XSKETCH Synopses for XML

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 path navigation and branching through the XML data graph 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 summarize our main results from our recent work on XSKETCHes, a novel approach to building and using statistical summaries of large XML data graphs for effective path-expression selectivity estimation. Our proposed graph-synopsis model exploits localized graph stability to accurately approximate (in limited space) the path and branching distribution in the data graph. To estimate the selectivities of complex path expressions over concise XSKETCH synopses, we develop an estimation framework that relies on appropriate statistical (uniformity and independence) assumptions to compensate for the lack of detailed distribution information. Given our estimation framework, We demonstrate that the problem of building an accuracy-optimal XSKETCH for a given amount of space is $\mathcal{NP}$-hard, and propose an efficient heuristic algorithm based on greedy forward selection. Extensive experimental results with synthetic as well as real-life data sets verify the effectiveness of our approach. To the best of our knowledge, ours is the first work to address this timely problem in the most general setting of graph-structured data and complex (branching) path expressions.

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*: the underlying data model comprises a labeled graph of *element* nodes, where each element can be either an atomic data item (i.e., raw character data) 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 rather than simply specifying how the element is to be displayed (as in HTML). Thus, XML data, like semistructured data, is graph-structured and self-describing.

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], XQL [12], and Lorel [15]. A common characteristic of all existing language proposals, is the existence of a *pattern-specification language* (like, e.g., XPath [8]) built around *path* and *subtree ("twig") expressions*. These expressions replace the traditional SQL `FROM` clause and enable 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 [8] (that lies at the core of XQuery [4] and XSLT [7], 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 and wild cards as well as branches at element nodes in the path predicated on the existence of specific sibling paths. As a concrete example, in a bibliography database the XPath expression `//author[book]/paper/sigmod/title` selects the set of all `title` data elements discovered by the label path `//author/paper/sigmod/title`, but only for `author` elements that have *at least one* `book` child (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 expressions 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 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 is a substantially different and more difficult problem than that of constructing synopses for flat, relational data (e.g., [22, 25]). Recent research studies [1, 5] 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 [3, 9]).

Recent proposals for exact and approximate *path-index structures* for XML (e.g., [13, 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 [13, 16]; and, (b) approximate indexes (e.g., the A(k)-index [13]) do not explicitly try to capture the essential statistical characteristics of the data-graph distribution.

Our Contributions. In this paper, we summarize the main results of our recent work on XSKETCHes [20], a novel approach to building and using concise statistical synopses for effectively estimating the selectivity of complex (branching) path expressions over general XML data graphs. Our proposed synopsis model exploits localized graph stability to accurately capture (in limited space) the important statistical characteristics of the path and branching distribution in the XML data graph. We develop a systematic estimation framework for approximating path-expression selectivities over concise XSKETCH synopses, and propose an efficient algorithm for XSKETCH construction. To the best of our knowledge, ours is the first work to address the timely problem of statistical synopses for XML in the most general setting of graph-structured data and complex (branching) path expressions. The key contributions of our work are summarized as follows.

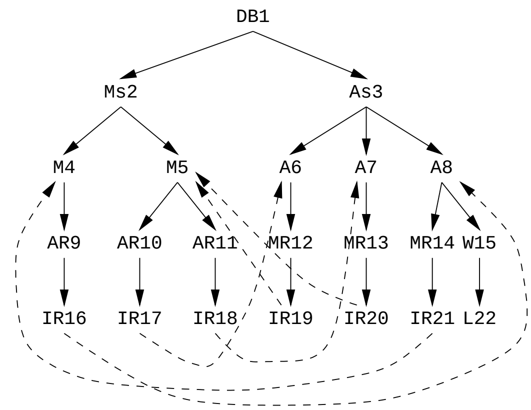
• **Definition and Systematic Estimation Framework for** XSKETCH **Synopses.** We give a formal definition of our XSKETCH synopses for XML data that exploit the concepts of localized *backward and forward graph stability* [18] to effectively explore the space between extremely coarse (but inaccurate) and extremely detailed (but large) summarizations of graph-structured data. We develop a systematic estimation framework that uses the information in the XSKETCH synopsis to parse a complex path expression and produce an approximate selectivity estimate. Like any estimation technique that uses concise data synopses (e.g., histograms [23]), our proposed framework relies on a set of appropriate statistical (uniformity and independence) assumptions to compensate for the lack of detailed distribution information.

