# Selectivity Estimation for XML Twigs

Neoklis Polyzotis*
University of Wisconsin-Madison
alkis@cs.wisc.edu

Minos Garofalakis
Bell Labs, Lucent Technologies
minos@research.bell-labs.com

Yannis Ioannidis
University of Athens, Hellas
yannis@di.uoa.gr

## Abstract

*Twig queries represent the building blocks of declarative query languages over XML data. A twig query describes a complex traversal of the document graph and generates a set of element tuples based on the intertwined evaluation (i.e., join) of multiple path expressions. Estimating the result cardinality of twig queries or, equivalently, the number of tuples in such a structural (path-based) join, is a fundamental problem that arises in the optimization of declarative queries over XML. It is crucial, therefore, to develop concise synopsis structures that summarize the document graph and enable such selectivity estimates within the time and space constraints of the optimizer. In this paper, we propose novel summarization and estimation techniques for estimating the selectivity of twig queries with complex XPath expressions over tree-structured data. Our approach is based on the* XSKETCH *model, augmented with new types of distribution information for capturing complex correlation patterns across structural joins. Briefly, the key idea is to represent joins as points in a multidimensional space of path counts that capture aggregate information on the contents of the resulting element tuples. We develop a systematic framework that combines distribution information with appropriate statistical assumptions in order to provide selectivity estimates for twig queries over concise* XSKETCH *synopses and we describe an efficient algorithm for constructing an accurate summary for a given space budget. Implementation results with both synthetic and real-life data sets verify the effectiveness of our approach and demonstrate its benefits over earlier techniques.*

## 1. Introduction

XML is gaining wide acceptance as the standard for data exchange and integration over the Internet. The simple self-describing nature of XML data enables a broad class of new applications that allow users to query effectively the vast amount of information available on the Internet. As a result, XML support is becoming increasingly ubiquitous in commercial database systems, while numerous research projects, like Niagara [10] and Xyleme (http://www.xyleme.com), look into the development of native XML database systems.

The successful deployment of XML query processors hinges upon the existence of high-level languages that allow users to formulate declarative queries over semi-structured data. Examples include XQuery [2] and XSLT [4], the proposed standards by W3C, and a variety of other languages that cover a wide range of querying models and programming styles. A common characteristic in all proposals is the use of pattern specification expressions, built around a path expression language, for selecting subsets of elements that satisfy certain structural relationships (i.e., they are connected with specific paths in the document graph). These expressions, commonly called *twig queries*, describe a complex traversal of the document graph and retrieve document elements through an intertwined (i.e., joint) evaluation of multiple path expressions. As a concrete example, consider the following query in the XQuery language over a document that contains movie data:

```
for $t_0 in //movie[/type=X],
  $t_1 in $t_0/actor,
  $t_2 in $t_0/producer
return $t_1, $t_2
```

The `for` clause corresponds to a twig query with three path expressions. The first path expression retrieves all movies of a specific type X, while the second and third expressions retrieve the actors and producers, respectively, of each qualifying movie. Conceptually, the result of the `for` clause is an unnested representation of this joint evaluation, i.e., a set of 3-tuples, where each tuple pairs a movie element with one possible combination of an actor and a producer that it contains. A qualifying movie, for example, with 10 actors and 3 producers will generate 30 tuples. The final result of the query, which is specified by the `return` clause, is formed by projecting the actor and producer elements from the set of 3-tuples. Overall, twig queries represent the equivalent of the SQL `FROM` clause in the XML world: they model the

∗ Author's current address: University of California Santa Cruz, alkis@cs.ucsc.edu generation of element tuples which will eventually be processed to compute the final result of the XML query.

Given the importance of twig queries as a basic element selection mechanism, their optimized execution is crucial for the efficient implementation of declarative query languages. Similar to the case of relational joins, an important problem in the optimization of these path-join expressions is estimating their selectivity, i.e., the number of matching element tuples that they will generate. This, in turn, hints at the existence of concise summary structures that can provide, at query compile time, effective selectivity estimates within the time and memory constraints of a query optimizer.

In order to provide accurate estimates for twig queries, a synopsis has to approximate reasonably the joint distribution of path result counts for different document elements. The problem, however, is particularly challenging due to the complicated correlation patterns that can arise in the underlying path distribution. If we consider the previous query, for example, we expect to retrieve more actors and producers (and hence produce more tuples) per movie if the type X is "Action" than if it is "Documentary". Similarly, the number of actors and producers can be affected by the paths that reach the qualifying movie elements, or even by paths that exist in the neighborhood of the elements. In general, selectivity estimation for twig queries is equivalent to estimating the result cardinality of relational tree joins with selection predicates, which is already known to be a difficult problem [7]. The problem becomes even harder when path expressions are predicated on the existence of specific paths (branching predicates) since these predicates would correspond to additional semi-join operations in a relational context. Overall, an effective synopsis needs to capture the complex dependencies that arise between and across the path structure and element values in order to provide accurate selectivity estimates for twig queries.

**Related Work**[1] A large body of earlier work has focused on the case of single path expressions [1, 8, 11, 12, 17], i.e., estimating the number of elements reached by a single path. The proposed summarization techniques can be used for twig queries that are equivalent to single path expressions, but it is not clear how they can be extended to handle the estimation problem in its most general form, where the query contains multiple correlated paths.

There are two recent proposals for estimating the selectivity of twig queries over tree-structured data, namely, Correlated Suffix Trees (CSTs) [3] and StatiX [6]. The CST method uses a pruned trie to summarize the path structure of the input data, while StatiX captures the underlying path distribution with one-dimensional histograms on element-ids. In both cases, the proposed estimation framework combines the stored statistics with statistical (independence and uniformity) assumptions, which compensate for the lack of detailed distribution information. The corresponding construction algorithms, however, do not take into account the estimation assumptions and it is thus not clear if those assumptions remain valid in the constructed synopses. In addition, the two techniqeus allocate the space budget equally among all parts of the generated synopses, regardless of the skew that might exist in certain regions of the data.

The problem of building statistical synopses has received a significant amount of attention in the relational literature as well. The focus, however, has been on approximating value distributions [7, 9, 15, 16], while the problem of summarizing hierarchical XML data is certainly more complex.

**Our Contributions** In this paper, we propose a novel summarization method for estimating the selectivity of twig queries over large XML data graphs. Our approach is based on our earlier XSKETCH synopsis model, augmented with a new type of distribution information for approximating the cardinality of structural joins among multiple complex path expressions. More specifically, our contributions can be summarized as follows:

• **Selectivity Estimation for Structural Joins**. We propose a novel summarization method for estimating the cardinality of a structural join among multiple path expressions. Our method maps a structural join to a collection of points in a multidimensional space of *edge counts* and thus captures aggregate information on the structure of matching tuples. The resulting multidimensional distribution of integer counts can then be approximated very effectively using standard summarization techniques, like, histograms or wavelets.

• **Extended XSKETCH Synopses**. Based on our method for approximating join cardinalities, we propose an extended XSKETCH synopsis model that records localized distribution information on structural joins. We develop a systematic framework that combines the stored information with appropriate statistical assumptions in order to compute selectivity estimates for twig queries with complex XPath expressions (i.e., paths with branching and value predicates). Based on the specifics of the estimation framework, we describe a construction algorithm that uses appropriate refinement operations in order to build an accurate summary from an initial coarse synopsis. Compared to previously proposed schemes, our construction algorithm takes directly into account the statistical characteristics of the data with respect to the estimation assumptions. This results in a constructed synopsis that is more refined in skewed regions, where data is highly correlated, and less refined in uniform regions, where the estimation assumptions are relatively valid.

• **Experimental Validation**. We present an experimental evaluation of our proposed framework on synthetic and real-

1 Due to space constraints, a detailed overview of related work can be found in the full version of the paper [13]

life data sets and a variety of query workloads. Our results demonstrate that concise XSKETCHes are effective synopses for estimating the selectivity of twig queries with complex XPath expressions, and offer improved estimation accuracy when compared to previous summarization schemes.

