the state-of-the-art techniques proposed in [3]. We use the R-tree with the quadratic split algorithm proposed in [8]. For each group of data sets, we conducted a series of R-tree joins varying buffer sizes from 100 to 1600 K-bytes and collected the I/Os. These costs correspond to the real costs of R-tree joins under various buffer sizes. Next, for each data group, we compute its ZEIO using Equation 3. In order to model the inter-access page fault (IPF) using the exponential distribution, the λ parameter in the cumulated distribution function (Equation 5) must be determined in order to be plugged into the formula. Recall that $\lambda = \frac{1}{\mu}$ where μ can be viewed as the mean of the distribution. We will present how we approximate the μ value below.

4.3 Approximating the Value of μ

Our heuristic in approximating μ is based on the hypothesis that the value of μ is positively associated with the average number of accesses of each tree page during the join process. To explain this, we use the following example. Let N be a tree page and s be the number of accesses of N during a join process. The MBR of N therefore overlaps with the MBRs of s tree pages from the other tree during the join process. Assume that the s accesses of N exhibit a high locality (see Section 3.2) and M is a page from the other tree such that the evaluation of N and M for MBR overlap traverses N for the first time. It is probable that most of the other $s-1$ pages whose MBRs overlap with that of N will be accessed at least once before M is accessed again (hint: a high locality exists among all accesses of N). Therefore, a higher s value will contribute to a higher probability for an IPF between two consecutive accesses of M. Based on this observation, we hypothesize that a positive linear relation exists between the average s and the average IPF (and thus μ) for all pages.

Since we expect a linear correlation exists between s and μ, we set $\mu = c \times s$ where c is a linear factor. In fact, $s = \frac{ZEIO}{n+m}$ where n and m are the number of pages for the two R-trees respectively. Therefore, we approximate the value of μ by the following:

$$\mu = c \times \frac{ZEIO}{n+m}. \tag{8}$$

Since c is a coefficient that cannot be determined theoretically, we conduct experiments based on G1 and G2 to measure the effect of c on our cost model. The results in Figures 1 and 2 both indicate that a value of c between 3 and 4 may be the best approximation. In this work, we set $c = 3.3$.

Table 1 displays the various parameters for all data groups. Note that the term $\frac{\widehat{DA}}{n+m}$ captures the data density characteristics for R-tree joins, and is different for each of

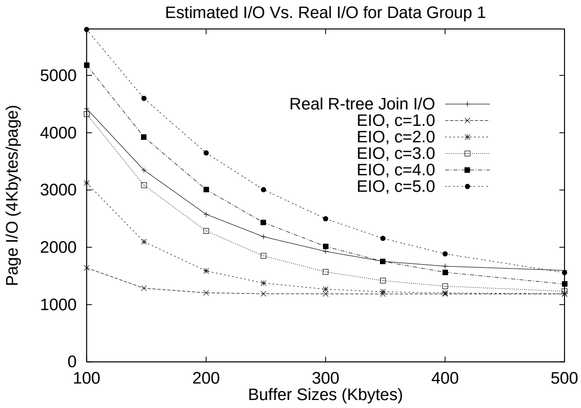

Figure 1. The Effect of Coefficient c (G1).

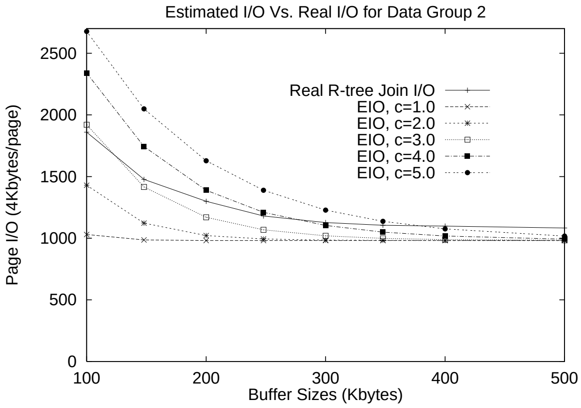

Figure 2. The Effect of Coefficient c (G2).

	n	m	$\widehat{DA}$	$\frac{\widehat{DA}}{n+m}$	c	μ
Data Group 1	601	584	9419	7.95	3.3	26.24
Data Group 2	221	760	5096	5.19	3.3	17.13
Data Group 3	1050	384	10309	7.19	3.3	23.73

Table 1. Parameters for the Three Data Groups G1, G2 and G3.

the three data groups. This means that our three data groups are not homogeneous in terms of distribution density of objects, indicating that our derivation of coefficient c is not a special case based on a unique density. Instead, our estimation of c captures the behavior of μ explained in the example given in the previous section. Different μ values are then derived based on the same value of coefficient c for all three data groups.

