

Estimating XML Structural Join Size Quickly and Economically

Cheng Luo Zhewei Jiang Wen-Chi Hou Feng Yan Chih-Fang Wang
Department of Computer Science
Southern Illinois University
Carbondale, IL 62901
U.S.A.

{cluozjianghou}@cs.siu.edu fyan1@yahoo.com cfw@cs.siu.edu

Abstract

XML structural joins, which evaluate the containment (ancestor-descendant) relationships between XML elements, are important operations of XML query processing. Estimating structural join size accurately and quickly is thus crucial to the success of XML query plan selection and the query optimization. XML structural joins are essentially complex unequal joins, which render well-known estimation techniques, such as cosine transform, wavelet transform, and sketch, not directly applicable. In this paper, we propose a relation model to capture the structural information of XML data such that the original complex unequal joins are converted to equal joins and those well-known estimation techniques become directly applicable to structural join size estimation. Theoretical analyses and extensive experiments have been performed on these estimation methods. It is shown that the cosine transform requires the least memory and yields the best estimates.

1 Introduction

Boosted by the popularity of the Internet and online applications, Extensible Markup Language (XML) has recently become the de facto standard for presenting, storing, and exchanging data on the Internet. Different from the relational paradigm, XML data are semi-structured and usually modeled as trees. Compared with relational data, XML data are more structure-oriented. Therefore, queries over XML data are usually specified as pattern trees [19] or path expressions [6, 9].

There have been considerable efforts devoted to XML query optimization with the objective to select an efficient query execution plan. Usually, the plan selection is based on the cost estimates of alternative plans, in which the selectivity estimate plays a major role. Existing approaches that estimate the XML query selectivity follow two trends. One

is to estimate the selectivity of path expressions or pattern trees [1, 7, 11, 20]. Methods in this direction rely on some statistics to capture the structures of XML documents. For example, path trees and Markov tables [1] were proposed to aid in estimating the selectivity of simple XML path expressions. Freire, et al. [11] adopted XML Schema types to gather statistics and used histograms to store statistics. Polyzotis, et al. [20] introduced a graph-synopsis model to provide statistical summaries for large XML data graphs. The model exploits localized graph stability to efficiently partition XML data nodes.

The other trend is to identify the key operations performed in the query and then estimate the selectivity of these operations. Structural joins that study the structural relationships between pairs of XML nodes have been recognized as vital operations of XML queries. Due to the importance of structural join operations, a variety of methods have been proposed. While most of them concentrate on efficient execution of structural join operations [14, 22, 25, 2, 17], few [24, 23] address the issue of structural join size estimation, which is nevertheless crucial to the query optimization.

Compared with the first trend, it is usually faster, easier, more flexible, and more precise to estimate structural join sizes between neighboring nodes along query paths. Therefore, in our research, we choose to estimate XML query cost by estimating structural join sizes, that is, to follow the second trend.

Wu, et al. [24] proposed a grid model for XML structural join size estimation. The grid model represents the entire XML dataset as a two-dimensional feature space and partitions this space into predefined grid cells. Each grid cell is associated with a count that indicates the number of nodes that fall in it. Wang, et al. [23] proposed an interval model and a position model. The interval model represents each ancestor node as an interval and each descendant node as a point. The position model represents the structural information in two tables, covering table and start table. The covering table models the structural information of ancestor

nodes while the start table stores the structural information of descendant nodes.

XML structural joins are essentially complex unequal joins, which render well-known estimation techniques, such as cosine transform, wavelet transform, and sketch, not directly applicable to their size estimation [23]. In this paper, we propose an innovative relation model for XML structural join size estimation. Our model captures the structural information of XML data by relations. It converts structural joins to simple equal joins and thus makes those well-known estimation techniques applicable to structural join size estimation. Unlike Wang’s methods [23], which require extensive search on external index structures, these three methods, namely cosine transform, wavelet transform, and sketch, require only simple computations and little memory space for structural join size estimation. In addition, all these methods are significantly faster (for instance, 10^5 times faster) than Wang’s methods and use much less space [15], although Wang’s methods also yield comparable estimation accuracy. Therefore, we shall not compare these methods with Wang’s methods here. Interested readers are referred to [15] for detailed comparisons between Wang’s methods and the cosine transform. Theoretical analyses and extensive experiments have been performed. The experimental results show that, the estimation method using cosine transform requires the least memory space and generates the best estimates among the three methods.

The rest of the paper is organized as follows. Section 2 briefly reviews related research in XML structural join size estimation. Section 3 introduces the three approximation techniques, namely cosine transform, wavelet transform and sketch. Section 4 discusses the relation model and the estimation methods utilizing the three approximation techniques. Section 5 compares the three estimation methods. Detailed theoretical analyses and experimental results are presented. Finally, Section 6 concludes this paper.

2 Preliminaries

To facilitate structural join operations, Wu, et al. [24] proposed a region coding scheme, which is similar to the one adopted in the Niagara [25] project. Specifically, the coding scheme assigns a pair of values, *start* and *end*, called the region codes, to each node in the XML data tree. The region codes specify the nodes’ locations and coverage. A structural join between an ancestor node *a* and a descendant node *d* is essentially to evaluate the logical expression of $a.start \leq d.start \ \&\& \ d.end \leq a.end$.