## 2. Preliminaries

**XML Data Model** Following common practice, we model an XML document as a tree $T(V, E)$, where each node $u_i \in V$ corresponds to an element, attribute or value and an edge $(u_i, u_j) \in E$ represents the containment of $u_j$ under $u_i$. Figure 1 shows a sample XML data tree that contains bibliographical data. The document contains `author` elements which point to a `name` and several `papers` and `books`. Each `paper` contains a `title`, a `year` of publication and one or more `keywords`, whereas a `book` points to its `title`. We assume that leaf elements contain values but we only show the numerical values for the year of publication in order to avoid clutter in the graph. Note that element nodes in the graph are named with the first letter of the element's tag plus a unique identifier.

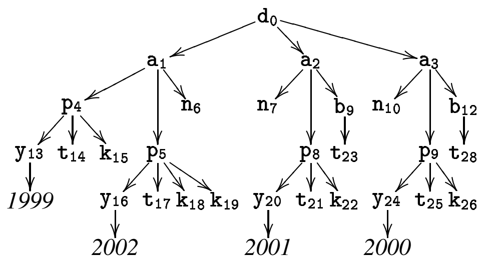
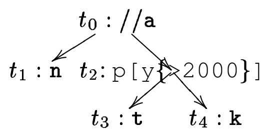


**Figure 1. Example XML Document**

**Twig Queries** We define a twig query $T_Q$ as a node-labeled tree $T_Q(V_Q, E_Q)$, where each node $t_i \in V_Q$ is labeled with a path expression $P_i$ (we will use the notation $t_i : P_i$ to refer to a twig node and its path expression together). At an abstract level, each node $t_i$ corresponds to a subset of elements, while path $P_i$ describes the structural relationship that must be satisfied between elements in $t_i$ and elements in its parent node. In our work, we focus on XPath expressions of the form $\mathtt{l}_1\{\sigma_1\}[\bar{\mathtt{l}}_1\{\bar{\sigma}_1\}]/\cdots/\mathtt{l}_n\{\sigma_n\}[\bar{\mathtt{l}}_n\{\bar{\sigma}_n\}]$, where $\mathtt{l}_i$ denotes a label, $\sigma_i$ is a value predicate that restricts element values at point $i$, and $[\bar{\mathtt{l}}_i\{\bar{\sigma}_i\}]$ is a (possibly empty) complex XPath expression that serves as a branching predicate, i.e., it requires the existence of at least one such branch at point $i$ of the expression. Figure 2(b) shows an example twig query, which corresponds to the XQuery `for` clause of Figure 2(a). The result of a twig query is a set of *binding tuples*, which describe the assignment of document elements to query variables according to the structural constraints of the corresponding path expressions. More formally, a binding tuple $\bar{t} = \{e_1, e_2, \ldots, e_m\}$ assigns element $e_i$ to query variable $t_i : P_i$ $1 \le i \le m$, so that, for any edge $(t_j, t_i) \in T_Q$, element $e_i$ is in the result set of $P_i$ when it is evaluated from $e_j$. The selectivity $s(T_Q)$ of query $T_Q$ is defined as the number of binding tuples that it generates.

```
for $t0 in //a,
  $t1 in $t0/n,
  $t2 in $t0/p[y>2000],
    $t3 in $t2/t,
    $t4 in $t2/k,
```

(a)



(b)

**Figure 2. (a) XQuery `for` clause, (b) Corresponding twig query**

**Example 2.1:** Consider the twig query of Figure 2(b). The query starts with all `author` elements ($t_0$) and then retrieves their `name` ($t_1$) and the contained `paper` elements that have a `year` child with value greater than 2000 ($t_2$). Finally, the `title` ($t_3$) and `keyword` ($t_4$) elements of the `paper` are included in the result. We can verify that the query generates three binding tuples $\bar{t} = [t_0, t_1, t_2, t_3, t_4]$ over the document of Figure 1:

| | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|---|
| $\bar{t}_1$: | a1 | n6 | p5 | t17 | k18 |
| $\bar{t}_2$: | a1 | n6 | p5 | t17 | k19 |
| $\bar{t}_3$: | a2 | n7 | p8 | t21 | k22 |

▮

Depending on the context, we will represent a twig query as a tree or as the equivalent `for` clause in the XQuery language. More specifically, let $t_0, t_1, t_2, \ldots, t_m$ be the sequence of tree nodes that are generated from a depth-first traversal of $T_Q$. Query $T_Q$ is then equivalent to the XQuery clause `for` $t_0$ `in` $P_0$, $t_1$ `in` $\mathtt{parent}(t_1)/P_1$, ..., $t_m$ `in` $\mathtt{parent}(t_m)/P_m$ , where $\mathtt{parent}(t_i)$ is the parent node of $t_i$ in $T_Q$.

## 3. Synopsis Model

### 3.1. Overview of XSKETCHes

The XSKETCH synopsis mechanism [11, 12] relies on a generic graph-summary model that captures the basic path structure of the input XML tree. Formally, given a data tree

$G = (V_G, E_G)$, a graph synopsis $\mathcal{S}(G) = (V_\mathcal{S}, E_\mathcal{S})$ is a directed node-labeled graph, where (1) each node $v \in V_\mathcal{S}$ corresponds to a subset of element (or attribute) nodes in $V_G$ (termed the *extent* of $v$ – $\mathtt{extent}(v)$) that have the *same label*, and (2) an edge in $(u, v) \in E_G$ is represented in $E_\mathcal{S}$ as an edge between the nodes whose extents contain the two endpoints $u$ and $v$. Each synopsis node $u$ stores a tag $\mathtt{tag}(u)$ for the common tag of its elements and a count field $|u|$ for the size of its extent (in what follows we use $u$ and $\mathtt{extent}(u)$ interchangeably). Figure 3(a) shows a graph synopsis for the document of Figure 1, where elements are partitioned according to their tag (synopsis nodes are named with the first letter of their tag in upper case).

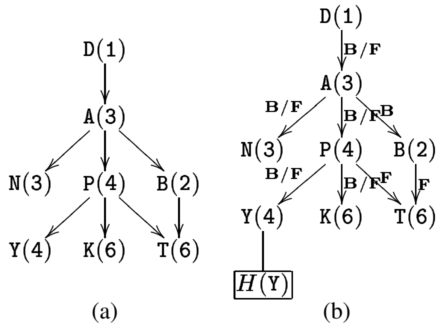


**Figure 3. (a) Graph Synopsis, (b) XSKETCH Synopsis**

XSKETCH synopses [11, 12], which target selectivity estimation for *single* XPath expressions, are specific instantiations of this generic graph-synopsis model. In order to cover key properties of the path distribution, the basic synopsis is augmented with edge-labels that capture localized *backward- and forward-stability* conditions across synopsis nodes. An edge $u \rightarrow v$ is **B**ackward (resp., **F**orward) stable, if all elements in the extent of $v$ (resp., $u$) have at least one parent (resp., child) element in the extent of $u$ (resp., $v$). As an example, Figure 3(b) shows the XSKETCH summary for the graph synopsis of Figure 3(a). We observe that edge A→P is both backward and forward stable since all `papers` have an `author` parent, and all `authors` have at least one `paper` child. As a result, $|\mathtt{P}| = 4$ is an accurate selectivity estimate for path expression `A/P`, while $|\mathtt{A}| = 3$ is an accurate estimate for `A[/P]`. In our earlier work [11], we have developed an estimation framework that uses this stability information in order to approximate the selectivity of single XPath expressions with branching predicates.