• XSKETCH **Construction: Hardness and Efficient Heuristic Algorithm.** Constructing effective XSKETCH synopses turns out to be a difficult optimization problem: we demonstrate that the problem of building an accuracy-optimal XSKETCH for a given space budget is $\mathcal{NP}$-hard. Given this intractability result, we propose an efficient heuristic algorithm for XSKETCH construction based on greedy forward selection. Briefly, our algorithm constructs an XSKETCH synopsis by successive refinements of the label-split graph, the coarsest summary of the XML data graph. Our refinement operations act locally and attempt to capture important statistical correlations between data paths. The end result is an XSKETCH synopsis that, abstractly, is more

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

```
<DB>
    <Movies>
        <Movie ID="M1">
            <ActorRef IDREF="A1"/>
            <ActorRef IDREF="A2"/>
        </Movie>
        <Movie ID="M2">><ActorRef IDREF="A3"/></Movie>
    </Movies>
    <Actors>
        <Actor ID="A1"> <MovieRef IDREF="M1"/> </Actor>
        <Actor ID="A2"> <MovieRef IDREF="M1"/> </Actor>
        <Actor ID="A3">
            <Web Link="http://www.imdb.com/actor?A3"/>
            <MovieRef IDREF="M3"/>
        </Actor>
    </Actors>
<DB>
```

(a)



(b)

Figure 1: Example XML document (a) and XML data graph (b).

refined where correlations are stronger and less refined where data paths are independent and uniformity assumptions are valid.

• **Experimental Results Verifying the Effectiveness of** XSKETCH **Synopses.** We present the results of an extensive experimental study of XSKETCHes with several synthetic as well as real-life data sets that validate our approach. Our results show that XSKETCHes are accurate, concise synopses for general graph-structured XML data, achieving estimation errors as low as 3% for low space budgets around 30–40 KBytes. The generated summaries are built utilizing small path samples from the original document, thus ensuring the efficiency of the XSKETCH construction algorithm. We experiment with both complex and simple path expressions and show that the constructed summaries yield accurate estimates in all cases; furthermore, our XSKETCHes perform better and more consistently than earlier approaches for the simpler problem of handling simple path expressions over tree-structured XML data.

2 Background

XML Data Model. Following previous work on XML and semistructured data [13, 16], we model an XML database as a large, 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 *unique object identifier (oid)* and a *label* (assigned from some alphabet of string literals) that captures the semantics of the element. (We use `label`(v) to denote the label of node $v \in V_G$.) Edges in E_G are used to capture both the element-subelement relationships (i.e., element nesting) and the explicit element references (i.e., `id`/`idref` attributes or XLink constructs [3, 9, 13, 15]). Note that non-tree edges, such as those implemented through `id`/`idref` constructs, are an essential component and a "first-class citizen" of XML data that can be directly queried in complex path expressions, such as those allowed by the XQuery standard specification [4]. We, therefore, focus on the most general case of XML data *graphs* (rather than just trees) for the remainder of this paper.

Example 2.1 Figures 1(a,b) show an example XML document and its corresponding data graph. The document is 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. Note that we use dashed lines to show graph edges that correspond to `id-idref` relationships.

XML Query Model. A path expression $\bar{\texttt{l}}$ in XQuery defines a navigational path over the graph of the document. A path expression can be represented abstractly as a sequence of traversal steps $\texttt{step}_1/\texttt{step}_2/\ldots/\texttt{step}_n$, where each $\texttt{step}_i$ computes a new node set from the node set generated by the previous step. The node set generated by the last step is called the target set of the expression and is denoted by $\mathtt{target}(\bar{\mathtt{l}})$. Any node $u \in \mathtt{target}(\bar{\mathtt{l}})$ is said to be discovered by path expression $\bar{\mathtt{l}}$. We will use ϵ to denote the empty path expression that contains no steps.