4.4 Experimental Results

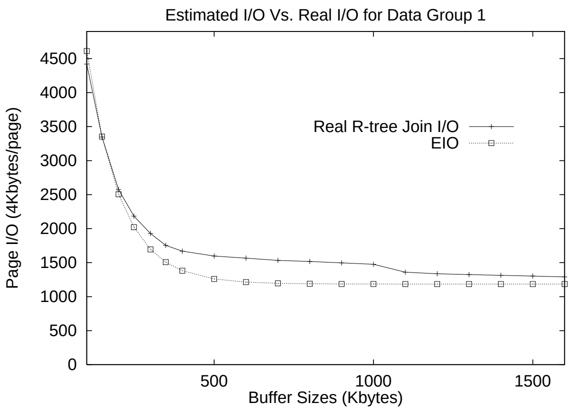

Figure 3. Cost Comparison for G1.

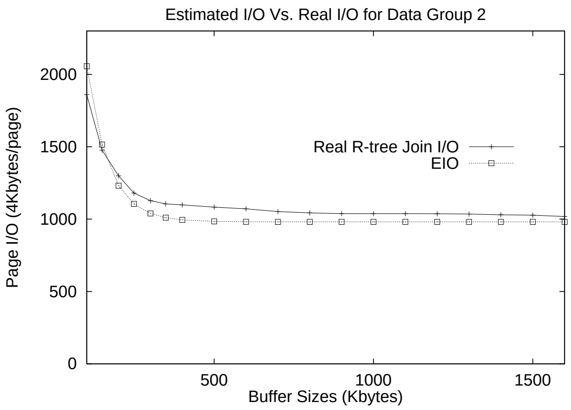

Figure 4. Cost Comparison for G2.

We now compare the EIO (by setting $c = 3.3$) with the real spatial join costs for each data group based on various buffer sizes. Figure 3 shows the difference between the real R-tree join costs and the estimated costs for G1 with various buffer sizes. Similar cost comparisons are displayed in Figure 4 for G2, and in Figure 5 for G3. Figure 6 shows the relative error ratios for all three data groups based on various buffer sizes.

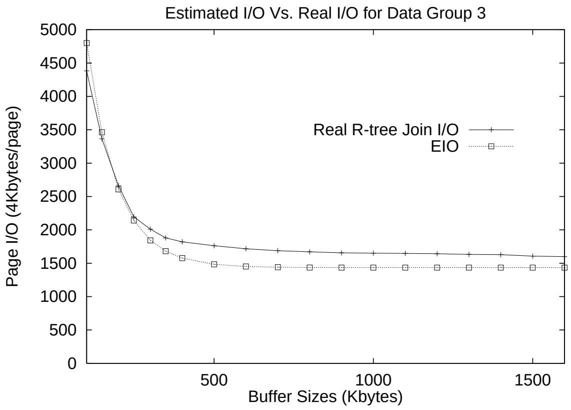

Figure 5. Cost Comparison for G3.

The results in Figures 3 - 6 show that our cost model is most effective when the buffer sizes are small ($< 6\%$ error rate), reaching a maximum relative error ratio of about 20% for medium buffer sizes. The gap closes again for larger buffer sizes. Theoretically, the two curves will meet as the buffer is large enough to cover the locality of the R-tree join I/O patterns. If the locality should not exist for some reason (which is not the case for the GIS data sets we observed), they shall meet when the buffer can hold all tree nodes from both R-trees.