In a follow-up study [12], the XSKETCH synopsis model is extended with localized, per-node distribution information that summarizes the value-content of the document. More specifically, for each node $v$ that represents elements with values, the synopsis stores a (multidimensional) value summary $H(v)$ that approximates the joint frequency distribution of values under $v$ and/or other synopsis nodes. In our example, the `Y` node is associated with a histogram $H(\mathtt{Y})$ that contains distribution information on `year` values, potentially with respect to values along paths that contain `year` elements. Given the path expression `A/P/Y[>2000]`, the estimation framework computes an estimate by combining (1) edge stability information on the structural part `A/P/Y`, and (2) value distribution information, from $H(\mathtt{Y})$, on the value predicate [>2000]. As we have demonstrated in our earlier work, the combination of edge stabilities and value distribution information captures effectively the complex correlations that exist between and across path structure and element values in the document graph.

## 3.2. XSKETCH Synopses for Twig Queries

Our proposed summarization model for twig queries is based on the general XSKETCH paradigm, i.e., a generic graph synopsis augmented with appropriate distribution information. Compared to XSKETCHes for single path expressions, however, the new synopses need to store information at a much finer level of detail, since the selectivity of twig queries depends on more complex correlation patterns than that of single-path expressions. We illustrate this point with a simple example below.

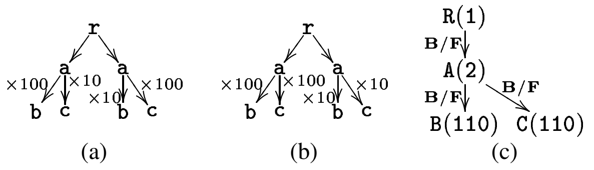


**Figure 4. (a) and (b) XML documents, (c) Structural XSKETCH.**

Consider the two example XML documents shown in Figure 4(a,b). Both documents have similar path-structure and differ only in the count of `b` and `c` children under each `a` element (child counts are shown along the corresponding edges). It is straightforward to show that any XPath expression has the same selectivity over the two documents, and thus they both map to the same zero-error XSKETCH summary of Figure 4(c) (the estimation error is zero since all edges are backward and forward stable). On the other hand, the selectivity of twig queries can vary significantly between the two documents. For instance, the twig query $T_Q$, `for` $t_0$ `in A,` $t_1$ `in` $t_0$`/B,` $t_2$ `in` $t_0$`/C` (which pairs together `b` and `c` elements with the same parent), generates 2000 binding tuples on the first document vs 10100 tuples on the second document. As our example shows, therefore, two documents that have the *same* zero-error XSKETCH summary, can have very different selectivities on twig queries. In effect, the existing *single-path* XSKETCH model does not store detailed enough information to capture the correlation patterns that affect the selectivity of twig queries with *multiple* path expressions.

The previous example indicates that an XSKETCH synopsis needs to capture distribution information at a finer level of detail in order to provide effective selectivity estimates for twig queries. Consider again the synopsis of Figure 4(c) and assume that node A records a two-dimensional distribution $f_A(b,c)$ for the fraction of elements in A that have $b$ children in B and $c$ children in C. For the document in Figure 4(a), this information would be the following: $f_A(10,100) = 0.5$, $f_A(100,10) = 0.5$. If we consider the previously mentioned query $T_Q$, we observe that any element of A in the fraction $f_A(b,c)$ will produce $b \cdot c$ binding tuples and thus the selectivity of the query will be $\sum_{b,c} |\mathtt{A}| \cdot f_A(b,c) \cdot b \cdot c$. Alternatively, we can compute the same selectivity through a distribution centered around node C. Thus, if $f_C(b)$ is the fraction of elements in C whose (sole) ancestor in A has $b$ children in B, then the selectivity is computed as $\sum_b |\mathtt{C}| \cdot f_C(b) \cdot b$. At an abstract level, both approaches compute the selectivity by capturing content information on the result tuples themselves: $f_A$ records how many result tuples have the same A element, while $f_C$ records how many result tuples have the same C element. In other words, they record information on the result of the join, rather than on the operands of the join, which are the actual elements in nodes A, B and C.

The key idea of our solution, therefore, is to record distribution information for the elements of a synopsis node with respect to outgoing paths (distribution $f_A$) or paths that emanate from ancestor elements (distribution $f_C$). In general, let $\mathtt{n}_i$ be a synopsis node, $\mathtt{n}_{d_j}$ be a child of $\mathtt{n}_i$ and $\mathtt{n}_{a_j}$ be an ancestor. An *edge distribution* $f_i(C_1, \ldots, C_m, C_{m+1}, \ldots, C_k)$ is a multi-dimensional fraction distribution over the elements of $\mathtt{n}_i$, where: (a) each dimension $C_j$, $1 \leq j \leq m$, corresponds to a synopsis edge $\mathtt{n}_i \rightarrow \mathtt{n}_{d_j}$ and is called a *forward count* (e.g., counts $b$ and $c$ of distribution $f_A$), and (b) each dimension $C_j$, $m < j \leq k$, corresponds to a synopsis edge $\mathtt{n}_{a_j} \rightarrow \mathtt{n}_{z_j}$ and is called a *backward count* (e.g., count $b$ of $f_C$). We will use $\mathtt{scope}(f_\mathtt{n})$ to denote the set of edges $\{\mathtt{n}_i \rightarrow \mathtt{n}_{d_j} | 1 \leq j \leq m\} \cup \{\mathtt{n}_{a_j} \rightarrow \mathtt{n}_{z_j} | m < j \leq k\}$. A specific point $f_i(c_1, \ldots, c_k)$ of the distribution is equal to the fraction of elements $e$ from $\mathtt{n}_i$ that satisfy the following properties: (a) for each forward count $C_j$ $(1 \leq j \leq m)$, $e$ has exactly $c_j$ children to $\mathtt{n}_{d_j}$, and (b) for each backward count $C_j$ $(m < j \leq k)$, the ancestor $e' \in \mathtt{n}_{a_j}$ of $e$ has exactly $c_j$ children to $\mathtt{n}_{z_j}$.

**Example 3.1 :** Consider node P of the synopsis in Figure 3(b) and consider the edge-distribution $f_\mathtt{P}(C_Y, C_K, C_P, C_N)$, where $C_Y$ is a forward count for $\mathtt{P} \rightarrow \mathtt{Y}$, $C_K$ is a forward count for $\mathtt{P} \rightarrow \mathtt{K}$, $C_P$ is a backward count for $\mathtt{A} \rightarrow \mathtt{P}$ and $C_N$ is a backward count for $\mathtt{A} \rightarrow \mathtt{N}$. The point $f_\mathtt{P}(c_y, c_k, c_p, c_n)$ records the fraction of elements in P that have $c_y$ children in Y, $c_k$ children in K and whose ancestor in A has $c_p$ children in P and $c_n$ children in N. The following table shows the contents of $f_\mathtt{P}(C_K, C_Y, C_P, C_N)$ and the `paper` elements that correspond to each fraction:

| $C_K$ | $C_Y$ | $C_P$ | $C_N$ | $f_\mathtt{P}$ | Elements |
|---|---|---|---|---|---|
| 2 | 1 | 2 | 1 | 0.25 | $\mathtt{p}_4$ |
| 1 | 1 | 2 | 1 | 0.25 | $\mathtt{p}_5$ |
| 1 | 1 | 1 | 1 | 0.50 | $\mathtt{p}_8, \mathtt{p}_9$ |

If we consider the twig query `for` $t_0$ `in A,` $t_1$ `in` $t_0$`/N,` $t_2$ `in` $t_0$`/P/K` (which contains paths from $\mathtt{scope}(f_\mathtt{P})$ only) then each element in the fraction $f_\mathtt{P}(c_y, c_k, c_p, c_n)$ will generate $c_k \cdot c_n$ binding tuples (these two counts correspond to the leaves of the query tree). As a result, the selectivity of the query can be calculated as $\sum_{c_k, c_y, c_p, c_n} |\mathtt{P}| \cdot f_\mathtt{P}(c_k, c_y, c_p, c_n) \cdot c_k \cdot c_n$. ∎

Intuitively, an edge-distribution $f_i$ enables selectivity estimates for twig queries that contain elements of $\mathtt{n}_i$ in their binding tuples and traverse the edges covered by the forward and backward counts. Given a synopsis node $\mathtt{n}_i$, however, different subsets of its elements have different sets of outgoing paths and are associated with different paths from ancestor nodes. Keeping separate distribution information is clearly impractical since there are exponentially many combinations and the problem simply becomes unmanageable. As a result, we limit the focus to forward and backward counts along paths that provably exist for all elements in a synopsis node. This set of common paths is captured naturally by the *twig stable neighborhood* $\mathtt{TSN}(\mathtt{n}_i)$, which is defined as the set of all nodes in the synopsis that either (a) reach $\mathtt{n}_i$ through a B-stable path (including $\mathtt{n}_i$ itself), or (b) are reached from any of the nodes found in (a) through an F-stable path of length 1. It is straightforward to show that each element in $\mathtt{n}_i$ is contained in a document twig that covers elements from all nodes in $\mathtt{TSN}(\mathtt{n}_i)$, and we will henceforth limit distribution $f_i$ to edge counts between nodes in $\mathtt{TSN}(\mathtt{n}_i)$. We note that the twig stable neighborhood is a subset of the *stable neighborhood* of a node, which serves a similar functionality, with respect to element values, within the single-path XSKETCH model [12].

