Structure and Value Synopses for XML Data Graphs

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

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

Abstract

All existing proposals for querying XML (e.g., XQuery) rely on a pattern-specification language that allows (1) *path navigation and branching through the label structure* of the XML data graph, and (2) *predicates on the values* of specific path/branch nodes, in order to reach the desired data elements. Optimizing such queries depends crucially on the existence of concise synopsis structures that enable accurate compile-time selectivity estimates for complex path expressions over graph-structured XML data. In this paper, we extent our earlier work on structural XSKETCH synopses and we propose an (augmented) XSKETCH synopsis model that exploits localized stability and value-distribution summaries (e.g., histograms) to accurately capture the complex correlation patterns that can exist between and across path structure and element values in the data graph. We develop a systematic XSKETCH estimation framework for complex path expressions with value predicates and we propose an efficient heuristic algorithm based on greedy forward selection for building an effective XSKETCH for a given amount of space (which is, in general, an $\mathcal{NP}$-hard optimization problem). Implementation results with both synthetic and real-life data sets verify the effectiveness of our approach.

1 Introduction

The Extensible Markup Language (XML) is rapidly emerging as the new standard for data representation and exchange on the Internet. The simple, self-describing nature of the XML standard promises to enable a broad suite of next-generation Internet applications, ranging from intelligent web searching and querying to electronic commerce. In many respects, XML represents an instance of *semistructured data* [13, 15]: the underlying data model comprises a labeled graph of *element* nodes, where each element can be either an atomic data item (i.e., raw values stored with elements) or a composite data collection consisting of references (represented as graph edges) to other elements in the graph. Further, *labels* (or, *tags*) stored with XML data elements describe the actual semantics of the data.

Sophisticated query-processing engines that allow users and applications to effectively tap into the large amounts of data stored in XML databases around the globe are going to be crucial to fulfilling the full potential of XML and enabling Internet-scale applications. Realizing such Internet-scale XML query processors (like, e.g., Xyleme (`www.xyleme.com`) or Niagara [17]), in turn, hinges on providing effective support for high-level, declarative XML query languages. A variety of languages have been proposed for querying semistructured and XML databases, including XQuery [4], Lorel [15], and UnQL [3]. A common characteristic of all existing language proposals, is the existence of a pattern-specification language (like, e.g., XPath [7]) built around *path* and *subtree ("twig") expressions*. These expressions replace the traditional SQL `FROM` clause and enable selections based on *value predicates* as well as *path navigation and branching* through the XML data graph in order to reach the relevant data elements. While simple path queries were popularized in the context of object-oriented databases, the pattern-specification languages proposed for graph-structured XML data are substantially more complex. In particular, the XPath language [7] (that lies at the core of XQuery [4] and XSLT [6], the dominant W3C language proposals for XML querying and transformation) allows *branching regular path expressions* that enable queries to navigate along paths in the data graph using label names, wild cards, value predicates and branching predicates on the existence of specific sibling paths. As a concrete example, in a bibliography database, the XPath expression `//author[book]/paper/vldb[year > 1997]/title` selects the set of all VLDB paper `titles` published after 1997 by `authors` that have published *at least one* `book` (specified by the `author[book]` branch).

Optimizing XML queries with complex path expressions depends crucially on the ability to obtain effective compile-time estimates for the selectivity of these expres-

**Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002**

sions over the underlying (large) graph-structured XML database. Similar to relational query optimization, selecting an efficient query-execution plan relies on the accurate estimation of the number of XML elements that are accessed from (i.e., "satisfy") a path-expression specification. Clearly, to be feasible at query-optimization time, this estimation process has to depend on a concise and accurate *statistical synopsis* of the structure and values of the XML data graph that can provide such selectivity estimates within the memory and time constraints of the optimizer. Of course, such a synopsis can also be an invaluable tool for providing users with fast approximate answers and quick feedback to their queries, either before or during query execution.

Prior Work.[1] Summarizing a large XML data graph for the purpose of estimating the selectivity of arbitrary path expressions with value predicates is a substantially different and more difficult problem than that of constructing synopses for flat, relational data (e.g., [22, 23]). Recent research studies [1, 5, 12, 24] have considered specialized variants of our XML summarization problem, focusing on the simplified case of *tree-structured* (rather than graph-structured) data and restricted path expressions (e.g., simple paths with no branching predicates). It is unclear if these earlier techniques can be extended to general, graph-structured XML databases (where non-tree edges can arise naturally as explicit element references through `id`/`idref` attributes or XLink constructs [2, 8]) with element values.

Recent proposals for exact and approximate *path-index structures* for XML (e.g., [13, 14, 16]) also attempt to capture the path structure in the underlying XML data graph. Unfortunately, the usefulness of such structures as optimization-time synopses for selectivity estimation is limited, since (a) exact indexes (e.g., the 1- and T-index) can grow to a fairly large proportion of the data-graph size [14, 16]; and, (b) approximate indexes (e.g., the A(k)-index [14]) do not explicitly try to capture the essential statistical characteristics of the data-graph distribution. Further, no path-index structure has addressed the issues that arise in the presence of element values.

Our Contributions. In our earlier work [20], we have proposed the XSKETCH synopsis model for effectively capturing the key *structural* (i.e., label path and branching) characteristics of large XML data graphs. In this paper, we tackle the difficult problem of augmenting our structural XSKETCH synopses with concise, yet accurate distribution information on the *element values* in the XML data. Our proposed (augmented) XSKETCH synopses provide an effective tool for selectivity estimation of fully-general, complex path expressions that can incorporate *value predicates* on *any* of the nodes in the label path or branches of the query structure. To the best of our knowledge, ours is the first work to address this timely problem in the most general setting of graph-structured XML data with element values. More concretely, the key contributions of our work are summarized as follows.

• **Definition and Systematic Estimation Framework for (Structure and Value)** XSKETCH **Synopses.** We give a formal definition of our XSKETCH synopsis model that exploits *localized stability* and *value-distribution summaries* (e.g., histograms) to accurately capture the complex correlation patterns that can exist between and across path structure and element values in the XML data graph. We develop a systematic estimation framework for parsing complex path expressions with value predicates over a concise XSKETCH synopsis and producing an approximate selectivity estimate. Like any estimation technique based on concise data synopses, our proposed framework relies on appropriate statistical (uniformity and independence) assumptions to compensate for the lack of detailed information.

• **Efficient** XSKETCH **Construction Algorithm.** Building effective XSKETCH synopses is a hard optimization problem that we have demonstrated to be $\mathcal{NP}$-hard even for the much simpler "structure-only" case [20]. Given the intractability of the problem, we propose an efficient heuristic algorithm for XSKETCH construction based on greedy forward selection. Briefly, our algorithm works by successive structural and value-distribution refinements that progressively evolve a very coarse initial summary to a more accurate synopsis.

• **Implementation Results Validating the** XSKETCH **Approach.** We present results from an experimental study of XSKETCHes with synthetic and real-life data sets that verify the effectiveness of our approach. Our results demonstrate the effectiveness of XSKETCHes in capturing important data path and value correlations using only limited space.

2 Background

XML Data Model. Following previous work on XML and semistructured data [13, 14, 16], we model an XML database as a directed, node-labeled *data graph* $G = (V_G, E_G)$. Each node in V_G corresponds to an XML element in the database and is characterized by (a) a unique *object identifier (oid)*, (b) a *label* (assigned from some alphabet of string literals) that captures the element's semantics, and (c) (possibly) a set of *raw data values* for the element. (We use $\texttt{label}(v)$, $\texttt{value}(v)$ to denote the label and value(s) of $v \in V_G$.) Edges in E_G are used to capture both the element-subelement relationships (i.e., element nesting or explicit element references through `id`/`idref` attributes or XLink constructs [2, 8, 14, 15]) and the element-value relationships in the XML data. Note that non-tree edges, such as those implemented through `id`/`idref` constructs, are an essential component and a

[1] Due to space constraints, a detailed overview of related work can be found in the full version of this paper [21].

"first-class citizen" of XML data that can be directly queried in complex path expressions [4]. As an example, Figure 1(a) depicts an example XML data graph modeled after the Internet Movie Database (IMDB) XML data set (`www.imdb.com`), showing two movies and three actors. The graph node corresponding to a data element is named with an abbreviation of the element's label and a unique id number. Also, as shown, each movie (`M`) element is associated with two values (i.e., `year` of production and `box-Office` sales), and each actor (`A`) element is associated with one value (i.e., `name` of the actor); the specific values for each element are not shown to avoid cluttering the figure. Dashed lines are used for graph edges corresponding to `id-idref` relationships.