Existing models for structural join include grid, interval and position models [23, 24]. The grid model [24] is a two-dimensional model. It represents the entire XML dataset as a two-dimensional feature space and partitions this space into predefined grid cells. Each grid cell rep-

resents a range of *start* region codes and a range of *end* region codes. It keeps count of XML nodes that fall in it. The structural join size is estimated by examining the spatial relationships between the grid cells based on the assumption that XML nodes are uniformly distributed in the two-dimensional space.

Wang, et al. [23] proposed the interval model and the position model. The interval model represents each ancestor node as an interval and each descendant node as a point. For each node with region code (*start*, *end*), it is represented as an interval of [*start*, *end*] when it acts as an ancestor and it is represented as a point of value *start* when it acts as a descendant. The position model stores the structural information of XML data in two tables, covering table and start table. Both tables have two attributes. The covering table has attributes position and coverage. The coverage attribute indicates the number of ancestor nodes that cover each respective position. The start table has attributes position and start. The start attribute indicates the number of descendant nodes that start at each respective position.

Fourier transform is a versatile linear transform that decomposes a waveform or function as a sum or integral of sinusoidal functions multiplied by some coefficients (amplitudes). As a variant of the Fourier transform, the Cosine transform is known to have excellent energy compaction properties, where most of the signal information tends to be concentrated in a few low-frequency components of the transform [5]. Therefore, the original waveform or function can be approximated, without much information loss, by its first few terms.

Wavelet transform is another special form of mathematical transforms. It decomposes the original signal by applying highpass and lowpass filters repeatedly until a predefined decomposition level is reached. The Haar transform is conceptually the simplest wavelet transform. For the Haar transform, it is proved [18] that the largest coefficients in absolute value carry the most important information of the original signal. Thus the original signal can be compressed using a few coefficients that have large absolute values.

Recently, sketch [3, 4] is proposed for join size estimation. The basic idea of sketch is to use a family of four-wise independent variables $\{\xi_i\}$, where each $\xi_i \in \{-1, 1\}$ and $\text{probability}[\xi_i = 1] = \text{probability}[\xi_i = -1] = 1/2$ (i.e., $E[\xi_i] = 0$), to generate an estimate of the original function, such that the variance of the estimate is bounded.

Cosine transform, wavelet transform, and sketch have been successfully applied in selectivity estimation. However, it is not clear how these techniques can be applied to structural join size estimation [23]. In this paper, we propose a relation model to accomplish this task.

3 Technical Background

In this section, we briefly discuss the Cosine transform, wavelet transform, and the sketch techniques.

3.1 Cosine Transform

The Cosine transform is a special form of mathematical transforms that attempt to approximate the original signal using only a few coefficients.

3.1.1 Normalization

Let X be an attribute of a relation. We attempt to describe the distribution of X by a mathematical function f . To simplify the notations and implementation of the Cosine transform, we opt to normalize the domain of the function to a predetermined domain $[0, 1]$. Let $maxX$ and $minX$ be the maximum and minimum value of X , respectively. Then, each value $x \in X$ is normalized as follows:

$$x^z = \frac{x - minX}{maxX - minX} \quad (1)$$

where x^z denotes the normalized value of x . For example, the set of x values, $\{0, 1, 2, 3, 4\}$, is normalized to $\{0, 1/4, 1/2, 3/4, 1\}$.

3.1.2 Cosine Transform

Without loss of generality, and especially for the XML range coding scheme, we assume the domain of the interested attribute X is $\{0, 1, 2, \dots, n\}$. Let N be the total number of tuples in the relation. Let v_i be the i th value in the domain of X and v_i^z be its normalized value, namely i/n . Let $count_{v_i}$ be the count of v_i appearing in the relation. The distribution function of X on the normalized domain $[0, 1]$, $f(x)$, is defined as

$$f(x) = n \sum_{i=0}^n count_{v_i} I_{v_i^z}(x) \quad (2)$$

where $I_{v_i^z}$ is an indicator function that is defined as: $I_{v_i^z}(x) = 1$, if x falls in the interval $[v_i^z - 1/2n, v_i^z + 1/2n)$; otherwise, $I_{v_i^z}(x) = 0$. The distribution function is a continuous function and has the following property: $\int_0^1 f(x)dx = N$.

By the theory of Cosine transform, a distribution function $f(x)$ can be represented as

$$f(x) = \sum_{k=0}^{\infty} \alpha_k \phi_k(x)$$

where $\phi_k(x) = N$ if $k = 0$; otherwise, $\phi_k(x) = \sqrt{2} \cos k\pi x$. α_k is obtained as the inner product of the

functions f and ϕ_k , denoted as $\alpha_k = \langle f, \phi_k \rangle$. The Cosine transform is known to have excellent energy compaction properties, where most of the signal information tends to be concentrated in a few low-frequency components of the transform [5]. Therefore, in practice, $f(x)$ is often approximated, without much information loss, by its first few terms as

$$f(x) \approx \sum_{k=0}^m \alpha_k \phi_k(x)$$

where m is a small number.

There often does not exist a simple analytical expression for the function $f(x)$, which makes $\alpha_k = \langle f, \phi_k \rangle$ difficult to derive. In practice, α_k is usually computed as:

$$\alpha_k \approx \sum_{j=1}^N \phi_k(t_j^z) = \sum_{i=1}^n count_{v_i} \phi_k(v_i^z) \quad (3)$$

where t_j^z is the normalized X attribute value of the j th tuple in the relation. v_i is the i th value of the X attribute and its normalized value is v_i^z , and $count_{v_i}$ is the number of tuples in the relation whose X values are v_i .