Using this model, the most complete distribution information amounts to keeping, for each synopsis node $\mathtt{n}_i$, an edge distribution that covers all possible backward and forward counts within the corresponding twig stable neighborhood. In practice, however, we limit the amount of information recorded in the synopsis in two ways: (a) only a subset of paths in $\mathtt{TSN}(\mathtt{n}_i)$ is accounted for in the corresponding edge-distribution, and (b) the synopsis contains compressed distribution information in the form of an *edge-histogram* $H_i(C_1, \ldots, C_k)$, which is an approximation of the edge distribution $f_i(C_1, \ldots, C_k)$. We note that an edge-distribution can be summarized very efficiently using multidimensional methods such as histograms and wavelets, since it is essentially defined over a space of integer edge counts. Our twig synopsis model can thus be defined as follows:

**Definition 3.1** *A* Twig XSKETCH *is a graph summary that records (a) edge stabilities and, (b) a multidimensional edge-histogram $H_i(C_1, \ldots, C_k)$ for each node $\mathtt{n}_i$, where counts $C_1, \ldots, C_k$ correspond to a set of edges* $\mathtt{scope}(\mathtt{n}_i)$ *that are contained entirely in* $\mathtt{TSN}(\mathtt{n}_i)$.

As noted, histogram $H_i$ covers only a subset of all possible paths in $\mathtt{TSN}(\mathtt{n}_i)$. If a twig query references a path that is not included in $H_i$, then our estimation framework makes an independence assumption for the missing path under the premise that it does not exhibit high correlation with the distribution of structural join cardinalities at $\mathtt{n}_i$. Based on this, the construction algorithm includes in $H_i$ the most highly correlated path counts in order to increase the validity of the estimation assumption and thus the accuracy of estimates. Of course, as more space is assigned to the synopsis, more dimensions can be included in order to improve the accuracy of the approximation.

Up to this point, we have focused solely on the structural part of the problem, ignoring the effect of element values on the distribution of structural join cardinality. It is straightforward to extend our definitions to cover value distributions and thus capture the possible correlations that exist between path structure and element values. More specifically, we introduce extended multi-dimensional value histograms $H_i^v(V_1, \ldots, V_l, C_1, \ldots, C_k)$, which approximate the joint distribution of elements in $\mathtt{n}_i$ with respect to values (dimensions $V_1, \ldots, V_l$) and edge counts (dimensions $C_1, \ldots, C_k$) that are contained entirely within $\mathtt{TSN}(\mathtt{n}_i)$. Due to lack of space, we do not discuss these summaries further; a detailed presentation can be found in the full version of this paper [13].

### 3.3. Discussion

Our extended XSKETCH model is based on a new type of distribution information that enables selectivity estimates for the cardinality of structural joins. The key novelty of our approach is that it approximates the cardinality of a join through the joint distribution of *result counts*, rather than through the distribution of the input join values. As a simplified example, consider the problem of estimating the cardinality of $R \bowtie S$ between two relations $R(a)$ and $S(b)$. Traditional methods build summaries $H(a)$ and $H(b)$ for the value distributions of $R.a$ and $S.a$ and then estimate the cardinality by multiplying approximate frequencies of joined values. Our technique, on the other hand, builds a single summary $H_R(c)$ for the fraction of tuples in $R$ that join with $c$ tuples in $S$ and estimates the result cardinality as $\sum_c |R| \cdot H_R(c) \cdot c$. Estimation, therefore, is decoupled from the actual value distributions and is based on a set of counts that capture aggregate information on the joined tuples (in this example, the number of $S$ tuples that join with a single $R$ tuple). The key benefit is that we can use existing summarization techniques, such as histograms or wavelets, to summarize very effectively the resulting distribution of integer counts and thus provide accurate selectivity estimates for the cardinality of the join, even if the value distribution of the join attributes cannot be approximated reasonably well with such techniques (e.g., attributes with categorical values).

## 4. Estimation

In this section, we present an estimation framework for approximating the selectivity of twig queries over concise Twig XSKETCH synopses. Our framework is based on the concept of a maximal twig query, which represents an expanded match of a twig query over a specific synopsis. A twig query $T_Q$ is called *maximal* if for every twig node $t_i : P_i \in T_Q$, path $P_i$ is a single-step expression of the form $\mathtt{l}_i\{\sigma_i\}[\bar{\mathtt{l}}_i]$, i.e., it consists of a single navigational step across tag $\mathtt{l}_i$ with a value predicate $\{\sigma_i\}$ and a branching predicate $[\bar{\mathtt{l}}_i]$ (either of which might be empty). Given an arbitrary twig query $T_Q$, our framework generates the set of maximal forms as follows: the '//' operator is expanded with a valid document path (using the structural information contained in the synopsis), and each node $t_i : \mathtt{l}_1[P_1]/\ldots/\mathtt{l}_m[P_m]$ is substituted with the chain of twig nodes $t_{i.1} : \mathtt{l}_1[P_1] \rightarrow t_{i.2} : \mathtt{l}_2[P_2] \rightarrow \ldots \rightarrow t_i : \mathtt{l}_m[P_m]$. As an example, if we consider the synopsis of Figure 3(b), then the twig query of Figure 5(a) will be mapped to the set of maximal twig queries shown in Figure 5(b). It is straightforward to show that the selectivity of a twig query on tree-structured data is equal to the sum of selectivities of its maximal twig queries. As a result, for the remainder of this paper, we focus on selectivity estimation for maximal twig queries only.

An embedding $T_e$ of query $T_Q$ represents a match of the query over specific synopsis paths. We compute an embedding of $T_Q$ by substituting each path $\mathtt{l}_i\{\sigma_i\}[\bar{\mathtt{l}}_i]$ with a synopsis path $\mathtt{n}_i[\mathtt{B}_i]$, where $\mathtt{tag}(\mathtt{n}_i) = \mathtt{l}_i$ and $\mathtt{B}_i$ matches path expression $\bar{\mathtt{l}}_i$. A twig embedding essentially describes the evaluation of the twig query on the elements of specific synopsis nodes and in effect represents a subset of the bindings of the query. The selectivity, therefore, of a twig query $T_Q$ can be estimated as the sum of selectivities for all of its unique twig embeddings and the estimation problem is further reduced to estimating the selectivity of a single twig

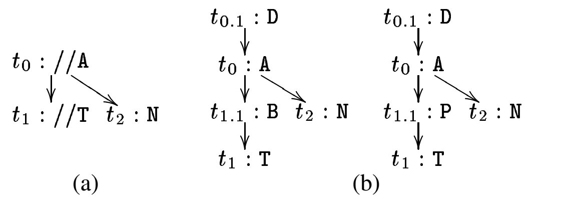
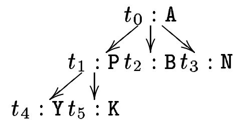


**Figure 5. (a) Twig Query, (b) Maximal Twig Queries**

embedding.