The simplest form of an XQuery path expression is a *simple path expression* of the form $\mathtt{l}_1/\mathtt{l}_2/\ldots/\mathtt{l}_n$ where $\mathtt{l}_i$ are document labels. The target set of the path expression includes all elements u_n for which there exists a document path $u_1/u_2/\ldots/u_n$ with $\mathtt{label}(u_i) = \mathtt{l}_i$. Note that XQuery distinguishes between containment (parent/child) edges and `id-idref` edges by providing an explicit dereference operator (`=>`); in the document graph of Figure 1 for example, the path expression `Actor/MovieRef/@IDREF=>Movie` will retrieve all movie elements referenced by actors. This distinction, however, is not important in the context of our work and we drop it in order to keep the presentation simple. Therefore, we will treat `Actor/MovieRef/IDREF/Movie` as a valid path expression.

In this paper, we focus on *branching path expressions* of the form $\bar{\mathtt{l}} = \mathtt{l}_1[\bar{\mathtt{l}}^1]/\mathtt{l}_2[\bar{\mathtt{l}}^2]/\ldots/\mathtt{l}_n[\bar{\mathtt{l}}^n]$, where $\mathtt{l}_i$ are labels and $\bar{\mathtt{l}}^i$ are simple path expressions or ϵ. A branching path expression is formed from a simple path expression $\mathtt{l}_1/\ldots/\mathtt{l}_n$ by attaching the *branch predicates* $\bar{\mathtt{l}}^i$ at specific labels. Each $[\bar{\mathtt{l}}^i]$ clause represents an *existential* predicate, requiring that there exists at least one $\bar{\mathtt{l}}^i$ path at point i of the expression. Consider, for example, the simple path expression $\mathtt{l}_1/\ldots/\mathtt{l}_k/\ldots/\mathtt{l}_n$ and let $u_1/\ldots/u_k/\ldots/u_n$ be a document path that matches it. Assume that we add a branch predicate $\bar{\mathtt{l}}^k$ at position k, thus forming the branching expression $\mathtt{l}_1/\ldots/\mathtt{l}_k[\bar{\mathtt{l}}^k]/\ldots/\mathtt{l}_n$; the document path $u_1/\ldots/u_k/\ldots/u_n$ will match the new expression only if there exists at least one document path that starts from u_k and matches the simple path $\mathtt{label}(u_k)/\bar{\mathtt{l}}^k$. This is extended to the general case by requiring each node u_i to be the root of a simple path that matches the path expression $\mathtt{label}(u_i)/\bar{\mathtt{l}}^i$. Consider for example the document graph of Figure 1 and the simple path expression `Actor/MovieRef/-IDREF/Movie` that retrieves elements with id 4 and 5; if we add a `[Link]` branch on `Actor`, then the new path expression `Actor[Link]/MovieRef/IDREF/Movie` will retrieve element 4 only. Note that if all branch predicates are empty, a branching path expression degenerates to a simple path expression $\mathtt{l}_1/\cdots/\mathtt{l}_n$.

Branching path expressions represent a very common and useful case of XQuery path expressions. In general, an XQuery path expression can be more elaborate including, for example, predicates on the document order of elements or nested branch predicates. Branching path expressions, however, cover a wide class of XML query expressions most often encountered in practice and are the main focus of this paper.

3 Graph Summarization

In this section, we describe the generic model of data-graph synopsis structures that is used in this paper. We then provide a concise statement of the problem of constructing and using such graph synopses for estimating path-expression selectivities.