Example Consider a function $f(x)$ defined on the normalized domain $[0, 1]$. The values of $f(x)$ are listed as : $f(0) = 1, f(0.32) = 2, f(0.56) = 1, f(1) = 1$. The Cosine transform of $f(x)$ is derived as follows. Given that $\phi_k(x)$ is $\sqrt{2} \cos k\pi x$ for $k \geq 1$, α_k is calculated as:

$$\begin{aligned} \alpha_0 &= 5 \\ \alpha_1 &= \sum_{all\ x} f(x) \phi_1(x) \approx 1.2504 \\ \alpha_2 &= \sum_{all\ x} f(x) \phi_2(x) \approx 0.3092 \\ \dots &\dots \dots \end{aligned}$$

Finally, function $f(x)$ is approximated as $f(x) \approx 5 + 1.2504\sqrt{2} \cos \pi x + 0.3092\sqrt{2} \cos 2\pi x + \dots$.

3.2 Wavelet Transform

Wavelet Transform [8] is another special form of mathematical transforms that attempt to compress a signal using a few coefficients.

Our application focuses on the Haar wavelet transform, which is conceptually the simplest and has been used in various database applications [16, 12]. Haar wavelet transform is usually computed by recursive averaging and differencing, as illustrated below by an example.

Example Consider a series of numbers, listed as $\{12, 9, 3, 7, 1, 8, 4, 6\}$. Note that the size of the series is deliberately selected as a power of 2, which simplifies the computation and analysis. The computation of its Haar transform H_f takes 3 stages.

Stage 1:

$$\begin{aligned} H_{f_1} &= \frac{\{12 + 9, 3 + 7, 1 + 8, 4 + 6\}}{\sqrt{2}} \\ H_{f_2} &= \frac{\{12 - 9, 3 - 7, 1 - 8, 4 - 6\}}{\sqrt{2}} \\ H_f &= H_{f_1} \cup H_{f_2} \\ &= \{21, 10, 9, 10, 3, -4, -7, -2\}/\sqrt{2} \end{aligned}$$

Stage 2:

$$\begin{aligned} H_f &= \frac{\{\frac{21+10}{\sqrt{2}}, \frac{9+10}{\sqrt{2}}, \frac{21-10}{\sqrt{2}}, \frac{9-10}{\sqrt{2}}, 3, -4, -7, -2\}}{\sqrt{2}} \\ &= \{\frac{31}{\sqrt{2}}, \frac{19}{\sqrt{2}}, \frac{11}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 3, -4, -7, -2\}/\sqrt{2} \end{aligned}$$

Stage 3:

$$\begin{aligned} H_f &= \frac{\{\frac{31+19}{(\sqrt{2})^2}, \frac{31-19}{(\sqrt{2})^2}, \frac{11}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 3, -4, -7, -2\}}{\sqrt{2}} \\ &= \{\frac{50}{(\sqrt{2})^2}, \frac{12}{(\sqrt{2})^2}, \frac{11}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 3, -4, -7, -2\}/\sqrt{2} \\ &\simeq \{17.68, 4.24, 5.5, -0.5, 2.12, -2.83, -4.95, -1.41\} \end{aligned}$$

To summarize, let the size of a signal be len . Then, its Haar transform takes \log_2^{len} stages to compute. In stage i , only the first $len/2^{i-1}$ elements change their values. The new values are resulted from pair-wise averaging and then differencing of the first $len/2^i$ elements in the previous stage $i - 1$. The computation of Haar wavelet is $O(len)$, which is much faster than the computation of Cosine transform whose complexity is $O(len^2)$.

The entries in the Wavelet transform are referred to as Wavelet coefficients. For Haar transform, it is proved [18] that the largest coefficients in absolute value carry the most important information and thus should be selected to represent the original signal. Assume that we are to represent the original series in the above example by three coefficients. Then, they are $\{17.68, 5.5, -4.95\}$.

3.3 Sketch

The sketch technique represents the original signal by a series of atomic sketches [3], which can be used later to estimate the join size. Suppose the original discrete signal S is of length n , its atomic sketch is computed as follows:

- Generate a four-wise independent random family F of length n , whose elements are either $+1$ or -1 .
- Let A_s stand for the atomic sketch, then $A_s = \sum_{i=1}^n S_i \cdot F_i$.

4 Estimating XML Structural Join Size

In this section, we develop a relation model to capture the structural information of XML data so that an XML structural join, which is essentially a complex un-equal join, can be modeled as an equal-join. This model makes those well known selectivity estimation techniques, such as cosine transform, wavelet transform, and sketch, applicable to structural join size estimation.

4.1 Assumptions and Definitions

An XML dataset usually consists of multiple XML documents. To ensure a coherent region coding scheme, all documents are first integrated into one single XML document, which is often accomplished by creating a pseudo root tag above all the existing root tags of the XML documents. Hereafter, for simplicity, we shall assume an XML dataset consists of only one document.

An XML document is generally represented by an XML data tree \mathcal{T} . We assume an XML data tree encompasses all the information of the original XML dataset, with tags, attributes, and text data of the dataset represented by nodes and their relationships indicated by edges in the tree. Consequently, queries on an XML dataset can be specified against its XML data tree.