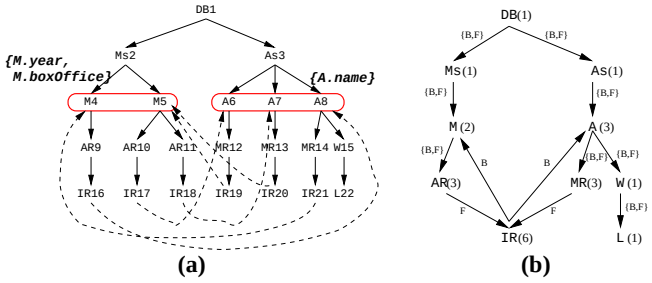


Figure 1: Example XML Data Graph (a) and Structural XSKETCH Synopsis (b).

XML Query Model. Abstractly, an XML path expression $\bar{\mathtt{l}}$ (e.g., in XQuery [4]) defines a navigational path over the XML data graph, specifying conditions on the labels and (possibly) the value(s) of data elements. A *simple path expression* is of the general form $\mathtt{l}_1\{\sigma_1\}/\cdots/\mathtt{l}_n\{\sigma_n\}$, where each $\mathtt{l}_i$ denotes an element label and each σ_i denotes a (possibly empty) *selection predicate* on the value(s) of corresponding elements. The result set of this path expression includes all elements e_n for which there exists a path $e_1/\cdots/e_n$ in the data with $\mathtt{label}(e_i) = \mathtt{l}_i$ and $\mathtt{value}(e_i)$ satisfying predicate σ_i. [2]

More complex, *branching path expressions* have the general form $\bar{\mathtt{l}} = \mathtt{l}_1\{\sigma_1\}[\bar{\mathtt{l}}^1\{\bar{\sigma^1}\}]/\cdots/\mathtt{l}_n\{\sigma_n\}[\bar{\mathtt{l}}^n\{\bar{\sigma^n}\}]$, where $\mathtt{l}_i$ and σ_i denote labels and value predicates, and $\bar{\mathtt{l}}^i\{\bar{\sigma^i}\}$ are (possibly empty) branching path expressions. A branching path expression is formed from a simple path expression $\mathtt{l}_1\{\sigma_1\}/\cdots/\mathtt{l}_n\{\sigma_n\}$ by attaching the *branch predicates* $\bar{\mathtt{l}}^i\{\bar{\sigma^i}\}$ at specific labels. Each $[\bar{\mathtt{l}}^i\{\bar{\sigma^i}\}]$ clause represents an *existential* condition, requiring that there exists at least one $\bar{\mathtt{l}}^i\{\bar{\sigma^i}\}$ twig at point i of the expression. For example, consider a query over the data graph of Figure 1(a) that looks for all movies starring an actor that has at least one WebLink (W) child and that have grossed over \$100M in box-office sales; we can express this query as the branching-path expression `Actor[WebLink]/MovieRef/IDREF/-Movie{boxOffice>100M}`. Note that, assuming that both M4 and M5 grossed over \$100M, our example expression would return only element M4. To simplify the notation, we often use $\bar{\mathtt{l}}$ and $\bar{\sigma}$ as a shorthand for the label structure and the set of value predicates in a path expression, and $\bar{\mathtt{l}}\{\bar{\sigma}\}$ as a shorthand for the full, "value-restricted" path expression.

[2] Our notation is slightly different from that of XQuery, as we are using {} to distinguish value predicates from element-branching predicates (denoted by []). Also, to simplify the presentation, we do not show explicit dereference operators for `id-idref` edges in our path expressions.

3 Structure and Value XSKETCH Synopses

3.1 Overview of Structural XSKETCHes

Abstractly, our *structural* XSKETCH synopsis mechanism [20] relies on a generic graph-summary model for an XML data graph $G = (V_G, E_G)$, which is essentially a node-labeled, directed graph structure $\mathcal{S}(G) = (V_\mathcal{S}, E_\mathcal{S})$, where: (1) each node in $v \in V_\mathcal{S}$ corresponds to a subset of data nodes in a partitioning of V_G (termed the *extent* of v – $\mathtt{extent}(v)$) that have the *same label* (denoted by $\mathtt{label}(v)$); 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. To enable selectivity estimates for complex path expressions, each node v of $\mathcal{S}(G)$ only captures summary information about G in the form of a *count* field ($\mathtt{count}(v)$) that records the number of elements in G that map to v, i.e., the size of v's `extent`. (We use v and $\mathtt{extent}(v)$ interchangeably in what follows.)

Our structural XSKETCH synopses are specific instantiations of this generic graph-synopsis model that record some additional edge-label information to capture localized *backward- and forward-stability* conditions across synopsis nodes [18, 20]. We say that a node u is *B(ackward)-stable* (*F(orward)-stable*) with respect to a parent (resp., child) node v in the synopsis, iff all data elements in $\mathtt{extent}(u)$ have at least one parent (resp., child) element in $\mathtt{extent}(v)$. Intuitively, B-stability guarantees that all elements in u descend from v and, therefore, $\mathtt{count}(u)$ is an exact estimate for the expression v/u; similarly, F-stability ensures that all elements in u reach at least one element in v and, therefore, $\mathtt{count}(u)$ is an exact estimate for $u[v]$.

Definition 3.1 A *structural* XSKETCH $\mathcal{XS}(G) = (V_\mathcal{XS}, E_\mathcal{XS})$ for a data graph G is an *edge-labeled* graph synopsis for G, where the label for each edge $(u, v) \in E_\mathcal{XS}$ is defined as: (1)$\mathtt{label}(u, v) = \{\mathbf{B}\}$, if v is B-stable wrt u; (2)$\mathtt{label}(u, v) = \{\mathbf{F}\}$, if u is F-stable wrt v; (3)$\mathtt{label}(u, v) = \{\mathbf{B}, \mathbf{F}\}$, if both (1) and (2) hold; and, (4)$\mathtt{label}(u, v) = \phi$ (empty), otherwise. ∎

Figure 1(b) depicts an example structural XSKETCH synopsis for the small IMDB data graph in Figure 1(a). Our earlier work [20] has proposed a systematic estimation framework that approximates the selectivity of branching label paths (with no value predicates) over concise structural XSKETCHes, and has described effective construction algorithms for building such synopses.

3.2 Incorporating Value-Distribution Information

A naive solution to adding information on element values would be to directly apply our structural XSKETCH ideas, simply treating different data values as different "labels" in the graph. Of course, the problem with such an approach is that the number of distinct values in an XML database is typically far greater than the number of distinct element labels; thus, such a naive solution is likely to cause an explosion in the size of any structural graph synopsis. Even the coarsest graph synopsis (i.e., the *label-split graph* that simply groups data nodes by label [20]) can become too large to be useful as an optimization-time structure, whereas the perfect graph synopsis (i.e., the fully *B/F-bisimilar graph* [20]) can easily be as large as the database itself. Instead, the key idea of our proposed approach is to incorporate compressed value-distribution information (using, e.g., histograms) in the nodes of an XSKETCH synopsis in order to effectively capture the distribution of element values in nodes' extents. Despite its deceptive simplicity, this turns out to be a rather difficult problem since, to be effective, the resulting XSKETCH synopses need to accurately capture complex correlation patterns that may exist in the underlying graph-structured data. More specifically, there are two key forms of correlations that our synopses need to model.

(1) Path/Value Correlations: Given a node v in the XSKETCH, the characteristics of the value distribution for data elements in $\texttt{extent}(v)$ can vary drastically depending on the specific *label path(s)* that reach these elements (or, leave from these elements) in the data. For example, consider a bookstore database and a `book`-labeled node v in the synopsis that records book-pricing information; obviously, the prices of elements in $\texttt{extent}(v)$ reached through the label path `cs/textbooks/book` will be very different from those reached through `poetry/rare` $-$ `collections/book`. Thus, just maintaining a histogram for the complete set of prices under v is very likely to produce inaccurate estimates for selections on `book` prices that specify either of the two label paths.