3.1 General Graph-Synopsis Model

Abstractly, our general model of a synopsis for an XML data graph $G = (V_G, E_G)$ is a node-labeled, directed graph structure $\mathcal{S}(G) = (V_\mathcal{S}, E_\mathcal{S})$, where each node in $v \in V_\mathcal{S}$ corresponds to a subset of identically-labeled nodes in a partitioning of V_G (termed the *extent* of v) and 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.

Definition 3.1 A graph synopsis for $G = (V_G, E_G)$ is a node-labeled, directed graph $\mathcal{S}(G) = (V_\mathcal{S}, E_\mathcal{S})$, where each node $v \in V_\mathcal{S}$ corresponds to a set $\mathtt{extent}(v) \subseteq V_G$ such that: (1) All elements in $\mathtt{extent}(v)$ have the *same label* (denoted by $\mathtt{label}(v)$, i.e., the label of the synopsis node); (2) $\cup_{v \in V_\mathcal{S}} \mathtt{extent}(v) = V_G$ and $\mathtt{extent}(u) \cap \mathtt{extent}(v) = \phi$ for each $u, v \in V_\mathcal{S}$; (3) $(u, v) \in E_\mathcal{S}$ if and only if there exist $u' \in \mathtt{extent}(u)$ and $v' \in \mathtt{extent}(v)$ such that $(u', v') \in E_G$; and, (4) Each node $v \in V_\mathcal{S}$ stores only an element count $\mathtt{count}(v) = |\mathtt{extent}(v)|$. ∎

Several recently-proposed path-index structures for XML data, including 1-indexes [16] and A(k)-indexes [13], are based on the "node-partitioning" technique described in our general graph-synopsis definition. As an example, the 1-index and the

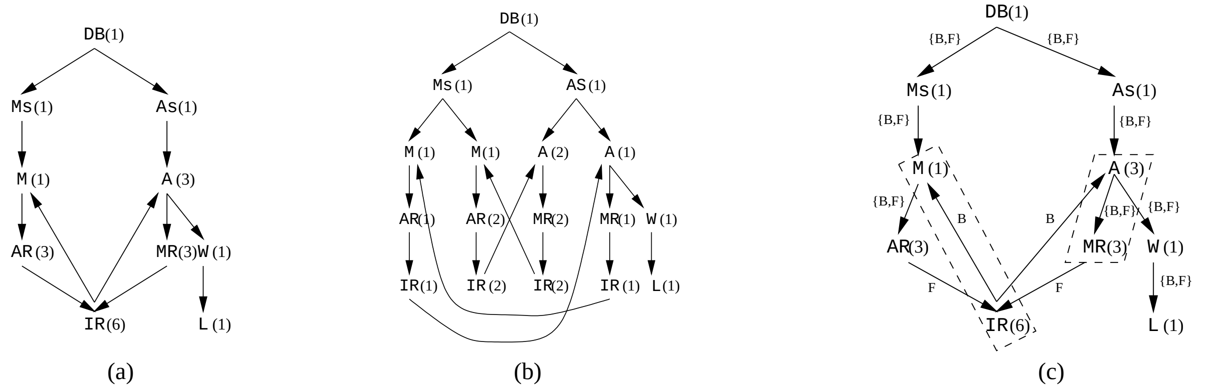


Figure 2: (a) Label-split graph; (b) B/F-bisimilar graph; and, (c) Example XSKETCH, for the XML data graph in Figure 1(b).

A(k)-index are based on the *bisimilarity* and *k-bisimilarity* partition of G, respectively. (Briefly, *bisimilarity* groups together data nodes that have identical sets of incoming label paths from the document root [16], whereas *k-bisimilarity* is based on a more relaxed rule that essentially groups data nodes based on their incoming label paths *of length at most k* [13].) However, given the stringent space limitations for our compile-time, selectivity-estimation problem, the graph synopsis can only store the extent counts (rather than the entire extents, typically stored in the aforementioned index structures). Our goal is to be able to evaluate the selectivity of complex path expressions over the data graph G based solely on a compact graph synopsis of G.