Without loss of generality, we assume that the region coding assigns a pair of *integer* values (*start*, *end*) to each node in the XML data tree. The root node has the region code $(0, n)$, where n is the smallest integer number for the root node to cover all of its descendants. The region $[0, n]$ is also termed the *coding domain Dom*.

A *predicate* is a selective condition based on the values and structures of the XML data. For instance, a predicate like "tag name=student" selects tags whose names equal *student*. Evaluating a predicate against the XML data tree returns the set of nodes that satisfy the predicate. For simplicity, hereafter we shall call nodes that satisfy a predicate p the p nodes.

4.2 A Relation Model for Structural Join

The range coding scheme [24] captures the structural information of XML data elegantly. A node d is a descendant of a node a iff $a.start \leq d.start \ \&\& \ d.end \leq a.end$. Wang, et. al [23] use two models to capture this structural information. In their Interval Model (IM), each node is represented by an interval (for use as an ancestor in the query) and a point (for use as a descendant). In their Position Model (PM), each position in the covering table is associated with the number of nodes covering it and in the start table, is associated with the number of nodes starting

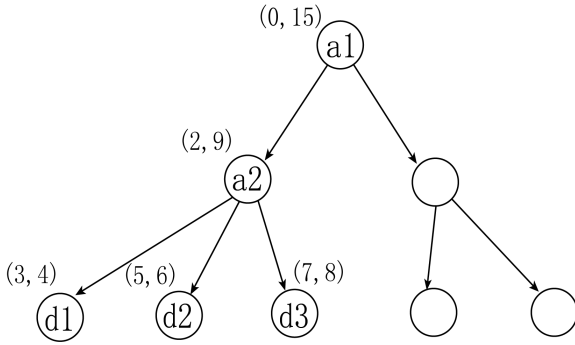


Figure 1. An XML Data Tree

at it. In this research, we represent the structural information of XML data as relations, so that a structural join of XML data can be modeled as an equal join of relations. Our model indeed can be viewed as a semantic extension of Wang's position model [23].

For each predicate p , we model it as two relations, called the Coverage and Start-position relations. Both relations have two attributes, Sequence number and Position. The Sequence number attribute is introduced to avoid generating duplicate tuples in a relation. The Position attribute has the coding domain Dom as its domain. For each p node with a region code $(start, end)$, it is represented by $end - start + 1$ tuples whose Position attribute values are $start, start + 1, \dots, end$ in the Coverage relation for p , and as a single tuple whose Position value is $start$ in the Start-position relation for p . The Coverage relation is used when p acts as an ancestor predicate, while the Start-position relation is used when p acts as a descendant predicate in a structural join.

Figure 1 shows an example of an XML data tree, in which two nodes, namely $a1$ and $a2$, satisfy the predicate A , and three other nodes, namely $d1$, $d2$ and $d3$, satisfy the predicate D . Nodes that are of interest also have their region codes labeled close by. For example, node $d2$ has region codes $(5, 6)$.

Tables 1 through 4 show how predicates A and D are represented in the relation model. Each node with a region code $(start, end)$ has $end - start + 1$ tuples in the Coverage relation and a single tuple in the Start-position relation. In the coverage relation, the Position attributes of these tuples are from $start$ to end . In the Start-position relation, the tuple's position attribute value is $start$.

Consider a structural join between predicates A and D , where A is the ancestor predicate and D is the descendant predicate. The structural join between A and D is to find all pairs of a, d such that $a \in A, d \in D$ and a contains d . It is equivalent to find all pairs of tuples from A 's Coverage relation and D 's Start-position relation, respectively, such that the two tuples have the same Position attribute.

Table 1. Coverage Relation for A

Sequence No.	Position
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11
13	12
14	13
15	14
16	15
17	2
18	3
19	4
20	5
21	6
22	7
23	8
24	9

Table 2. Start-position Relation for A

Sequence No.	Position
1	0
2	2

Table 3. Coverage Relation for D

Sequence No.	Position
1	3
2	4
3	5
4	6
5	7
6	8

Table 4. Start-position Relation for D

Sequence No.	Position
1	3
2	5
3	7

Lemma 1 *The structural join between any pair of ancestor predicate A and descendant predicate D is the equal-join between A 's Coverage relation and D 's Start-position relation on the Position attribute.*

The above reasoning can be easily extended to multi-structural join. For example, consider a query $//A_1//A_2//D$, where only the deepest predicate acts as the descendant while the other predicates the ancestors. Theorem 1 generalizes the multi-structural join size computation.

Theorem 1 *The structural join among ancestor predicates A_1, A_2, \dots, A_n and a descendant predicate D , is the equal-joins among the Coverage relations of A_1, A_2, \dots, A_n and D 's Start-position relation on the Position attribute.*

4.3 Join Size Computation

To compute the equal-join size, we need to know the number of occurrences of each position in the relations. For each predicate, we define two functions, the Coverage and Start-position functions, to describe the distributions of the Position attributes in the Coverage and Start-position relations, respectively.

Definition 1 *The coverage function of a predicate p at position $pos \in \text{Dom}$, denoted as $C_p(pos)$, is the number of p nodes whose region codes cover pos or the number of tuples whose Position attributes are pos in the Coverage relation of p .*

Definition 2 *The start-position function of a predicate p at position $pos \in \text{Dom}$, denoted as $S_p(pos)$, is the number of p nodes whose start region codes equal pos or the number of tuples whose Position attribute is pos in the Start-position relation of p .*

Figures 2 through 5 show the corresponding distribution functions for the two predicates A and D in the example XML data tree. For instance, $C_A(6)$ equals 2 because both A nodes, namely $a1$ and $a2$, cover position 6.