(2) Value Correlations: Given a node v in the XSKETCH, the distribution characteristics for elements in $\texttt{extent}(v)$ that are reached through (or, lead to) a *specific label path*, can depend to a large extent on the *values* of other elements on that path. Continuing with our bookstore example, assume that we have separated out in a node v' of our synopsis all `book` elements that are reached through the label path `publisher/cs/textbooks/book`, and that we have both an "expensive" and a "cheap" publisher in our database; then, clearly, the prices under node v' are correlated to the names at the `publisher` node that lead to them. Thus, the selectivity of the path expression `publisher{name = X}/cs/textbooks/book{price>$100}` estimated at v' can be very different depending on whether `X` = "expensive" or `X` = "cheap".

3.2.1 Our Solution: Structure and Value XSKETCHes

Correlations in (flat) relational-data synopses are typically modeled using concise, *multi-dimensional* representations (e.g., histograms, wavelets) for the joint distribution of correlated attributes [9, 22, 23]. As the above discussion demonstrates, the graph-structured nature of XML data poses additional challenges for the effective summarization of element-value distributions in an XSKETCH, as we need to capture correlations across both data *values* and data *structure*. More specifically, consider the set of all elements in the extent of a specific XSKETCH node v. Different subsets of elements in $\texttt{extent}(v)$ (with, possibly, different value characteristics) may be reachable by different label paths in the data (path/value correlations) and different value predicates on different labels on the path (value correlations). Clearly, keeping separate joint-distribution information (e.g., multi-dimensional histograms or wavelet synopses) for subsets of elements in $\texttt{extent}(v)$ for all possible combinations of incoming label paths and value-predicate assignments is impractical – the number of combinations is simply too large and, for accurate estimates, we would also need to capture overlap information between such subsets of $\texttt{extent}(v)$, thus exploding the complexity and space requirements of the synopsis. Instead, our XSKETCH nodes capture joint-distribution information only along paths and branches of the synopsis that are *common to all elements* in a node's extent. Thus, given an XSKETCH node v, the joint-distribution information recorded for elements in $\texttt{extent}(v)$ considers only the value correlations in v's *stable twig neighborhood* in the XSKETCH.

Definition 3.2 Let v be an XSKETCH node, and let $B(v)$ denote the set of all nodes in the XSKETCH that reach v through a B-stable path (including v itself). Also, let $F(v)$ denote the set of all nodes in the XSKETCH that can be reached starting from any node in $B(v)$ through an F-stable path. The *stable twig neighborhood (*$\texttt{STN}$*)* of v is defined as $\texttt{STN}(v) = B(v) \cup F(v)$. ∎

The key observation here is that, by virtue of stability, the stable twig neighborhood of an XSKETCH node v captures path and branching structure that is common to all data elements in $\texttt{extent}(v)$. The joint-distribution information recorded for v in the XSKETCH tries to capture the frequencies of elements in $\texttt{extent}(v)$ and their possible correlation(s) with the values of (a subset of) other nodes in $\texttt{STN}(v)$ along specific paths (or, in general, twigs) within $\texttt{STN}(v)$. In other words, our XSKETCH synopses rely on (a) *structural B- and F-stability* to model the dependence of element values on path and branching structure (i.e., path/value correlations), and (b) *multi-dimensional distribution synopses (e.g., histograms)* to model value correlations within stable "neighborhoods" of the XSKETCH.

Consider an XSKETCH node v and let T be a twig contained within $\texttt{STN}(v)$. Let $\texttt{dep}(v)$ ($\subseteq \texttt{STN}(v)$) denote the "correlation scope" of v; that is, the set of nodes in the

XSKETCH for which correlations with the element distribution in $\texttt{extent}(v)$ are captured in the joint-distribution information maintained in v. (If v itself contains elements with values then $\texttt{dep}(v)$ must contain at least v.) We also let $\texttt{dep}(v,T)$ denote the restriction of $\texttt{dep}(v)$ in T (i.e., $\texttt{dep}(v,T) = \texttt{dep}(v) \cap T$). Then, the joint-distribution information recorded in v can give direct estimates for the number of v elements that are reached through the twig query $T\{\overline{\sigma}\}$, where the value predicates σ can be on any subset of $\texttt{dep}(v,T)$. As a more concrete example, consider the twig $T = v_1[v_2]/v_3[v_4]/v_5$ (shown in Figure 2) that is contained within $\texttt{STN}(v_5)$, and assume $\texttt{dep}(v_5) = \{v_1, v_4\}$; then, the joint-distribution kept at v_5 can be used to directly estimate the number of v_5 elements discovered by $v_1\{\sigma_1\}[v_2]/v_3[v_4\{\sigma_4\}]/v_5$, where σ_1, σ_4 are value predicates on nodes v_1 and v_4, respectively. We can now give a formal definition for our model of structure and value XSKETCH synopses for XML data.

Definition 3.3 An XSKETCH synopsis $\mathcal{XS}(G)$ for an XML data graph G *with element values* is a structural XSKETCH $(V_{\mathcal{XS}}, E_{\mathcal{XS}})$ for G, where each node $v \in V_{\mathcal{XS}}$ can also contain a (possibly) multi-dimensional synopsis for the joint distribution of elements in $\texttt{extent}(v)$ and the values of any subset of XSKETCH nodes $\texttt{dep}(v) \subseteq \texttt{STN}(v)$ along specific paths/twigs within $\texttt{STN}(v)$. ∎

Note that, by the above definition, $\texttt{dep}(v) \subseteq \texttt{STN}(v)$ since, in general, only a subset of the value distributions in a node's STN will actually be correlated, and it is only these correlations (along a given stable twig) that we need to capture in the XSKETCH node. For non-correlated value distributions, an *independence assumption* is valid and will give accurate estimates [9]. Consider once again our example stable twig $T = v_1[v_2]/v_3[v_4]/v_5$ with $\texttt{dep}(v_5) = \{v_1, v_4\}$, and assume that the distribution of v_5 elements is independent of the values in $v_3 \notin \texttt{dep}(v_5)$ (i.e., $v_3 \perp v_5$). Even though the distribution information in v_5 cannot directly give an estimate for the count of $v_1\{\sigma_1\}[v_2]/v_3\{\sigma_3\}[v_4\{\sigma_4\}]/v_5$ (which contains a value predicate on v_3), a product estimate based on the independence of the distributions in v_5 and v_3 is going to give a good approximation (Figure 2).

Having formally defined our XSKETCH synopsis model, the remainder of this paper focuses on the two key, interrelated challenges of our XML-graph summarization problem, namely: (1) *Defining an estimation framework for complex path expressions over* XSKETCH *synopses* that relies on well-founded statistical assumptions (e.g., independence or uniformity) to compensate for the lack of detailed information and the approximate nature of the XSKETCH; and, (2) *Designing practical algorithms for effective* XSKETCH *construction* that, given a limited space budget, try to minimize the approximation error in the selectivity estimates (based on our estimation framework). Before that, however, we discuss the joint-distribution information maintained at individual XSKETCH nodes in more detail.

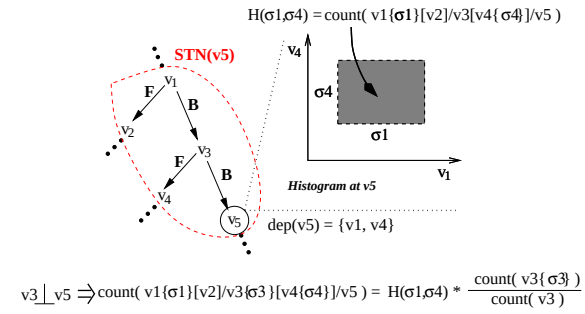


Figure 2: Example XSKETCH Value-Distribution Information.

3.2.2 Distribution Summaries for XSKETCH Nodes

Thus far, we have been deliberately vague about the exact form of distribution summaries maintained in the nodes of our XSKETCH synopsis. A main reason for this is that, as mentioned earlier, several forms of multi-dimensional data synopses (e.g., histograms or wavelets) can be used to produce a concise description of the distribution of elements in a node's extent across the values in its correlation scope. More specifically, let v be an XSKETCH node and let $\mathcal{D}_v$ denote the cross-product of the value domains for elements in v's correlation scope, i.e., $\mathcal{D}_v = \times_{u \in \texttt{dep}(v)} domain(u)$. Then, the distribution of v's elements within its correlation scope can be described by the joint-frequency table $f_v[c_1, \ldots, c_k]$ that gives the number of elements in $\texttt{extent}(v)$ that are reached from (or, lead to) the tuple of values $(c_1, \ldots, c_k) \in \mathcal{D}_v$ in the corresponding nodes of $\texttt{dep}(v)$. This frequency table $f_v[]$ can be summarized in our XSKETCH using conventional multi-dimensional histograms [22], Haar wavelets [23], or even more complex summarization techniques based on statistical modeling [9]. For concreteness, we use the term "histogram" to refer to the distribution information maintained in XSKETCH nodes in the remainder of this paper.