For larger buffer sizes, our model moderately underestimates the real costs for the following three reasons: First, during an R-tree join process, there exist some costs not captured by our model. For example, while our model only captures the downward traversing (see Section 3.1) cost, the real R-tree join processing proposed, for example in [3], is based on a depth-first tree traversal which requires backtracking that may cause additional page faults. It is possible that the query processor may adjust the EIO by incorporating these extra costs (e.g., backtracking), based on the knowledge of which R-tree join technique it uses and possibly of historical data. This adjustment of further fine-tuning the model based on the particular join implementation techniques is beyond our current model which models general R-tree joins. Second, some of our test maps, such as the railway and river map, are not fully evenly distributed as preferred by our model. Third, while the exponential distribution has been widely used to model inter-arrival time in queueing theory [25], it may not 100% fit the I/O patterns of the R-tree join performance. Therefore, we must allow for certain confidence level (in our case 20%).

Constrained by these limitations, our model still achieves an average error ratio of about 10% with a maximum error of 20% as shown in Figure 6. We believe these error rates are within acceptable confidence level of cost prediction for R-tree joins. In summary, this paper represents the first re-

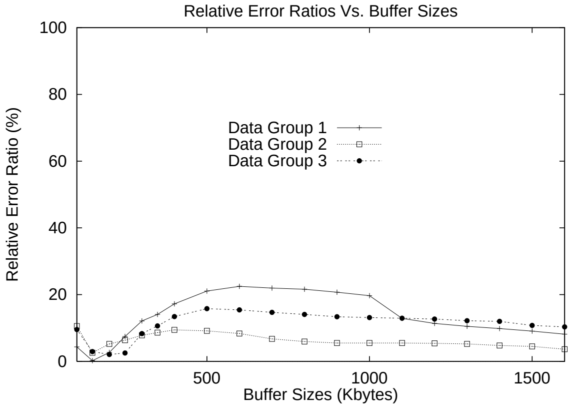

Figure 6. Error Ratios.

sult on a cost model for predicting the performance of spatial joins using R-trees. And, as our experimental studies confirm, the proposed model can be a very effective tool in R-tree join cost estimation during the join query optimization step.

5 Conclusion

This paper presents the first cost model for estimating the performance of R-tree based spatial joins (or R-tree joins). Based on the R-tree window query cost model proposed by [13, 19], our model first measures the number of zero-buffer expected I/Os required during R-tree join processing. To model the buffer impact, we use a probability distribution function to model the paging behavior during the join process. We conduct R-tree join experiments using three groups of real GIS spatial maps to compare the cost estimated by our proposed cost model with actual costs of R-tree join processing on real data sets. Our results show that the average error ratio of our model is about 10% (with a maximum of about 20%), demonstrating that our model can be an effective tool in estimating the R-tree join cost during the join query optimization step.

6 Acknowledgements

We would like to thank our database research group at the University of Michigan for providing an lifely environment in which to conduct this work, and for offering feedback at many stages of our project. In particular, we thank Matthew Jones, Nauman Chaudhry, Stacie Hibino, Anisoara Nica, Lei Zhou, and Amy Lee,

References

[1] Bayer, R. and McCreight, E., "Organization and Maintenance of Large Ordered Indexes", *Acta Informatica,* Vol. 1, No. 3, 1972, pp. 173 – 189.

[2] Bechmann, N., Kriegel, H., Schneider, R., and Seeger, B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", *Proc. of the 1990 ACM SIGMOD Int. Conf. on Management of Data,* May 1990, pp. 322 – 332.

[3] Brinkhoff, T., Kriegel, H., and Seeger, B., "Efficient Processing of Spatial Joins Using R-trees", *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data,* 1993, pp. 237 – 246.

[4] Bureau of Census., "Tiger/Lines Precensus Files: 1990 Technical Documentation", Technical report, Bureau of Census, Washington, D.C., 1989.

[5] Faloutsos, C., Sellis, T., and Roussopoulos, N., "Analysis of Object Oriented Spatial Access Methods," *Proc. of the ACM SIGMOD/PODS Int. Conf. on Principle Of Data,* 1987.

[6] Faloutsos, C. and Kamel, I. "Beyond Uniformity and Independence: Analysis of R-tree Using the Concept of Fractal Dimension," *Proc. of the 1994 ACM SIGMOD/PODS Int. Conf. on Principle Of Data,* 1994, pp. 4 – 13.