The coverage function is intended for use when the predicate acts as the ancestor in a structural join and the start-position function is used when the predicate acts as the descendant. The value of $C_p(pos)$ can be greater than 1 since the nodes satisfying p and covering pos might be nested; the value of $S_p(pos)$ can only be either 0 or 1 because no nodes can have two identical start region codes.

Thus, the structural join size between any pair of ancestor predicate A and descendant predicate D is the inner product of $C_A(pos)$ and $S_D(pos)$, denoted as $\langle C_A, S_D \rangle$, over the coding domain, namely:

$$\sum_{pos \in \text{Dom}} C_A(pos) \times S_D(pos) \quad (4)$$

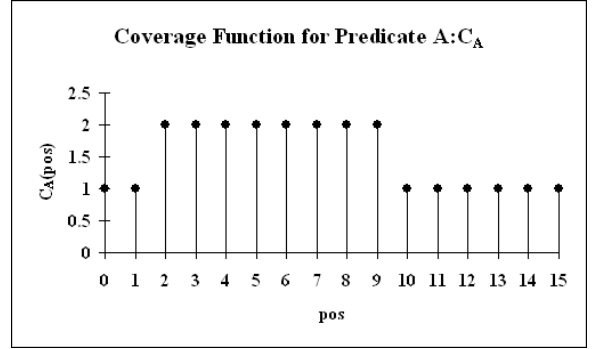


Figure 2. Coverage Function for A

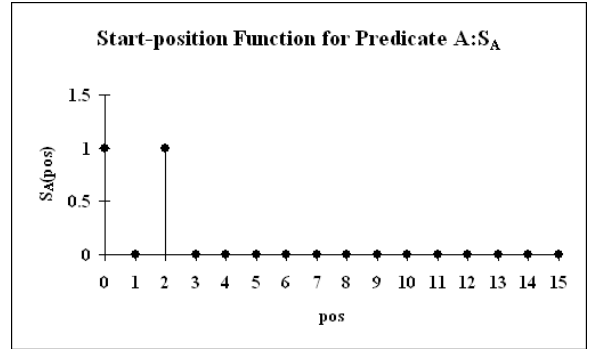


Figure 3. Start-position Function for A

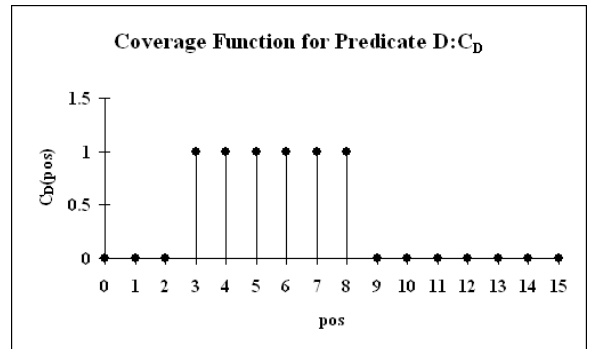


Figure 4. Coverage Function for D

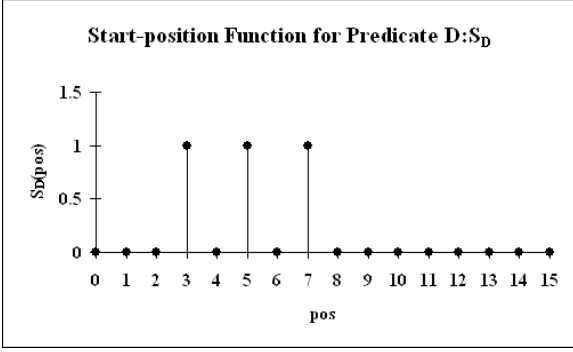


Figure 5. Start-position Function for D

Moreover, the structural join size among any ancestor predicates A_1, A_2, \dots, A_n and a descendant predicate D , is the inner product of $\mathcal{C}_{A_1}(pos), \mathcal{C}_{A_2}(pos), \dots, \mathcal{C}_{A_n}(pos)$ and $\mathcal{S}_D(pos)$, denoted as $\langle \mathcal{C}_{A_1}, \mathcal{C}_{A_2}, \dots, \mathcal{C}_{A_n}, \mathcal{S}_D \rangle$, over the coding domain, namely: $\sum_{pos \in Dom} \mathcal{C}_{A_1}(pos) \times \dots \times \mathcal{C}_{A_n}(pos) \times \mathcal{S}_D(pos)$.

4.4 Structural Join Size Estimation

The storage of the distribution functions can consume a lot of space. Therefore, we opt to use three techniques, namely the cosine transform, wavelet transform, and sketch, that need little storage space to approximate the distribution functions. In addition, unlike Wang's approaches[23] that could require a lot of disk accesses, these join size estimation methods need only simple computations. In this section, we shall mainly focus on the cosine transform as the other two methods, namely the wavelet transform and sketch, have been discussed thoroughly in the literature [3, 4, 8, 18].

4.4.1 Estimation via the Cosine transform

Recall that before applying the Cosine transform to a function, a normalization of the function's domain is performed first. In our case, the coding domain Dom , namely $[0, n]$, is mapped to the region $[0, 1]$ and each position pos in Dom is normalized to pos^z by formula 1 accordingly. For example, $0^z = 0$ and $n^z = 1$.