Unlike the relational case, however, the joint-frequency table $f_v[]$ is *not* sufficient to provide accurate estimates for arbitrary value-selection predicates over the nodes of $\texttt{dep}(v)$. The graph-structured nature of XML data once again introduces novel challenges in capturing the element distribution in a node with respect to other nodes in its STN. As a simple example, consider a v-labeled node with 10 u-labeled children $(u_1, \ldots, u_{10})$ in the data graph, and 9 other v-labeled nodes all with a single u-labeled child (u_{11}). Further, assume that v nodes carry no values whereas each u_i node carries a value of i $(i = 1, \ldots, 10)$. Clearly, in our XSKETCH synopsis this simple data configuration will result in a single B/F-stable (v, u) edge with $\texttt{count}(v) = \texttt{count}(u) = 10$. Assume that $\texttt{dep}(v) = \{u\}$. Then, the joint-frequency table $f_v[]$ will have entries $f_v[i] = 1$ for $i = 1, \ldots, 10$ and $f_v[11] = 9$. Now consider

the branch query $v[u\{\sigma\}]$ where $\sigma = (1 \leq \texttt{value}(u) \leq 10)$. Clearly, the correct $\texttt{count}(v[u\{\sigma\}])$ is 1; however, using the $f_v[]$ table in the conventional manner to estimate the selectivity of σ (assuming no summarization whatsoever) we get an erroneous count of 10. The reason, of course, is *"double counting"*: even though the frequency table tells us that each u-value from 1 to 10 is reached by one v-element, it has no way of telling us that they are in fact reached by the *same* v-element! It is easy to see that this double-counting problem can become much more complicated when the element distribution involves more complex path structures and overlap patterns between elements. Unfortunately, this problem is inherent in all approximation techniques that estimate the selectivity of ranges by summing point frequencies and there is no easy solution based on traditional frequency tables and histogram structures[3].

We have proposed an initial solution for avoiding double-counting in XSKETCH nodes, using specialized *Range Histograms* that explicitly capture the overlap between different value ranges. Intuitively, range histograms approximate range frequencies instead of point frequencies and essentially map selectivities of ranges to points in a higher-dimensional space. The complete details can be found in the full version of this paper [21].

4 Estimation over XSKETCH Synopses

We now define our estimation framework for approximating path-expression selectivities over a compact XSKETCH synopsis. Our framework generalizes that described in our earlier work for the "structure-only" case [20] to deal with predicates on node values.

4.1 Parsing Path Expressions over an XSKETCH

Let $\overline{\texttt{l}}\{\overline{\sigma}\} = \texttt{l}_1\{\sigma_1\}/\cdots/\texttt{l}_n\{\sigma_n\}[\texttt{l}_{n+1}\{\sigma_{n+1}\}/\cdots/\texttt{l}_{n+k}\{\sigma_{n+k}\}]/\texttt{l}_{n+k+1}\{\sigma_{n+k+1}\}/\cdots/\texttt{l}_{n+k+m}\{\sigma_{n+k+m}\}$ denote a branching label path over an XML data graph G, with one branching predicate on label $\texttt{l}_n$ and a sequence of value predicates $\overline{\sigma}$ defined over all labels in the path. (Our discussion can be simplified if predicates are only specified for some of the nodes in the label.) Even though we consider the case of a single-branch path expression, our methodology can be easily generalized for multi-branch twigs. We use $\texttt{count}(\overline{\texttt{l}}\{\overline{\sigma}\})$ to denote the (estimated) number of data elements that are discovered by the path expression $\overline{\texttt{l}}\{\overline{\sigma}\}$ in G, i.e., the *selectivity* of $\overline{\texttt{l}}\{\overline{\sigma}\}$. Consider an XSKETCH synopsis $\mathcal{XS}(G)$ of the data, and let $\overline{v} = v_1/\cdots/v_n[v_{n+1}/\cdots/v_{n+k}]/v_{n+k+1}/\cdots/v_{n+k+m}$ be a path in $\mathcal{XS}(G)$ such that, for each i, $\texttt{label}(v_i) = \texttt{l}_i$; we term such an XSKETCH path $\overline{v}$ an *embedding* of the label path $\overline{\texttt{l}}$. Similar to $\overline{\texttt{l}}$, the nodes of the embedding $\overline{v}$ can also be augmented with the appropriate value predicates σ_i to obtain the "value-restricted" embedding $\overline{v}\{\overline{\sigma}\}$. An element e_n in $\texttt{extent}(v_n)$ is discovered by this embedding $\overline{v}\{\overline{\sigma}\}$ if there exists a document path $e_1/\cdots/e_n[e_{n+1}/\cdots/e_{n+k}]/e_{n+k+1}/\cdots/e_{n+k+m}$ such that: (1) $e_i \in \texttt{extent}(v_i)$, and (2) the value of element e_i satisfies the corresponding predicate σ_i, for each $i = 1, \ldots, n+k+m$. It is obvious that if an element e is discovered by embedding $\overline{v}\{\overline{\sigma}\}$, then it also belongs in the target set of our original path expression $\overline{\texttt{l}}\{\overline{\sigma}\}$. Thus, if we use $\varepsilon(\overline{\texttt{l}})$ to denote the set of all *distinct embeddings* $\overline{v}$ of $\overline{\texttt{l}}$ in our XSKETCH synopsis (i.e., embeddings that differ in at least one node in the XSKETCH path), then the selectivity of $\overline{\texttt{l}}\{\overline{\sigma}\}$ is estimated by summing the selectivities over all embeddings in $\varepsilon(\overline{\texttt{l}})$; that is, $\texttt{count}(\overline{\texttt{l}}\{\overline{\sigma}\}) = \sum_{\overline{v}\in\varepsilon(\overline{\texttt{l}})} \texttt{count}(\overline{v}\{\overline{\sigma}\})$, where $\texttt{count}(\overline{v}\{\overline{\sigma}\})$ denotes the estimated number of elements discovered by a (value-restricted) embedding $\overline{v}\{\overline{\sigma}\}$. (Of course, we ensure that a synopsis node cannot contribute more than its total count to this estimate.)

Our selectivity estimation problem, therefore, essentially reduces to estimating the count of the data elements discovered by each distinct embedding of the label path in $\mathcal{XS}(G)$. This count can be expressed as $\texttt{count}(\overline{v}\{\overline{\sigma}\}) = \texttt{count}(v_{n+k+m}) \times f(\overline{v}\{\overline{\sigma}\})$, where $f(\overline{v}\{\overline{\sigma}\})$ denotes the estimated *fraction* (i.e., empirical probability) of elements in $\texttt{extent}(v_{n+k+m})$ that are discovered by the (value-restricted) embedding $\overline{v}\{\overline{\sigma}\}$. Estimating the fraction $f(\overline{v}\{\overline{\sigma}\})$ over the XSKETCH is the key problem that needs to be addressed in our estimation framework; we now explain our solution in detail.

Consider a single-branch path embedding $\overline{v} = v_1/\cdots/v_n[u_1/\cdots/u_k]/v_{n+1}/\cdots/v_{n+m}$ (note that we are using u_i for branch nodes to simplify the notation). The first step in our estimation process is to parse the embedding $\overline{v}$ into a sequence of *maximal, non-overlapping stable twigs*; that is, we break the embedding $\overline{v}$ into a collection of sub-twigs $T_1, T_2, \ldots$, such that every XSKETCH node in the embedding is "covered" by exactly one of the T_i's, and each T_i is a maximal stable twig in $\overline{v}$, i.e., all path (branch) edges in T_i are B-stable (resp., F-stable). This parsing can be done as follows. Starting from the last node in the embedding (v_{n+m}), build the first tree T in the decomposition by taking the intersection of the embedding $\overline{v}$ with the stable twig neighborhood of v_{n+m}; i.e., $T = \texttt{STN}(v_{n+m}) \cap \overline{v}$. Then, take the nodes of T out of $\overline{v}$ and repeat the process from those nodes of $\overline{v} - T$ that were directly connected to T nodes (i.e., at the outside "border" of T).