[7] Gunther, O., "Efficient Computation of Spatial Joins," *Proc. of the 9th Int. Conf. on Data Eng.,* 1993, pp. 50 – 59.

[8] Guttman, A., "R-tree: a dynamic index structure for spatial searching", *Proc. of the 1984 ACM SIGMOD Int. Conf. on Management of Data,* 1984, pp. 45 – 57.

[9] Huang, Y.W., Jing, N., and Rundensteiner, E.A., "Spatial Join Using R-tree: Breadth-First Traversal with Global Optimizations," *Proc. of the 24th Very Large Data Bases Conference*, 1997, to appear.

[10] Huang, Y.W., Jing, N., and Rundensteiner, E.A., "Integrated Query Processing Strategies for Spatial Path Queries," *Proc. of the 13th Int. Conf. on Data Eng.,* 1997, pp. 477 – 486.

[11] Huang, Y.W., Jing, N., and Rundensteiner, E.A., "Improving Spatial Intersect Joins Using Symbolic Intersect Detection," *Proc. of the 5th Int. Symposium on Spatial Databases,* 1997, to appear.

[12] Intergraph Corporation, "GIS/AM/FM Information," *http://www:intergraph.com/utilmap.shtml*, 1995.

[13] Kamel, I. and Faloutsos, C., “On Packing R-tree,” *Proc. of the CIKM*, 1993, pp. 490 – 499.

[14] Lo, M.L. and Ravishankar, C.V., “Spatial Join Using Seeded Trees,” *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data,* May 1994, pp. 209 – 220.

[15] Lo, M.L. and Ravishankar, C.V., “Spatial Hash-Joins,” *Proc. of the 1996 ACM SIGMOD Int. Conf. on Management of Data,* June 1996, pp. 247 – 258.

[16] Lu, W. and Han, J., “Distance-associated Join Indices for Spatial Range Search,” *IEEE 8th Int. Conf. on Data Engineering,* 1992, pp. 284 – 292.

[17] Nievergelt, J. and Hinterberger, H., “The Grid File: A Adaptable, Symmetric Multikey File Structure”, *ACM Transactions on Database Systems*, Vol. 9, No. 1, Mar. 1984, pp. 39 – 71.

[18] Orenstein, J.A., “Spatial Query Processing in an Object-Oriented Database System”, *Proc. of the 1986 ACM SIGMOD Int. Conf. on Management of Data,* 1986.

[19] Pagel, B., Six, H., Toben, H. and Widmayer, P., “Towards an Analysis of Range Query Performance in Spatial Data Structures,” *Proc. of the 1993 ACM SIGMOD/PODS Int. Conf. on Principle Of Data,* 1993, pp. 214 – 221.

[20] Patel, J.M. and DeWitt, D.J., “Partition Based Spatial-Merge Join,” *Proc. of the 1996 ACM SIGMOD Int. Conf. on Management of Data,* June 1996, pp. 259 – 270.

[21] Rotem, D., “Spatial Join Indices,” *IEEE 7th Int. Conf. on Data Engineering,* 1991, pp. 500 – 509.

[22] Sellis, T., Roussopoulos, N. and Faloutsos, C., “The R^+-Tree: A Dynamic Index for Multi-dimensional Objects,” *Proc. of the VLDB Conf.*, Brighton, England, 1987, pp. 3 – 17.

[23] Stonebraker, M., Frew, J., Gardels, K., and Meredith, J., “The SEQUOIA 2000 Storage Benchmark”, *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data,* 1993.

[24] Stonebraker, M., Rowe, L., and Hirohama, M., “The Implementation of POSTGRES,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, No. 1, 1990, pp. 125-142.

[25] Tanner, M., “Practical Queueing Analysis,” *McGraw-Hill, London, England*, 1995.

[26] Theodoridis, Y. and Sellis, T., “A Model for the Prediction of R-tree Performance,” *Proc. of the ACM SIGMOD/PODS Int. Conf. on Principle Of Data,* 1996.

[27] Ubell, M., “The Montage Extensible DataBlade Architecture,” *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data*, May, 1994.