We denote the coverage function \mathcal{C}_A on the new domain $[0, 1]$ as \mathcal{C}'_A , such that $\mathcal{C}_A(pos) = \mathcal{C}'_A(pos^z)$. Likewise, the start-position function \mathcal{S}_D is redefined on $[0, 1]$ as \mathcal{S}'_D and $\mathcal{S}_D(pos) = \mathcal{S}'_D(pos^z)$.

The two new distribution functions can now be expressed in Cosine series as: $\mathcal{C}'_A(pos^z) = \sum_{k=0}^{\infty} a_k \phi_k(pos^z)$ and $\mathcal{S}'_D(pos^z) = \sum_{k=0}^{\infty} b_k \phi_k(pos^z)$ respectively, where $a_k = \langle \mathcal{C}'_A, \phi_k \rangle$, and $b_k = \langle \mathcal{S}'_D, \phi_k \rangle$.

Now the structural join size computation (Equation 4)

can be rewritten as:

$$\begin{aligned}
& \sum_{pos \in Dom} \mathcal{C}_A(pos) \times \mathcal{S}_D(pos) \\
&= \sum_{pos=0}^n \mathcal{C}_A(pos) \times \mathcal{S}_D(pos) \\
&= \sum_{i=0}^n countA_i \cdot countD_i
\end{aligned} \tag{5}$$

where $countA_i$ is the number of tuples whose Position values are i in the coverage relation for A and $countD_i$ is the number of tuples whose Position values are i in the start-position relation for D .

On the other hand, by definition of the distribution function 2, we obtain

$$\begin{aligned}
\langle \mathcal{C}'_A, \mathcal{S}'_D \rangle &= n^2 \sum_{i=0}^n countA_i \cdot countD_i \cdot \int_{i-1/2n}^{i+1/2n} I_i(x) dx \\
&= n \sum_{i=0}^n countA_i \cdot countD_i
\end{aligned} \tag{6}$$

By Parseval's identity [13], the following equation holds:

$$\langle \mathcal{C}'_A, \mathcal{S}'_D \rangle = \sum_{k=0}^{\infty} a_k \times b_k \tag{7}$$

From Equations 5, 6 and 7, the structural join size is computed as $\frac{1}{n} \sum_{k=0}^{\infty} a_k \times b_k$.

The join size can be approximated by using only the first m coefficients of each series as $\frac{1}{n} \sum_{k=0}^{m-1} a_k \times b_k$.

4.4.2 Estimation via wavelet transform

The Haar transforms of the distribution functions can be computed as discussed before. The structural join size between an ancestor predicate A and a descendant predicate D is equal to the inner product between A 's coverage function and D 's start-position function, which is also equal to the inner product of their respective Haar transform [16].

The structural join size between an ancestor predicate A and a descendant predicate D is estimated as follows. Firstly, the m largest coefficients in absolute value of A 's and D 's Haar transforms are selected respectively. Secondly, those non-selected coefficients are assumed to be 0s. Finally, the estimation is carried out by computing the inner product of the new coefficients.

4.4.3 Estimation via sketch

Consider a structural join between an ancestor predicate A and a descendant predicate D . We attempt to use $s_1 \cdot s_2$ atomic sketches to approximate each predicate, where s_1

Table 5. Statistics for XMARK

Tag Name	Node Count
annotation	21750
closed_auction	9750
emailaddress	25500
item	21750
itemref	21750
name	48250
open_auction	12000
person	25500
quantity	43500
seller	21750
text	105114
type	21750

Table 6. Statistics for DBLP

Tag Name	Node Count
article	213949
author	1331301
cite	172406
incollection	1535
inproceedings	357848
title	581570
year	580420

determines the estimation accuracy and s_2 determines the estimation confidence. After the computation of atomic sketches, we multiply each pair of them, which represent the ancestor and the descendant predicates respectively and use the same instantiation of four-wise independent random variable family. Let us call the products J_i 's. Totally there are $s_1 \cdot s_2$ J_i 's. The J_i 's have s_2 groups with s_1 members in each group. The structural join size estimate is the median of the averages for the groups.

5 Experimental Results

We have implemented all three estimation methods, namely the cosine transform, the Haar wavelet transform, and sketch. We conducted experiments on a PC with a Pentium 4 processor and 256M RAM.

The datasets we used include a synthetic XML benchmark dataset XMARK [21] and a real XML database DBLP [10]. For each dataset, we select pairs of predicates by the element tag name and then perform a structural join on each pair of predicates. Table 5 and 6 show the selected predicates and their detailed statistics. Table 7 and 8 show the set of queries performed on each data set.

Table 7. Queries on XMARK

Query	Ancestor	Descendant
Q1	closed_auction	annotation
Q2	closed_auction	quantity
Q3	closed_auction	seller
Q4	item	name
Q5	item	quantity
Q6	person	name
Q7	person	emailaddress
Q8	open_auction	itemref
Q9	open_auction	quantity
Q10	open_auction	seller
Q11	open_auction	text
Q12	open_auction	type

Table 8. Queries on DBLP

Query	Ancestor	Descendant
Q1	inproceedings	author
Q2	inproceedings	title
Q3	inproceedings	cite
Q4	article	title
Q5	article	cite
Q6	incollection	year

5.1 Storage Space

The cosine transform and the sketch methods estimate the structural join size by processing the products of the cosine coefficients or atomic sketches. Only the cosine coefficients or atomic sketches need to be stored for estimation.