It is easy to see that, for the case of our single-branch embedding $\overline{v}$, all the stable twigs resulting from the above decomposition will be simple paths except perhaps for the single twig, say T_j that contains the only branching node v_n. More concretely, let the stable-twig decomposition of $\overline{v}$ be as follows: $T_1 = v_1/\cdots/v_{k_1}$, ...,

[3] A naive approach would be to incorporate element-id information in the dimensions of our frequency table. Such an approach, however, is very inflexible with respect to updates in the database and (perhaps most importantly) it is not at all clear how element-id axes should be handled during the summarization (histograming) of the table.

$T_j = v_{k_{j-1}+1}/\cdots/v_n[u_1/\cdots/u_{m_1}]/\cdots/v_{k_j}$, $\ldots$ $T_q = v_{k_{q-1}+1}/\cdots/v_{k_q}$, $T_{q+1} = u_{m_1+1}/\cdots/u_{m_2}$, $\ldots$, $T_{q+l} = u_{m_l+1}/\cdots/u_{m_{l+1}}$, where $0 < k_1 < k_2 < \cdots < n+m = k_q$ and $0 < m_1 < m_2 < \cdots < k = m_{l+1}$. Note that, in the above decomposition, the stable twigs $T_1, \ldots, T_q$ essentially cover the main path of the expression (with the only "true" twig T_j possibly covering part of the branch), whereas twigs $T_{q+1}, \ldots, T_{q+l}$ cover the remainder of the $u_1/\cdots/u_k$ branch. Given this decomposition of the embedding $\overline{v}$, we now employ the well-known *Chain Rule* from probability theory [10] to rewrite the required fraction as follows (for simplicity, we use $\overline{\sigma_i}$ for the set of all predicates in twig T_i and π_j for the predicate on branch node u_j):

$$\begin{aligned} f(\overline{v}\{\overline{\sigma}\}) = & f(T_q\{\overline{\sigma_q}\}) \cdot \prod_{i=j}^{q-1} f(T_i\{\overline{\sigma_i}\}/v_{k_i+1} \mid T_{i+1}\{\overline{\sigma_{i+1}}\}/\cdots/T_q\{\overline{\sigma_q}\}) \\ & \cdot \prod_{i=1}^{l} f(u_{m_i}[T_{q+i}\{\overline{\sigma_{q+i}}\}] \mid T_j\{\overline{\sigma_j}\}[u_{m_1+1}\{\pi_{m_1+1}\}/ \\ & \qquad \cdots/u_{m_i}\{\pi_{m_i}\}]/T_{j+1}\{\overline{\sigma_{j+1}}\}/\cdots/T_q\{\overline{\sigma_q}\}) \\ & \cdot \prod_{i=1}^{j-1} f(T_i\{\overline{\sigma_i}\}/v_{k_i+1} \mid T_{i+1}\{\overline{\sigma_{i+1}}\}/\cdots/T_j\{\overline{\sigma_j}\} \\ & \qquad [u_1\{\pi_1\}/\cdots/u_k\{\pi_k\}]/\cdots/T_q\{\overline{\sigma_q}\}). \end{aligned}$$

Note that the second product term above captures the selectivity of the complex expression along the "existential" branch $u_1/\cdots/u_k$, whereas the first and third product terms capture the selectivity along the main path. Of course, by the Chain Rule, the $f()$ frequency terms always condition on the remainder of the complex path as it is parsed in a "bottom-up" fashion; intuitively, the reason for this conditioning is to capture the *correlations* between the various twigs that comprise the overall path embedding $\overline{v}\{\overline{\sigma}\}$. Also, note that the value predicates for the v_{k_i+1} and u_{m_i} nodes are not included in the fraction expressions for the T_i and T_{q+i} terms above, since they are already accounted for in the expression for T_{i+1} and T_{q+i-1} (respectively) and included in the corresponding conditionals for T_i and T_{q+i}. To simplify the above expression, our estimation process makes the following "Twig-Independence" assumption.

A1. [Twig Independence] Given a node v in $\mathcal{XS}(G)$, the distribution of incoming twigs $T\{\overline{\sigma}\}$ to v is *independent* of the distribution of outgoing twigs $T'\{\overline{\sigma'}\}$ from v for any value predicates $\overline{\sigma}$, $\overline{\sigma'}$; more formally, $f(T\{\overline{\sigma}\}/v \mid v/T'\{\overline{\sigma'}\}) \approx f(T\{\overline{\sigma}\}/v)$ and $f(v[T'\{\overline{\sigma'}\}] \mid T\{\overline{\sigma}\}/v) \approx f(v[T'\{\overline{\sigma'}\}])$.

Twig Independence essentially generalizes our earlier Path- and Branch-Independence assumptions [20] to our more general twig-decomposition scheme for path expressions with value predicates. Using Assumption (A1), we can eliminate most of the conditionings and simplify our expression for $f(\overline{v}\{\overline{\sigma}\})$ to:

$$\begin{aligned} f(\overline{v}\{\overline{\sigma}\}) \approx & f(T_q\{\overline{\sigma_q}\}) \cdot \prod_{i=1}^{q-1} f(T_i\{\overline{\sigma_i}\}/v_{k_i+1} \mid v_{k_i+1}\{\sigma_{k_i+1}\}) \\ & \cdot \prod_{i=1}^{l} f(u_{m_i}[T_{q+i}\{\overline{\sigma_{q+i}}\}] \mid u_{m_i}\{\pi_{m_i}\}) \end{aligned} \quad (1)$$

That is, the only conditioning that remains over the $f()$ fractions is on the value predicate imposed on the referenced node in our XSKETCH synopsis. Now, let B_{q+i} denote the *branch expression* corresponding to the F-stable path T_{q+i} in our decomposition of the $u_1/\cdots/u_k$ branch; that is, $B_{q+i} = u_{m_i+1}[u_{m_i+2}/\cdots/u_{m_{i+1}}]$, for $i = 1, \ldots, l$. Applying the Chain Rule once again for the individual terms in the above products, we have:

$$\begin{aligned} f(T_i\{\overline{\sigma_i}\}/v_{k_i+1} \mid v_{k_i+1}\{\sigma_{k_i+1}\}) &= f(v_{k_i}/v_{k_i+1} \mid v_{k_i+1}\{\sigma_{k_i+1}\}) \\ &\quad \cdot f(T_i\{\overline{\sigma_i}\} \mid v_{k_i}/v_{k_i+1}\{\sigma_{k_i+1}\}) \\ &\approx f(v_{k_i}/v_{k_i+1} \mid v_{k_i+1}\{\sigma_{k_i+1}\}) \cdot f(T_i\{\overline{\sigma_i}\}), \end{aligned} \quad (2)$$

where the last derivation follows from Twig Independence (A1). Similarly,

$$\begin{aligned} f(u_{m_i}[T_{q+i}\{\overline{\sigma_{q+i}}\}] \mid u_{m_i}\{\pi_{m_i}\}) &\approx f(u_{m_i}[u_{m_i+1}] \mid u_{m_i}\{\pi_{m_i}\}) \\ &\quad \cdot f(B_{q+i}\{\overline{\sigma_{q+i}}\}]). \end{aligned} \quad (3)$$

To simplify the resulting fractions further, we make one more independence assumption that aims to compensate for the lack of statistical-correlation information across non-stable edges.

A2. [Edge-Value Independence Across Non-Stable Edges] Consider a node v in $\mathcal{XS}(G)$ and let u (w) be a non-B-stable parent (resp., non-F-stable child) of v in $\mathcal{XS}(G)$. Then, the distribution of incoming (outgoing) edges from u (resp., to w) across the elements in $\texttt{extent}(v)$ is *independent of the elements' values*; that is, for any predicate σ on the values of v elements, we have $f(u/v \mid v\{\sigma\}) \approx f(u/v)$ and $f(v[w] \mid v\{\sigma\}) \approx f(v[w])$.