We now proceed to describe our estimation framework for maximal twig embeddings. Due to space constraints, we will focus on embeddings that contain simple paths only; the extension of our framework to embeddings with branching and value predicates can be found in the full paper [13]. We will base our presentation on the example twig embedding $T$ shown in Figure 6, which is defined over the synopsis of Figure 3(b). Figure 6(b) shows the two edge histograms that are recorded in our synopsis for the specific example (for simplicity, $x$ denotes the path count for an edge that points to synopsis node X). As shown, histogram $H_A(P,N)$ contains two forward counts $P$ and $N$, corresponding to edges $\texttt{A} \rightarrow \texttt{P}$ and $\texttt{A} \rightarrow \texttt{N}$, while histogram $H_P(K,Y,P)$ records two forward counts $K$ and $Y$, for edges $\texttt{P} \rightarrow \texttt{K}$ and $\texttt{P} \rightarrow \texttt{K}$, and one backward count $P$ for edge $\texttt{A} \rightarrow \texttt{P}$. Note that count $P$ corresponds to the same edge in both histograms, but it represents a forward count in $H_A$ and a backward count in $H_P$. In general, let $f_{\texttt{X}}(y_1,\ldots,y_k)$ be an edge distribution with forward counts $y_1,\ldots,y_k$ to nodes $\texttt{Y}_1,\ldots,\texttt{Y}_k$. We will use $\sum F_{\texttt{X}}(y_1,\ldots,y_k) = \sum f_{\texttt{X}}(y_1,\ldots,y_k)\cdot\prod y_i$ to denote the average number of binding tuples, for each element of X, along edges $\texttt{X} \rightarrow \texttt{Y}_1,\ldots,\texttt{X} \rightarrow \texttt{Y}_k$. We extend this notation to $F_X(y_1,\ldots,y_k \mid w_1,\ldots,w_l) = f_X(y_1,\ldots,y_k \mid w_1,\ldots,w_l)\cdot\prod y_i$, when the probability distribution $f_X$ is conditioned on backward counts $w_1,\ldots,w_l$. As we will discuss later, such a conditional distribution essentially correlates edge distribution information between a node of a twig embedding and its ancestor nodes.

Based on the definition of twig queries, the selectivity of $T$ is equal to the cardinality of the set of binding tuples $B = \{(e_a,e_p,e_b,e_n,e_k,e_n) \mid e_a \rightarrow e_p, e_a \rightarrow e_b, e_a \rightarrow e_n, e_p \rightarrow e_k, e_p \rightarrow e_k\}$, where $e_x$ denotes an element of synopsis node X and $e_x \rightarrow e_y$ a document edge. Our approach for estimating $s(T)$ is based on the following conceptual method for computing set $B$: at step (1), form the set $B_1 = \{e_a \mid e_a \in A\}$; at step (2), iterate over each element of $B_1$ and compute set $B_2 = \{(e_a,e_p,e_b,e_n) \mid e_a \in B_1, e_a \rightarrow e_p, e_a \rightarrow e_b, e_a \rightarrow e_n\}$; finally, at step (3), iterate over each 4-tuple of $B_2$ and compute the set $B_3 = \{(e_a,e_p,e_b,e_n,e_k,e_y) \mid (e_a,e_p,e_b,e_n) \in B_2, e_p \rightarrow e_k, e_p \rightarrow e_y\}$. Obviously, $B_3 = B$ and the selectivity of the embedding is equal to $s(T) = |B_3|$. Our estimation methodology essentially models the generation of sets $B_i$ and computes an estimate for $s(T)$ by approximating the cardinality of the final set. More specifically, the proposed technique works as follows. We observe that set $B_1$, which corresponds to the first step of the process, will contain exactly $|\texttt{A}|$ elements. Based on our discussion on edge-distributions, each element $e \in B_1 = \texttt{A}$ will generate $\sum F_A(p,b,n)$ binding tuples to nodes P, B, N, and the size of set $B_2$ can thus be computed as follows:



(a)

| $p$ | $n$ | $H_A$ |
|---|---|---|
| 2 | 1 | 1/3 |
| 1 | 1 | 2/3 |

| $k$ | $y$ | $p$ | $H_P$ |
|---|---|---|---|
| 2 | 1 | 2 | 1/4 |
| 1 | 1 | 2 | 1/4 |
| 1 | 1 | 1 | 2/4 |

(b)

**Figure 6. (a) Twig Embedding, (b) Edge Histograms**

$$|B_2| = |B_1|\cdot\sum_{p,b,n} F_A(p,b,n) = |A|\cdot\sum_{p,b,n} f_A(p,b,n)\cdot p\cdot b\cdot n$$

Let us consider a 4-tuple $b_2 = (e_a,e_p,e_b,e_n)$ in $B_2$. Note that $b_2$ corresponds to a count assignment $(p,b,n)$ which is interpreted as follows: $e_p$ is one of the $p$ children of $e_a$ in P, $e_b$ is one of the $b$ children of $e_a$ in B, and $e_n$ is one of the $n$ children of $e_a$ in N. At step (3) of the evaluation algorithm, $b_2$ will be combined with every possible pair $(e_k,e_y)$ of $e_p$'s children to nodes K and Y, resulting in a set of 6-tuples. We observe that the distribution of $k$ and $y$ edge counts in the 4-tuples of $B_2$ is described by $f_P(k,y \mid p,b,n)$, and we can thus compute the average number of 6-tuples per 4-tuple as $\sum_{k,y} F_P(k,y \mid p,b,n)$. This observation leads to the following expression for the cardinality of set $B_3$ (and thus the selectivity of the embedding):

$$s(T) = |B_3| = |A|\cdot\sum_{p,b,n}\left(F_A(p,b,n)\cdot\sum_{k,y} F_P(k,y \mid p,b,n)\right)$$

Overall, the final expression will compute the selectivity of $T$ with zero error if the synopsis records full information on distributions $f_A(p,b,n)$ and $f_P(k,y \mid p,b,n)$. In our example, however, the synopsis only records histograms $H_A(p,n)$ and $H_P(k,y,p)$, which approximate the distributions $f_A$ and $f_P$ on *subsets* of the needed edge counts. To compensate for the lack of detailed distribution information, our framework makes specific statistical assumptions that enable the approximation of the needed terms with the information recorded in the synopsis. Returning to our example, we initially apply the following independence assumption:

**Forward Independence Assumption.** Let $\mathtt{n}_i$ be a synopsis node with edge histogram $H_i$. The distribution of elements in $\mathtt{n}_i$ with respect to forward counts that are not covered in $\mathtt{scope}(H_i)$ is independent of other edge counts. More specifically, if $C$ are the counts covered in $\mathtt{scope}(H_i)$, $U = \{U_j\}$ are the forward counts for edges $\mathtt{n}_i \rightarrow \mathtt{n}_j$ that are not included in $\mathtt{scope}(H_i)$, and $C'$ is a set of edge counts. then the following holds:

$$F_i(U \cup C \mid C') \approx F_i(C \mid C') \cdot \prod_{U_j \in U} F_i(U_j)$$

Using Forward Independence, we can approximate term $F_A$ as $F_A(p, b, n) \approx F_A(p, n) \cdot F_A(b)$. In order to handle the needed $F_P$ term, our framework applies the following independence assumption:

**Correlation Scope Independence Assumption.** Let $\mathtt{n}_i$ be a synopsis node with edge histogram $H_i$. The distribution of elements in $\mathtt{n}_i$ is independent of edge counts that are not covered in $\mathtt{scope}(H_i)$. More specifically, if $C$ is the set of edge counts that corresponds to $\mathtt{scope}(H_i)$ and $C'$ is a set of arbitrary edge counts, then the following holds:

$$F_i(C - C' \mid C') \approx F_i(C - C' \mid C \cap C') \approx \frac{H_i((C - C') \cup (C \cap C'))}{H_i(C \cap C')}$$