As for the Haar transform method, it needs not only to store the wavelet coefficients but also record their indexes so that the coefficients can be correctly matched. Therefore, it needs twice as much space as the other two methods for storing the same number of coefficients or atomic sketches.

5.2 Estimation Error

The relative estimation error is used as the metric to determine the accuracy of the estimates. It is defined as $\frac{|x - \hat{x}|}{x} \times 100\%$, where x is the actual join size and \hat{x} is the estimate.

The estimation accuracy is measured based on various memory constraints: 200, 400, and 800 bytes. A 200-byte memory constraint would mean 50 cosine coefficients, or 50 atomic sketches, or 25 wavelet coefficients, recalling that each wavelet coefficient is stored with its index.

In general, sketch performs much worse than the other two methods. Therefore, we separate its results from the others so that the comparisons between the cosine and

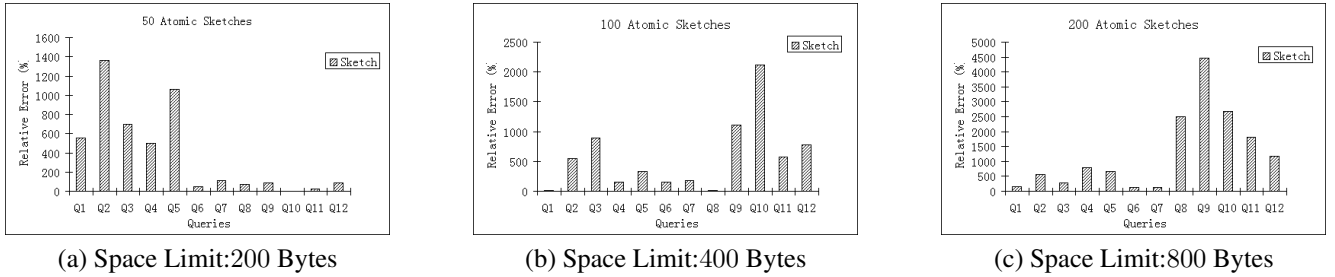


Figure 6. Sketch on XMARK

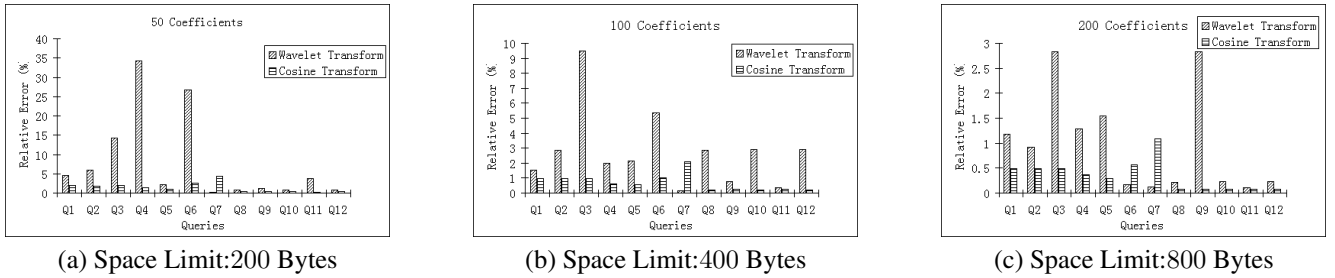


Figure 7. Performance on XMARK

wavelet methods are clearer. Figure 6 shows the performance of the sketch method on the XMARK dataset. The performance of the sketch method on the DBLP dataset is very similar and is thus omitted. Figure 6 has three subfigures that correspond to the three memory space constraints respectively. The x axis denotes the queries executed and the y axis shows the absolute value of the relative estimation error.

As shown by Figure 6, the sketch method performs very poorly in structural join size estimation. The relative estimation error is very large. For example, query 10 in subfigure (b) has a relative estimation error 20 times as high as the actual join size. In addition, the variance of the relative estimation error is also quite large, which is demonstrated by the huge performance variation of query 9 under the 400 and 800 byte constraints. The titles of the subfigures indicate the total number of atomic sketches used for estimation.

Figure 7 shows the performance of the cosine transform and wavelet transform methods on the XMARK dataset while figure 8 shows their performance on the DBLP database. Results from both methods are placed side by side for easy comparison. The titles of the subfigures indicate the total number of cosine coefficients used for estimation.

As shown by Figures 7 and 8, overall the cosine method performs better than the wavelet method on both XML datasets. Indeed, the estimation accuracy of the cosine method is generally 2 ~ 3 times better than that of the wavelet method. Even with the same number of coefficients for both methods, the cosine method is still generally better than the wavelet method, which can be observed by com-

paring their performance in neighboring figures.

5.3 Estimation Time

All three methods perform simple mathematical calculations to estimate the structural join size. For the cosine transform and the Haar transform methods, the estimation is performed by summing up the products of pairs of coefficients. For the sketch method, the estimate is derived by selecting the median of the averages for the groups of products of pairs of atomic sketches.

Consider using 400 pairs of coefficients or atomic sketches (with $s_1 = s_2 = 20$) as an example. To derive a join size estimate, it takes less than 80, 80 and 82 us for the cosine transform, wavelet transform and sketch methods, respectively. In short, all the three methods estimate the XML structural join size quickly.