Assumption (A2) essentially simplifies path-value correlations along non-stable edges; obviously, since it directly deals with node values and value predicates, it does not have an analog in our estimation framework for structural XSKETCHes [20]. Combining Equations (1), (2), (3), and Assumption (A2), we obtain the following final expression for the selectivity estimate of our branching-path embedding:

$$\begin{aligned} f(\overline{v}\{\overline{\sigma}\}) \quad \approx \quad & \prod_{i=1}^{q} f(T_i\{\overline{\sigma_i}\}) \cdot \prod_{i=1}^{l} f(B_{q+i}\{\overline{\sigma_{q+i}}\}]) \\ & \cdot \prod_{i=1}^{q} f(v_{k_i}/v_{k_i+1}) \cdot \prod_{i=1}^{l} f(u_{m_i}[u_{m_i+1}]) \end{aligned} \quad (4)$$

Intuitively, the above formula gives the selectivity estimate for the embedding $\overline{v}\{\overline{\sigma}\}$ as a product of two key components: (1) the fractions of elements discovered by the stable

paths and branches (in general, stable *twigs*) in the embedding, i.e., the first two product terms in Equation (4) (note that these terms capture all value predicates in $\overline{\sigma}$); and, (2) the selectivities of path or branch edges along the *"stability breaks"* in the $\overline{v}$ embedding, i.e., the last two product terms in Equation (4). It is easy to extend the estimation formula above to the most general case of *multi-branch* complex path expressions. Equation (4) generalizes our earlier selectivity-estimation formula for complex paths without value predicates [20] that, essentially, comprised only the last two product terms in (4); of course, if no value predicates are involved (i.e., all $\overline{\sigma_i}$'s are empty), then both of the leading product terms in (4) would evaluate to 1 since all the referenced paths and branches are stable [20].

The selectivity terms across "stability-break" edges in the XSKETCH ($f(v_{k_i}/v_{k_i+1})$ and $f(u_{m_i}[u_{m_i+1}])$) are estimated using the Backward- and Forward-Edge Uniformity assumptions (A3-A4) exactly as in our structural estimation framework [20]. Thus, the new challenge that arises in the presence of path expressions with both structural and value predicates is to utilize the XSKETCH-summary information to provide sound estimates for the selectivities of stable twigs with selection predicates on node values. We address this problem next.

4.2 Stable-Twig Estimation Algorithm

Consider a *stable twig embedding* T in an XSKETCH synopsis $\mathcal{XS}(G)$, and let $\overline{\sigma}$ denote a collection of value predicates over the nodes of T. Our goal is to utilize the statistical information in $\mathcal{XS}(G)$ to obtain an estimate for $f(T\{\overline{\sigma}\})$, the fraction of data elements that are discovered by the "value-restricted" twig embedding $T\{\overline{\sigma}\}$.

Once again, to simplify the exposition, we consider a single-branch stable twig T; our discussion can easily be extended to the general, multi-branch case. (We also use T as the *set* of XSKETCH nodes in the twig when no confusion arises.) Let v be a node in T. Given a set S of ancestor and/or descendant nodes of v in T, let $\texttt{twig}(v, S)$ denote the sub-twig of T that connects v to all the nodes in S, and let $\overline{\sigma}(S)$ denote the set of value predicates on nodes of S in our twig query (i.e., the restriction of $\overline{\sigma}$ to S).

Our algorithm for estimating the selectivity of a stable twig embedding $T\{\overline{\sigma}\}$ (termed TWIGEST) is depicted in Figure 3. Briefly, our TWIGEST algorithm examines each node v in the main path of the twig embedding (in reverse order) and considers the set of value predicates that can be directly *"covered"* by the node's joint-distribution histogram based on its correlation scope $\texttt{dep}(v)$. When a branching node (e.g., v_n) is encountered, TWIGEST traverses the branch top-down once again using nodes' correlation scopes to cover value predicates in the branch. This covering of value predicates is based on successive applications of the Chain Rule as we parse the twig embedding; of course, since each node carries only limited value-correlation information, our estimate relies on one final independence assumption for element-value distributions.

A5. [Value-Independence Outside Correlation Scope] The distribution of elements in the `extent` of an XSKETCH node v is *independent* of the values in other XSKETCH nodes u that are *not* in v's direct correlation scope, i.e., $u \notin \texttt{dep}(v)$.

Assumption (A5) allows us to simplify the Chain-Rule conditionals to obtain the conditional probability f^* in Step 11 of TWIGEST, which can be directly estimated using the histogram information in v. Once all value predicates in the embedding $T\{\overline{\sigma}\}$ have been covered, TWIGEST returns the accumulated selectivity estimate for $f(T\{\overline{\sigma}\})$.

---

procedure TWIGEST($\mathcal{XS}(G)$,$T\{\overline{\sigma}\}$)
Input: XSKETCH $\mathcal{XS}(G)$; stable twig $T = v_1/\cdots/v_{n-k-1}/-v_n[v_{n-1}/\cdots/v_{n-k}]/v_{n+1}/\cdots/v_{n+m}$ with value predicates $\overline{\sigma}$.
Output: Estimate of the selectivity fraction $f(T\{\overline{\sigma}\})$.
begin
1. `Covered` := ϕ; `Uncovered` := { nodes with value predicate in $\overline{\sigma}$}
2. result := 1; index := $n+m$; v := v_{index}
3. **while** `Uncovered` $\neq \phi$ **do**
4. // check for predicates covered by node v's statistics
5. **if** (`Uncovered` $\cap$ $\texttt{dep}(v,T) \neq \phi$) **then**
6. // find covered and uncovered ancestors/descendants of
7. // v that are in its direct "correlation scope"
8. C := `Covered` $\cap$ $\texttt{dep}(v,T)$
9. U := `Uncovered` $\cap$ $\texttt{dep}(v,T)$
10. Using the element distribution information (histogram) at v, compute the conditional fraction:
11. f^* := $f(\texttt{twig}(v,U)\{\overline{\sigma}(U)\} \mid \texttt{twig}(v,C)\{\overline{\sigma}(C)\})$
$$= \frac{f(\texttt{twig}(v,U\cup C)\{\overline{\sigma}(U\cup C)\})}{f(\texttt{twig}(v,C)\{\overline{\sigma}(C)\})}.$$
12. // update result estimate and covered/uncovered nodes
13. result := result $\times f^*$; `Covered` := `Covered` $\cup U$
14. `Uncovered` := `Uncovered` $- U$
15. **endif**
16. index := index -1; v := v_{index} // move to next twig node
17. **endwhile**
end

Figure 3: The TWIGEST Algorithm for Selectivity Estimation over Stable XSKETCH Twigs.

---

5 XSKETCH Construction

In this section, we turn our attention to the important problem of effective XSKETCH construction for a given synopsis space budget. Briefly, our approach is based on using successive, localized *refinement operations* to gradually evolve an initial, coarse XSKETCH into a more detailed synopsis that captures the important path and value correlations in the data. We first discuss the specific set of XSKETCH refinement operations that we use, and then present our construction algorithm in more detail.

5.1 Refinements

In order to approximate path-expression selectivities, our XSKETCH estimation framework (Section 4) relies on a number of statistical (uniformity and independence) assumptions that compensate for the lack of detailed path and value information in the synopsis. Clearly, the accuracy of an XSKETCH (and the resulting selectivity estimates) depend crucially on the validity of these assumptions and the degree to which they reflect the statistical characteristics of the underlying path/value distribution in the actual data. To build an effective XSKETCH, we need to be able to appropriately *refine* the synopsis structure for regions of the data graph where our estimation assumptions fail, since these regions are likely to result in high estimation errors. (The relational-world analog would be allocating more buckets to "difficult" data regions during histogram construction [22].) In this section, we introduce such localized refinement operations for XSKETCH synopses. We categorize our refinement operations into two types: (1) *structural refinements* that refine the path structure in the XSKETCH, and (2) *value refinements* that refine the value-distribution information maintained in XSKETCH nodes.

Structural Refinements. Our structural-refinement operations try to improve estimation accuracy by locally refining the path and branching structure of the XSKETCH in order to capture important structural correlations in the data. Abstractly, each refinement operation uses a partitioning criterion to *split* an XSKETCH node u into a set of new nodes $\{u_i\}$, so that either some uniformity/independence assumption(s) are eliminated for the new synopsis nodes $\{u_i\}$, or at least such assumptions are much more realistic for the new nodes $\{u_i\}$ than u (similar to histogram-bucket splits). Thus, successive refinements evolve the synopsis to a larger and more precise structure. Our three structural-refinement operations, namely `b-stabilize`, `f-stabilize`and `b-split`, are defined exactly as for our structural XSKETCH synopses [20], with the addition of two post-processing steps (common to all structural refinements) that handle the value distribution information (possibly) maintained in the XSKETCH node u being split. The definition of our structural refinements as well as details on the post-processing steps can be found in the full version of this paper [21].