Note that the terms of the fraction in the second independence assumption are appropriate marginals on histogram $H_i$. Returning to our example, the application of Correlation Independence yields the approximation $F_P(k, y \mid p, b, n) \approx F_P(k, y \mid p) \approx H_P(k, y, p)/H_P(p)$. Overall, the estimation expression is rewritten as follows:

$$s(T) \approx |\mathtt{A}| \cdot \sum_b F_A(b) \cdot \sum_{k,y,p,n} F_A(p, n) \cdot F_P(k, y \mid p)$$

We observe that the summation term involving term $F_A(b)$ essentially computes the average number of children in $B$ per element in $A$. Since the synopsis does not record detailed distribution information on $F_A(b)$, we approximate the needed selectivity with the following uniformity assumption:

**Forward Uniformity Assumption.**Let $\mathtt{n}_i$ a synopsis node, $\mathtt{n}_j$ be a child node that is not covered by any forward count (i.e., $\mathtt{n}_i \rightarrow \mathtt{n}_j \notin \mathtt{scope}(H_i)$), and $|\mathtt{n}_i \rightarrow \mathtt{n}_j|$ be the estimated number of elements in $\mathtt{n}_j$ that have a parent in $\mathtt{n}_i$. On the average, each element in $\mathtt{n}_i$ reaches the same number of elements in $\mathtt{n}_j$, which can be approximated as follows:

$$\sum_{c_j} F_{\mathtt{n}}(c_j) = \frac{|\mathtt{n}_i \rightarrow \mathtt{n}_j|}{|\mathtt{n}_i|}$$

```
procedure TREEPARSE($T_Q$)
Input: A twig embedding $T_Q$ with nodes $\{t_1, \ldots, t_m\}$
Output:   Count sets $E_i$, $U_i$, $D_i$ for each $t_i \in T_Q$
begin
1.  $covered := \emptyset$
2.  for each twig node $(t_i : \mathtt{n}_i)$ in depth-first order do
3.      if ($|\mathtt{children}(t_i)| == 0$) continue // Skip leaf nodes
4.      Let $H_i$ be the edge-histogram at $\mathtt{n}_i$
5.      $U_i := \{$forward counts not covered in $\mathtt{scope}(H_i)\}$
6.      $D_i := \mathtt{scope}(H_i) \cap covered$
7.      $E_i := \mathtt{scope}(H_i) - covered$
8.      $covered := covered \cup E_i$
9.  done
10. return $< (E_i, U_i, D_i) >$
end
```

**Figure 7. Algorithm TREEPARSE**

The application of this last assumption yields the approximation $\sum_b F_A(b) \approx |\mathtt{A} \rightarrow \mathtt{B}|/|\mathtt{A}|$ and results in the final form of the expression:

$$\begin{aligned} s(T) &= |\mathtt{A}| \cdot \frac{|\mathtt{A} \rightarrow \mathtt{B}|}{|\mathtt{A}|} \cdot \sum_{k,y,p,n} F_A(p, n) \cdot F_P(k, y \mid p) \\ &= |\mathtt{A} \rightarrow \mathtt{B}| \cdot \sum_{k,y,p,n} F_A(p, n) \cdot F_P(k, y \mid p) \end{aligned}$$

Note that $|\mathtt{A} \rightarrow \mathtt{B}|$ is estimated using the single-path XSKETCH estimation framework and is equal to $|\mathtt{B}|$ if the edge is backward stable. Overall, the end result is quite intuitive. Term $|\mathtt{A} \rightarrow \mathtt{B}|$ is the number of binding tuples for the sub-embedding that contains nodes A and B, while the summation term represents the average number of binding tuples, per element in A, for the sub-embedding that does not contain node B. By multiplying the two, we compute the total number of bindings for $T$, the union of the two sub-embeddings. According to the contents of histograms $H_A$ and $H_P$ (Figure 6(b)), there are two value combinations for $n, p$ and two value combinations for $y, k$ which results in the expansion of the sum to $2 \times 2 = 4$ terms:

$$\begin{aligned} s(T_e) = 2 \times (&H_A(2,1) \cdot H_P(2,1 \mid 2) \cdot 2 \cdot 1 \cdot 2 \cdot 1 + \\ &H_A(2,1) \cdot H_P(1,1 \mid 2) \cdot 1 \cdot 1 \cdot 1 \cdot 1 + \\ &H_A(1,1) \cdot H_P(2,1 \mid 1) \cdot 1 \cdot 1 \cdot 2 \cdot 1 + \\ &H_A(1,1) \cdot H_P(1,1 \mid 1) \cdot 1 \cdot 1 \cdot 1 \cdot 1) \quad = 10/3 \end{aligned}$$

Overall, the selectivity expression is based on a top-down traversal of the embedding that combines and correlates information from different edge distribution histograms. In the general case, let $T$ be a twig embedding, where each node $t_i$ is labeled with the corresponding synopsis node $\mathtt{n}_i$. Figure 7 shows Algorithm TREEPARSE for processing an arbitrary twig embedding and producing the components of the selectivity expression. The algorithm performs a depth-first traversal of the embedding and uses histogram $H_i$ of the current node $t_i$ in order to determine an expansion set $E_i$, an uncovered set $U_i$, and a correlation set $D_i$. Set $E_i \subset \mathtt{scope}(\mathtt{n}_i)$ contains forward counts from $H_i$ that compute the distribution of binding tuples to child nodes of $t_i$; set $U_i$ contains forward counts to children of $\mathtt{n}_i$ that are not covered in $H_i$; and set $D_i \subset (E_0 \cup \cdots \cup E_{i-1})$ contains backward counts from $H_i$ that correlate the expansion from $t_i$ to the previous traversal steps ($D_0 = \emptyset$). Given this information, the final selectivity expression is written as follows:

$$s(T_Q) \quad = \quad |\mathtt{n}_0| \cdot \left( \prod_{0 \le i \le m} \prod_{C \in U_i} \sum_{C} F_i(C) \right) \cdot \sum_{E_1,\ldots,E_m} F_0(E_0 \mid D_0) \ldots F_m(E_m \mid D_m)$$

where terms $\sum F_i(C)$ are computed using the Forward Uniformity assumption, while terms $F_i(E_i \mid D_i)$ are computed using the information available in histogram $H_i$.

As we mentioned at the beginning of this section, our estimation framework extends naturally to the general case of twig embeddings with branching and value predicates. In the interest of space, we defer the complete details to the full version of this paper [13].

## 5. Construction

We now turn our attention to the difficult problem of constructing an effective Twig XSKETCH for a given space budget. The build process of an XSKETCH synopsis can be abstracted to three interrelated steps: (a) determine a partitioning of document elements in synopsis nodes, (b) determine the dimensions of value and edge histograms at each synopsis node, and (c) allocate buckets to the histograms of step (b). The goal, of course, is to perform these three steps so that the estimation error is minimized and the storage budget is not exceeded. As we have shown in our earlier work, however, constructing an optimal (in terms of estimation error) XSKETCH for a given space budget is $\mathcal{NP}$-hard even for the case of single-path structural XSKETCHes where steps (b) and (c) are not relevant. Based on this hardness result, we describe a heuristic construction algorithm that is effective in building accurate XSKETCH synopses.

Our approach is based on an adaptation of the construction algorithm for single-path XSKETCH synopses. More specifically, we define a set of *refinement operations* that perform localized transformations on the synopsis and result in a larger summary that fits better the assumptions of the estimation framework in the transformed region. The refinement operations used by the algorithm can be categorized into three types:

- *Structural Refinements*: This group includes two operations, namely `b-stabilize` and `f-stabilize`, which refine the path structure of the XSKETCH. In short, both refinements split a node into two new nodes, in order to create an additional backward or forward stable edge in the synopsis.

- *Value Refinements*: This group includes two operations: `value-refine`, which allocates more memory to a value distribution summary, and `value-expand`, which insets an additional dimension to a value histogram in order to capture the correlation to another value distribution.

- *Edge Refinements*: This group includes two refinements: `edge-refine`, which allocates more storage to a specific edge histogram, and `edge-expand` which augments the scope of a histogram with an additional edge and thus lifts the need for an independence assumption across the inserted edge.