6 Conclusions

Well-known estimation techniques, such as cosine transform, wavelet transform, and sketch, have been applied successfully in selectivity estimation. These methods perform simple calculations and require little memory space. They provide a quick and economical estimation approach. However, no existing models make these techniques applicable to XML structural join size estimation.

In this paper, we propose an innovative relation model for XML structural join size estimation. Our model captures the structural information of XML data by relations.

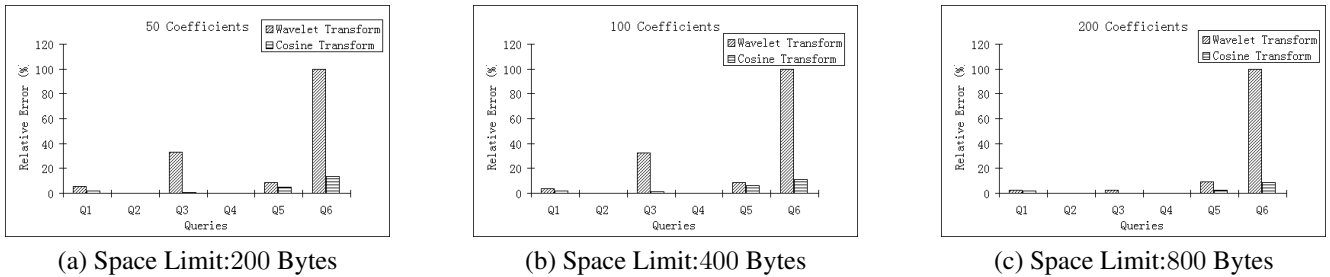


Figure 8. Performance on DBLP

It converts structural joins, which are essentially complex unequal joins, to simple equal joins and makes those well-known estimation techniques applicable to structural join size estimation.

Theoretical analyses and extensive experiments have been performed. The experimental results show that, the estimation method using cosine transform requires the least memory space and generates the best estimates among the three methods.

References

- [1] A. Aboulnaga, A. R. Alameldeen, and J. F. Naughton. Estimating the selectivity of xml path expressions for internet scale applications. *Proceedings of 27th International Conference on Very Large Data Bases*, pages 591–600, 2001.
- [2] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu. Structural joins: A primitive for efficient xml query pattern matching. *ICDE*, pages 141–152, 2002.
- [3] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems*, pages 10–20, 1999.
- [4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Proceedings of the 28th annual ACM symposium on Theory of computing*, pages 20–29, 1996.
- [5] W. L. Briggs and V. E. Henson. *DFT: an owner's manual for the discrete Fourier transform*. Philadelphia: Society for Industrial and Applied Mathematics Published, 1995.
- [6] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. Xquery 1.0: An xml query language. *W3C Working Draft*, 2004. www.w3.org/TR/xquery/.
- [7] Z. Chen, H. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. T. Ng, and D. Srivastava. Counting twig matches in a tree. *Proceedings of the 17th International Conference on Data Engineering*, pages 595–604, 2001.
- [8] C. K. Chui. *An Introduction to Wavelets*. Academic Press, 1992.
- [9] J. Clark and S. DeRose. XML Path Language (XPath). *W3C Working Draft*, 1999. <http://www.w3.org/TR/xpath>.
- [10] D. data set. <http://www.informatik.uni-trier.de/ley/db/index.html>.
- [11] J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, and J. Siméon. Statix: making xml count. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 181–191, 2002.
- [12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. *Proceedings of the 27th International Conference on VLDB*, pages 79–88, 2001.
- [13] E. Issacson and H. B. Keller. *Analysis of Numerical Methods Theorem 3*. Dover Publications, 1994.
- [14] Q. Li and B. Moon. Indexing and querying xml data for regular path expressions. *VLDB*, pages 361–370, 2001.
- [15] C. Luo, Z. Jiang, W. Hou, and C. Wang. Applying cosine transform to xml structural join size estimation. *manuscript*, <http://www.cs.siu.edu/~cluo/Estimate.pdf>, 2005.
- [16] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. *SIGMOD*, 1998.
- [17] J. McHugh and J. Widom. Optimizing branching path expressions. *VLDB*, pages 315–326, 1999.
- [18] Y. Nievergelt. *Wavelets Made Easy*. Birkhauser, 1999.
- [19] S. Paparizos, S. Al-Khalifa, A. Chapman, H. V. Jagadish, L. V. S. Lakshmanan, A. Nierman, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A Native System for Querying XML. *VLDB J.*, 11(4):274–291, 2002.
- [20] N. Polyzotis and M. N. Garofalakis. Statistical synopses for graph-structured xml databases. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 358–369, 2002.
- [21] A. Schmidt, F. Waas, M. Kersten, D. Florescu, L. Manolescu, M. J. Carey, and R. Busse. The XML benchmark project. Technical report, CWI, 2001.
- [22] D. Srivastava, S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, and Y. Wu. Structural joins: A primitive for efficient xml query pattern matching. *ICDE*, pages 141–152, 2002.
- [23] W. Wang, H. Jiang, H. Lu, and J. X. Yu. Containment join size estimation: models and methods. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 358–369, 2003.
- [24] Y. Wu, J. M. Patel, and H. V. Jagadish. Estimating answer sizes for xml queries. *8th International Conference on Extending Database Technology*, pages 590–608, 2002.
- [25] C. Zhang, J. F. Naughton, D. J. DeWitt, Q. Luo, and G. M. Lohman. On supporting containment queries in relational database management systems. *SIGMOD*, 2001.