Value Refinements. Our value-refinement operations try to improve estimation accuracy for value predicates by increasing the granularity of the value-distribution information maintained at individual XSKETCH nodes. Abstractly, there are two ways to improve the accuracy of the distribution information at an XSKETCH node u: (1) give more space (i.e., additional buckets) to the histogram(s) already in u, and (2) expand the correlation scope of u, so that additional value correlations within u's stable twig neighborhood are captured. Thus, we introduce two new value-refinement operations for XSKETCH nodes u with histogram information: (1) `add-bucket` which allocates more space to histograms, and (2) `expand`, which expands the correlation scope of a node. A detailed discussion of our two value-refinement operations can be found in the full paper [21].

5.2 Construction Algorithm

Building an XSKETCH that accurately summarizes a large XML data graph within a given space budget is, in many respects, similar to other statistical-model inference problems, where the goal is to infer an "optimal" statistical model (e.g., Bayesian or Markov) from an underlying data set. Unfortunately, we have demonstrated that (like most such problems [19]) our effective XSKETCH construction problem is $\mathcal{NP}$-hard, even for the much simpler case of purely structural XSKETCHes (i.e., when no values are present) [20].

Based on this intractability result, we propose a computationally-efficient heuristic algorithm for building XSKETCH synopses. Abstractly, our algorithm views XSKETCH construction as a search problem over the space of all possible XSKETCH synopses, and uses our localized XSKETCH refinement operations to effectively explore this space. More specifically, our algorithm (termed BUILDXSKETCH) is based on a *greedy, forward-selection* paradigm that starts out with a very coarse synopsis model and incrementally adds more complexity using our localized XSKETCH refinements. The initial (coarse) synopsis is basically the *label-split graph* [20] (that partitions data element nodes into synopsis node based solely on their label) augmented with minimal distribution information: for each synopsis node v that contains values, a trivial (i.e., single-bucket) one-dimensional histogram is created to capture the distribution of values in `extent`(v). Our XSKETCH-refinement strategy is based on the idea of *marginal gains* [11]: at each step, the refinement operation that results in the largest increase in accuracy per unit of extra space required (and, of course, does not violate our overall space budget for the synopsis) is selected for inclusion in the XSKETCH. To avoid getting trapped in local minima, BUILDXSKETCH takes a number of random steps (i.e., random node refinements) when no accuracy-improving neighbor can be found. We rely on two key techniques in order to keep the construction process tractable: (1) using a reasonably accurate (and large) *reference synopsis* of the data graph instead of the raw data for evaluating XSKETCH configurations, and (2) *biased path and node sampling* to select "interesting" regions of the data for our refinements and XSKETCH evaluations. The complete details of our XSKETCH-construction algorithm can be found in the full version of this paper [21].

6 Experimental Study

In this section, we present results from an empirical study that we have conducted using our novel XSKETCH synopses over real-life and synthetic XML data sets. Our re-

		IMDB	XMark
No. of Elements		102,755	87,480
Label Split Graph	No. Nodes	164	80
	Total Size	9.1KB	4.1KB
	Size of Hists	1.7KB	688B
Reference Synopsis	No. Nodes	49,181	83,466
	Total Size	1.9MB	3.3MB
	Size of Hists	388KB	667KB

Table 1: Characteristics of the Two Data Sets.

		IMDB	XMark
Branching Paths	w/o predicates	1901	1057
	with predicates	478	254
Simple Paths	w/o predicates	933	771
	with predicates	483	302

Table 2: Average Query Workload Result Sizes.

sults demonstrate the ability of XSKETCHes to capture important path and value correlations in the underlying data using only limited space, and to provide accurate selectivity estimates for complex path expressions.

6.1 Testbed and Methodology

Techniques. Our prototype XSKETCH implementation implements the full estimation framework of Section 4, using one-dimensional histograms to summarize value distributions under synopsis nodes with values. As a result, our estimation algorithm relies on value independence to approximate path expressions with value predicates. We found that, even though such independence assumptions are not valid in general, they are sufficiently accurate for the data sets used in our evaluation. We are currently extending our prototype with (conventional and range) multi-dimensional histograms.

Our construction algorithm considers refinements on a biased 10% sample of all summary nodes and determines the score of each operation based on a biased sample of 100 label paths. Note that, in our current implementation, XSKETCH construction does not consider `expand` refinement operations, since all node histograms are one-dimensional.

Data Sets. We use one real-life and one synthetic data set in our evaluation.

IMDB: This is a real-life, graph-structured data set from the Internet Movie Database (`www.imdb.com`). It contains a large number of `IDREF` edges resulting in a highly irregular and cyclic path structure.

XMark: This is a synthetic, graph-structured data set, modeling the activities of an on-line auction site (`www.xml-benchmark.org`). The path structure is highly irregular but the data set does not contain any cycles.

Table 1 summarizes the main characteristics of our data sets. In both cases, we observe that our accurate reference synopsis (i.e. the B/F-bisimilar graph augmented with an accurate histogram for each extent with values) requires a considerable amount of storage and is, therefore, impractical as a concise compile-time synopsis. This is obviously expected, given the highly irregular structure of the IMDB and XMark data sets. Note that the sizes reported do not include the space required to store the actual text of each label; each label is hashed to an integer and stored in a separate structure that is not part of the summary.

Workload. We evaluate the accuracy of each synopsis against a workload of positive path expressions, i.e., expressions that have a non-zero result size in the original data. We form the workload by sampling document twigs or paths from the reference graph and converting them to the corresponding label paths. The length of path expressions is uniformly distributed between 2 and 5, and the workload contains 500 path expressions with no predicates and 500 expressions with one value range predicate. Each value predicate covers a random 10% range of the value domain of the tag it is attached to. We form two types of workloads: (1) *Branching Paths*, where half of the path expressions (with and without a value predicate) contain a branching predicate, and (2) *Simple Paths*, where no path expression contains branching predicates. Table 2 summarizes the average result size of each type of path expression across our two data sets.

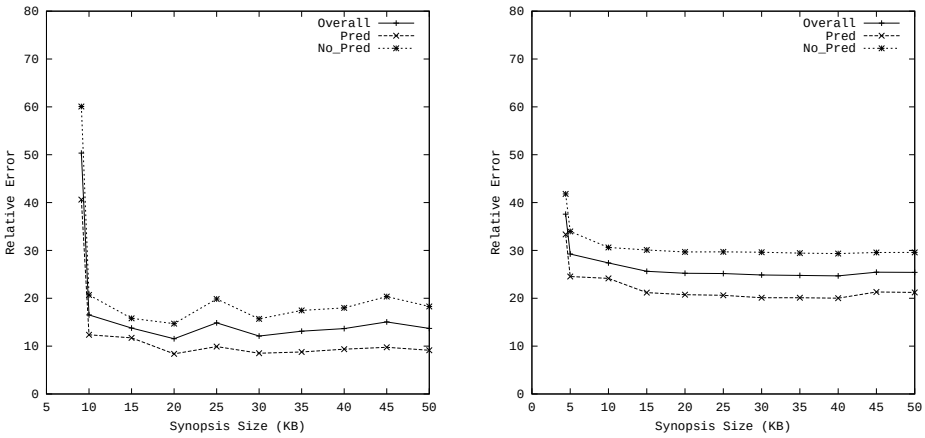
We have also experimented with negative workloads, containing queries that have a zero count in the original data. Our XSKETCH summaries consistently produced close to zero estimates with negligible error, and, therefore, we omit these results from our presentation.

Evaluation Metric. We quantify the accuracy of an XSKETCH summary based on the *average absolute relative error* of result estimates over path expressions in our workload. Given a path expression p with true result size c, the absolute relative error of the estimated count e is computed as $|e-c|/\max(c,s)$. Parameter s represents a *sanity bound* that essentially equates all zero or low counts with a default count s and thus avoids inordinately high contributions from low-count path expressions. We set this bound to the 10-percentile of the true counts in the workload (i.e., 90% of the path expressions in the workload have a true result size $\geq s$).