Structural and value refinements are essentially similar to the corresponding operations defined in the single-path XSKETCH framework, while edge refinements are unique to Twig XSKETCHes. In the interest of space, we do not discuss further the specifics of these operations. The complete details, including the pseudo-code for each operation, can be found in the full version of this paper [13]

```
procedure XBUILD($G$, $B$)
Input: A document graph $G$; space budget $B$.
Output: An Twig XSKETCH $\mathcal{XS}$ of size at most $B$.
begin
1.  $\mathcal{XS}$:=$\mathcal{S}_0(G)$ // Coarse synopsis
2.  while (size($\mathcal{XS}$) $\le B$) do
3.     $R$ := gen_refinements($\mathcal{XS}$,$B$)
4.     Generate a sample workload $W$
5.     foreach $r \in R$ do
6.        Let $\mathcal{XS}_r$ be the synopsis after applying $r$
7.        $q_r$:=error($\mathcal{XS}_r$,$W$); $s_r$:=size($\mathcal{XS}_r$)
8.     done
9.     $q$:=error($\mathcal{XS}$,$W$); $s$:=size($\mathcal{XS}$)
10.    Apply $r$ that maximizes $\frac{q-q_r}{s_r-s}$
11. done
12. return $\mathcal{XS}$
end
```

**Figure 8. Algorithm XBUILD.**

The construction algorithm, termed XBUILD, starts from a coarse synopsis and, using the refinement operations, adds incrementally more complexity until the available space-budget is exhausted. The pseudo-code for XBUILD is shown in Figure 8. The initial synopsis $\mathcal{S}_0(G)$ partitions document elements into nodes based solely on their tag, and includes single-dimensional edge-histograms that cover path counts to forward-stable children only. At each step of the build process, the algorithm generates a set of candidate refinements on a sample of synopsis nodes (line 3), where the probability of sampling a node is proportional to its element count and the number of incoming or outgoing unstable edges. To select a refinement from the pool of candidates, the algorithm uses a *marginal gains* criterion: it scores each refinement based on the accuracy of the resulting summary, which is measured as the relative estimation error against a carefully chosen workload $W$, and it applies the refinement that results in the largest increase of accuracy per unit of extra space required. Workload $W$ is generated by sampling twig embeddings around the regions transformed by the candidate operations, so that the estimation error on $W$ essentially reflects the effects of each refinement. The true selectivities of queries in $W$, which are needed for the estimation error, are approximated with low error through a large enough *reference summary*, thus avoiding costly accesses to the database. In the interest of space, we do not discuss further the details of the XBUILD algorithm; a complete description can be found in the full paper [13].

## 6. Experimental Study

In this Section, we present the results from an experimental study that we have conducted with our novel synopses. We have evaluated the efficiency of our summarization methods on synthetic and real-life XML datasets with different workloads. The results show that our synopses can be effective in providing accurate selectivity estimates for twig queries with complex path expressions and demonstrate the benefits of our approach over previously proposed techniques.

### 6.1. Methodology

**Techniques**. We have performed experiments with two summarization techniques.

XSketches. We have implemented a prototype of the XSKETCH framework that we propose in this paper. Our prototype can handle twig queries that contain complex path expressions with branching predicates and value predicates. The implementation is constrained in two ways: first, it uses multi-dimensional edge histograms that capture the distribution of edges from a node to its forward-stable children only and do not include any backward counts; second, value-histograms are single-dimensional and only cover the distribution of values under a specific synopsis node. As a result, our framework needs to apply independence assumptions at each step of the estimation methodology. We have found that, although the underlying assumptions are not valid in general, they gives satisfactory results for the datasets that we have tested. We will be extending our prototype to add support for backward counts to ancestor nodes and multi-dimensional value-histograms.

Correlated Suffix Trees. Correlated Suffix Trees (CSTs) have been proposed by Chen et al. [3] for estimating the selectivity of twig queries over tree-structured data. Recall that CSTs support twig queries with simple path expressions and suffix value predicates on string values, while our technique supports twig queries with branching path expressions and range predicates on integer values. Due to this difference, we compare the two techniques on a workload of twig queries with simple path expressions and no value predicates. Accordingly, we have modified the CST construction algorithm to ignore element values and build a trie on the path structure of the document only. Among the estimation techniques proposed by the authors, we have used P-MOSH since it produced the most accurate results.

| | XMark | IMDB | SProt |
|---|---|---|---|
| Element Count | 103,136 | 102,755 | 69,599 |
| Text Size (MB) | 5.40 | 2.90 | 4.50 |
| Coarsest Synopsis (KB) | 12.20 | 8.10 | 9.70 |

**Table 1. Data Sets**

**Data Sets**. We have used three XML data sets in our experiments: *XMark*, a synthetic data set that models an auction-site, *IMDB*, a real-life data set with movie data, and *SwissProt*, a real-life data set that contains annotations on proteins. The key characteristics of the three data sets are summarized in Table 1. For each data set, we report its text size, which is the size of the corresponding disk file, and the size of the coarsest XSKETCH synopsis.

| | XMark | | IMDB | | SProt |
|---|---|---|---|---|---|
| | P | P+V | P | P+V | P |
| Avg. Result | 2,436 | 1,423 | 3,477 | 961 | 24,034 |
| Avg. Fanout | 1.99 | 1.60 | 1.66 | 1.53 | 1.97 |

**Table 2. Workload Characteristics**

**Workload.** We evaluate the generated summaries against a workload of "positive" twig queries, i.e., queries with non-zero selectivity. Table 2 summarizes the characteristics of the workloads used in terms of the average cardinality and the average fanout of internal twig nodes. Unless otherwise noted, each workload contains 1000 queries and the total number of twig nodes per query is distributed uniformly between 4 and 8. Depending on the experiment, we either use a P (Path) workload, where twig queries do not contain value predicates, or a P+V (Path+Value) workload, where 500 of queries contain one or two value predicates that cover a random 10% range of the corresponding value domain. We have also experimented with "negative" workloads (selectivity equal to zero) and we have found that our synopses consistently give close to zero estimates for this type of queries.

**Evaluation Metric**. We evaluate the accuracy of a summary by measuring the average absolute relative error with respect to the workload of queries. Given a twig query $T_Q$ that has $c$ binding tuples, the absolute relative error of an estimate $r$ is defined as $|r - c|/max(s, c)$, where $s$ is a sanity bound. We use $s$ in order to avoid the artificially high percentages of low count twig queries, and also in order to define the metric for negative queries, where $c = 0$. In our experiments, we set $s$ to the 10-th percentile of true query counts, i.e., 90% of the queries have a true count greater than $s$.

## 6.2. Results

**Twig Queries with Complex Paths** In this experiment, we evaluate the performance of our proposed synopses against a workload of twig queries that contain path expressions with branching predicates (P workload).

Figure 9(a) shows the error for the XMark and IMDB data sets. In all the plots that we show, the point at the lowest storage corresponds to the label split graph, the coarsest summary in our model. The results indicate that XSKETCHes constitute effective synopses for evaluating the selectivity of twig queries with complex branching path expressions. In the IMDB data set, we observe that the coarsest summary starts with a high estimation error of 124% which is reduced to 20% as more space is allocated to the synopsis. More notably, this low estimation error comes for a modest space budget of 50KB that represents a small fraction (1%) of the space required for the compact representation of the IMDB document (Table 1). The XMark data set exhibits a low estimation error for all storage sizes (even for the coarsest summary), since it is generated from uniform distributions and is thus more regular in structure than IMDB.

In the IMDB data set, we observe that the estimation error is reduced faster during the first stages of the build process and follows a more gradual improvement afterward. This indicates that the construction algorithm refines the summary with respect to the most important correlations first and then allocates the remaining space to regions that correspond to less dominant dependencies. In the XMark data set, on the other hand, the error remains constantly low due to the regular structure of the document. The minor fluctuations that we observe are attributed to the sampling methods of the construction algorithm, which might lead to the application of a sub-optimal refinement.