In our results, we report the estimation error for path expressions with value predicates, termed *Pred*, the estimation error for path expressions without value predicates, termed *No Pred*, and the overall error for both types of path expressions in the workload, termed *Overall*.

6.2 Experimental Results

XSKETCH **Performance for Branching Paths.** In this experiment, we evaluate the performance of our XSKETCHes



(a) IMDB Data Set (b) XMark Data Set

Figure 4: XSKETCH Estimation Error for Branching Paths.

for branching path expressions with range value predicates.

Figure 4 depicts the estimation error of XSKETCHes for the IMDB and XMark data sets as a function of the synopsis size. Note that, in all the graphs that we present, the estimation error at the smallest summary size corresponds to the label-split graph synopsis. Our results indicate that XSKETCHes constitute an efficient and accurate summarization method for graph-structured XML documents: even for a small space budget of 10KB-20KB, estimation error drops substantially and is lower than the error of the coarsest summary (label-split graph). The improvement is more evident for the IMDB data set where estimation error drops close to 10% using a small fraction (1%) of the space required by the "near-perfect" reference synopsis (Table 1).

In both workloads, the estimation error drops faster during the first few iterations of our construction algorithm and follows a more gradual decrease thereafter. This indicates that our XSKETCH construction algorithm captures the most important correlations early in the build process and then gradually refines the summary with respect to "less dominant" dependencies. We note that the locally optimal decisions of the greedy construction algorithm can cause minor fluctuations in the accuracy of the generated summaries, but, in general, estimation error is decreased as more storage is allocated.

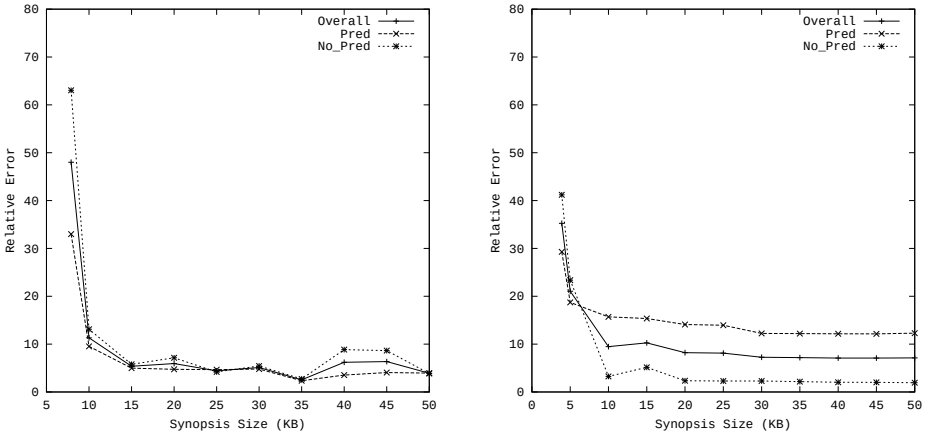
Overall, our results indicate that our XSKETCH synopses can yield accurate selectivity estimates with low space overhead and can be efficiently constructed with our forward selection algorithm.

XSKETCH **Performance for Simple Paths.** In this experiment, we focus on the simpler case of *simple* (i.e., non-branching) path queries with value predicates.

Figure 5 depicts the XSKETCH estimation error for the IMDB and XMark data sets as a function of the synopsis size. Our results clearly show that XSKETCHes can accurately estimate the selectivities of simple path expressions with value predicates over graph-structured XML data. In both datasets and for an alloted space budget of 15-20 KBytes, the estimation error drops below 10% and is significantly lower than the error of the coarsest summary, the label-split graph. In addition, this low error is achieved for a fraction of the size of the reference synopsis (Table 1). In the IMDB data set, for example, XSKETCHes drop the estimation error from 50% to 5% for a space budget equal to only 0.07% of the size of the reference synopsis. Regarding the construction process itself, we once more observe that our XSKETCH construction algorithm is able to capture the most important path and value correlations in the data during the first few steps of the build process, thus reducing the estimation error substantially for small XSKETCH sizes. Overall, XSKETCHes can efficiently capture, in limited space, the simple path structure and value distribution of the input data, thus providing accurate estimates for simple path expressions with value predicates.

7 Conclusions

In this paper, we have proposed a novel XSKETCH graph-synopsis model for XML data graphs with raw data values. Our proposed synopses exploit localized stability and value-distribution summaries to accurately capture the complex correlation patterns that can exist between and across path structure and element values in the data graph. We have also developed a systematic XSKETCH estimation framework for path expressions with value predicates that relies on appropriate statistical assumptions to compensate for the lack of detailed information in the synopsis. Finally, we have proposed an efficient XSKETCH construction algorithm based on greedy forward selection. Results from



(a) IMDB Data Set (b) XMark Data Set

Figure 5: XSKETCH Estimation Error for Simple Paths.

our XSKETCH implementation have verified the effectiveness of our approach.

References

[1] A. Aboulnaga, A.R. Alameldeen, and J.F. Naughton. "Estimating the Selectivity of XML Path Expressions for Internet Scale Applications". In *Proc. of the 27th Intl. Conf. on Very Large Data Bases*, September 2001.

[2] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler. "Extensible Markup Language (XML) 1.0 (Second Edition)". W3C Recommendation (available from `www.w3.org/TR/REC-xml/`), October 2000.

[3] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. "A Query Language and Optimization Techniques for Unstructured Data". In *Proc. of the 1996 ACM SIGMOD Intl. Conf. on Management of Data*, June 1996.

[4] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. "XQuery 1.0: An XML Query Language". W3C Working Draft 07 (available from `www.w3.org/TR/xquery/`), June 2001.

[5] Z. Chen, H.V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. Ng, and D. Srivastava. "Counting Twig Matches in a Tree". In *Proc. of the 17th Intl. Conf. on Data Engineering*, April 2001.

[6] J. Clark. "XSL Transformations (XSLT), Version 1.0". W3C Recommendation (available from `www.w3.org/TR/xslt/`), November 1999.

[7] J. Clark and S. DeRose. "XML Path Language (XPath), Version 1.0". W3C Recommendation (available from `www.w3.org/TR/xpath/`), November 1999.

[8] S. DeRose, E. Maler, and D. Orchard. "XML Linking Language (XLink), Version 1.0". W3C Recommendation (available from `www.w3.org/TR/xlink/`), June 2001.

[9] A. Deshpande, M. Garofalakis, and R. Rastogi. "Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data". In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data*, May 2001.

[10] W. Feller. *"An Introduction to Probability Theory and its Applications – Volume I".* John Wiley & Sons, 1968.

[11] B. Fox. "Discrete Optimization Via Marginal Analysis". *Management Science*, 13(3):211–216.

[12] J. Freire, J.R. Haritsa, M. Ramanath, P. Roy, and J. Siméon. "StatiX: Making XML Count". In *Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data*, June 2002.

[13] R. Goldman and J. Widom. "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases". In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, August 1997.

[14] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. "Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data". In *Proc. of the 18th Intl. Conf. on Data Engineering*, February 2002.

[15] J. McHugh and J. Widom. "Query Optimization for XML". In *Proc. of the 25th Intl. Conf. on Very Large Data Bases*, September 1999.

[16] T. Milo and D. Suciu. "Index structures for Path Expressions". In *Proc. of the 7th Intl. Conf. on Database Theory*, January 1999.

[17] J.F. Naughton, D.J. DeWitt, D. Maier, et al. "The Niagara Internet query system". *IEEE Data Eng. Bulletin*, 24(2).

[18] R. Paige and R.E. Tarjan. "Three Partition Refinement Algorithms". *SIAM Journal on Computing*, 16(6).

[19] J. Pearl. *"Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference".* Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1988.

[20] N. Polyzotis and M. Garofalakis. "Statistical Synopses for Graph-Structured XML Databases". In *Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data*, June 2002.

[21] N. Polyzotis and M. Garofalakis. "Structure and Value Synopses for XML Data Graphs". Bell Labs Tech. Memorandum, February 2002.

[22] V. Poosala and Y.E. Ioannidis. "Selectivity Estimation Without the Attribute Value Independence Assumption". In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, August 1997.

[23] J.S. Vitter and M. Wang. "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets". In *Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of Data*, May 1999.

[24] Y. Wu, J.M. Patel, and H.V. Jagadish. "Estimating Answer Sizes for XML Queries". In *Proc. of the 8th Intl. Conf. on Extending Database Technology (EDBT'02)*, March 2002.