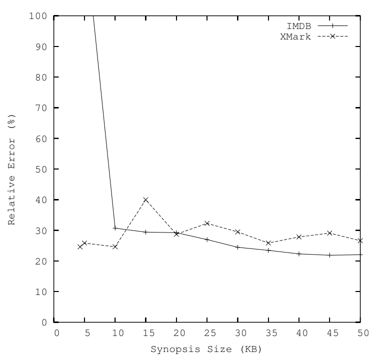
We have also performed a limited set of experiments that compare the performance of Twig XSKETCHes against Structural XSKETCHes [11] on workloads of *single* XPath expressions. Our results have shown that Twig XSKETCHes compute low-error estimates of path selectivities, but, as expected, Structural XSKETCHes enable more accurate approximations since they target specifically the problem of selectivity estimation for single paths.

**Twig Queries with Value Predicates** In this experiment, we evaluate the performance of our XSKETCH synopses against a workload of twig queries that contain path expressions with branching and value predicates (P+V workload).

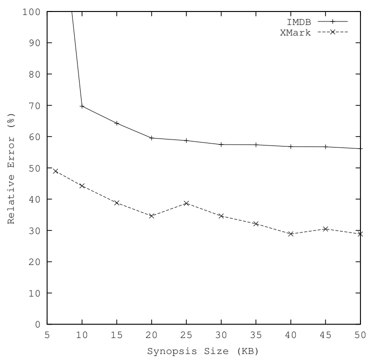
Figure 9(b) shows the estimation error of XSKETCHes for the XMark and IMDB data sets. We observe that the high estimation error of the coarsest summary is significantly reduced for larger synopsis sizes, but we notice an increase in the overall error compared to the previous experiment, where the workload contains branching predicates only. Recall, however, that the estimation problem in this context is equivalent to estimating the result size of tree-formed relational joins (twig queries with simple paths) coupled with selection predicates (value predicates) and semi-joins (branching predicates). This is a hard problem and it is not uncommon to observe estimation errors of even a 100% [7]. The increased error is also attributed to the limitations of our prototype, which uses single-dimensional histograms on values and edge-histograms with no backward counts. Still, the overall results show a significant decrease in error and they are a strong indication that XSKETCHes can be effective in estimating the selectivity of twig queries with branching and value predicates.

**Twig Queries with Simple Paths** In this experiment, we compare our XSKETCH synopses against the Correlated Suffix Trees of Chen et. al [3]. For each data set, we use a workload of 500 twig queries with simple path expressions and we compare the performance of the two techniques using the ratio $\mathtt{err}_{CST}/\mathtt{err}_X$, where $\mathtt{err}_{CST}$ and $\mathtt{err}_X$ is the average relative estimation error for CSTs and XSKETCHes respectively. We note that CSTs exhibited extremely large estimation errors (more than 1000%) on certain queries of the initial workloads for IMDB and XMark. We have decided, however, to exclude these "outliers" from the results that we present next, in order to keep the error ratio within reasonable bounds.

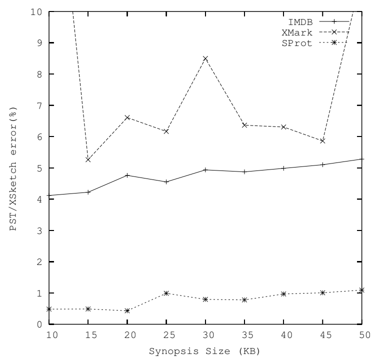
Figure 9(c) shows the ratio of CST error vs. XSKETCH error for the three data sets. Overall, we observe that XSKETCHes perform better than CSTs and yield estimates with significantly lower estimation error. More specifically, for 50KB of storage space, the two techniques seem to be equally accurate on the more regular SwissProt dataset (both have an error of 14% ), but the estimation error of CSTs is significantly worse for the other two data sets: 44% for IMDB (vs. 8% for XSKETCHes), and 26% for XMark (vs. 3% for XSKETCHes). The results also indicate that XSKETCHes make better use of the alloted space budget and manage to reduce the error faster, as evidenced by the increasing trend for the error ratio. This is due to the effec-



(a)



(b)



(c)

**Figure 9. (a) Branching Predicates: IMDB and XMark, (b) Branching and Value Predicates: IMDB and XMark, (c) Simple Paths: CSTs vs. XSKETCHes**

tiveness of the XSKETCH construction algorithm, which takes directly into account the assumptions of the estimation framework and thus allocates more space to the more correlated regions of the summary. CST construction, on the other hand, is based on the greedy pruning of low-frequency nodes and is not flexible in handling skewed regions of the data graph.

## 7. Conclusions

In this paper, we have proposed novel summarization and estimation methods for twig queries with complex XPath expressions. Our synopsis model is based on the XSKETCH model, augmented with a new type of distribution information that captures aggregate information on the cardinality of structural joins and thus enables selectivity estimates for twig queries. We have developed a systematic framework that matches a twig query over a concise XSKETCH and uses appropriate statistical assumptions in order to combine information from matching parts and compute a selectivity estimate. In order to construct an accurate XSKETCH synopsis for a given space budget (which is an $\mathcal{NP}$-hard optimization problem), we have described an efficient algorithm that starts with a coarse summary and gradually refines it by applying a set of transformation operations that we have introduced. Experimental results on synthetic and real-life data sets have verified the effectiveness of our approach and have demonstrated its benefits over previously proposed schemes.

## References

[1] A. Aboulnaga, A. R. Alameldeen, and J. F. Naughton. "Estimating the Selectivity of XML Path Expressions for Internet Scale Applications". In *Proc. VLDB*, 2001.

[2] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. "XQuery 1.0: An XML Query Language". W3C Working Draft 02, 2003.

[3] Z. Chen, H. V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. Ng, and D. Srivastava. "Counting Twig Matches in a Tree". In *Proc. ICDE*, 2001.

[4] J. Clark. "XSL Transformations (XSLT), Version 1.0". W3C Recommendation, 1999.

[5] A. Deshpande, M. Garofalakis, and R. Rastogi. "Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data". In *Proc. SIGMOD*, 2001.

[6] J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, and J. Siméon. "StatiX: Making XML Count". In *Proc. SIGMOD* , 2002.

[7] L. Getoor, B. Taskar, and D. Koller. "Selectivity Estimation using Probabilistic Models". In *Proc. SIGMOD* , 2001.

[8] L. Lim, M. Wang, S. Padmanabhan, J. Vitter, and R. Parr. XPathLearner: An On-Line Self-Tuning Markov Histogram for XML Path Selectivity Estimation. In *Proc. VLDB*, 2002.

[9] R. J. Lipton, J. F. Naughton, D. A. Schneider, and S. Seshadri. "Efficient sampling strategies for relational database operations". *Theoretical Comput. Sci.*, 116, 1993.

[10] J. F. Naughton, D. J. DeWitt, and D. M. et al. "The Niagara Internet query system". *IEEE Data Engineering Bulletin*, 24(2), 2001.

[11] N. Polyzotis and M. Garofalakis. "Statistical Synopses for Graph Structured XML Databases". In *Proc. SIGMOD* , 2002.

[12] N. Polyzotis and M. Garofalakis. "Structure and Value Synopses for XML Data Graphs". In *Proc. VLDB*, 2002.

[13] N. Polyzotis, M. Garofalakis, and Y. Ioannidis. Selectivity Estimation for XML Twigs. 2003.

[14] V. Poosala and Y. E. Ioannidis. "Selectivity Estimation Without the Attribute Value Independence Assumption". In *Proc. VLDB*, 1997.

[15] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". In *Proc. SIGMOD* , 1996.

[16] J. S. Vitter and M. Wang. "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets". In *Proc. SIGMOD* , 1999.

[17] Y. Wu, J. M. Patel, and H. Jagadish. "Estimating Answer Sizes for XML Queries". In *Proc. EDBT*, 2002.