3.2 Problem Formulation

Graph-Synopsis Space Extremes. Let us now consider the two extreme points in our model space of graph synopses for estimating the selectivity of complex path queries. At one extreme, the *label-split graph* represents a very succinct but, at the same time, coarse and inaccurate synopsis of the data graph; at the other extreme, the *Backward/Forward-bisimilar (B/F-bisimilar)* graph represents an exact but prohibitively-large synopsis for branching-path selectivities.

• *The Coarsest Graph Synopsis: Label-Split Graph* ($\mathcal{S}_0(G)$). The label-split (or, 0-bisimilar [13]) graph synopsis groups data nodes into synopsis nodes based solely on their node labels; that is, all nodes in G sharing the same label are mapped onto a unique node in $\mathcal{S}_0(G)$. The label-split graph is a very succinct representation of the data graph: the number of nodes in $\mathcal{S}_0(G)$ is exactly the number of distinct labels in G. Unfortunately, the label-split graph also presents a very poor picture of the path distribution in G since, exactly due to its coarseness, it typically contains several *false paths and cycles* (that did not exist in the original data graph).

• *The Perfect Graph Synopsis: B/F-bisimilar Graph* ($\mathcal{S}_{B/F}(G)$). The Backward-bisimilarity (B-bisimilarity or, simply, bisimilarity) data-node partitioning (used, for example, in the exact 1-index for simple paths [16]) can be readily refined to capture B/F-bisimilarity. The key idea is that data nodes are mapped to the same node of $\mathcal{S}_{B/F}(G)$ only if they share the same set of *incoming and outgoing paths* in G. It is easy to verify that such a B/F-bisimilar graph synopsis is guaranteed to return exact selectivity estimates for any branching path expression. The problem, of course, is that, by its definition, a B/F-bisimilar graph is even larger (possibly, much larger) than the already problematic B-bisimilar graph.

Figure 2(a) depicts the two extremes of synopsis graphs for the example document of Figure 1. In both synopses, the figure shows, for each summary node, the label of the elements in its extent and its count attribute (the size of the extent) in parentheses. Figure 2(a) depicts the label-split graph, the coarsest summary of the document. We observe that it captures only part of the original path structure, while introducing a number of false paths, e.g., `A/MR/IR/A`). Figure 2(b) shows the B/F-bisimilar graph which represents an exact summary. Note that the summary groups together actor elements `A6, A7` since they cannot be distinguished based on their incoming and outgoing paths and the same holds for elements (`MR12, MR13`), (`IR19, IR20`), (`AR10, AR11`), and (`IR17, IR18`). Overall, the B/F-Bisimilar graph contains all the paths of the original document and no false paths, yet its size is close to that of the original data graph.

Problem Statement: Construction and Usage of Effective Graph Synopses. The label-split and B/F-bisimilar graphs lie at the two ends of the graph-synopsis spectrum: $\mathcal{S}_0(G)$ is very small and concise but typically results in extremely inaccurate estimates, whereas $\mathcal{S}_{B/F}(G)$ captures the entire path-distribution information in G accurately, but it requires too much space to be useful as an optimization-time synopsis structure. Given an amount of space (i.e., size limit) for the synopsis determined, e.g., by optimizer time and space constraints, the "most effective" graph synopsis for complex-path selectivity estimation lies somewhere between these two extremes. Of course, the notion of "effectiveness" for a graph synopsis needs to be defined based on an *estimation framework* that uses the summary information in the synopsis to parse an input complex-path expression and produce an (approximate) selectivity estimate. Given the concise and approximate nature of the synopsis, this estimation process obviously has to rely on a set of *statistical assumptions* that compensate for the lack of detailed information (similar, for example, to the intra-bucket uniformity assumptions typically made during histogram-based estimation [22, 23]). Thus, our XML-graph summarization problem comprises two important and interrelated components.

1. **[Estimation Framework for Complex Path Queries over Graph Synopses.]** Given a complex, branching path expression P and graph synopsis $\mathcal{S}(G)$ representing a statistical summary of a large XML data graph G, process P over $\mathcal{S}(G)$ (using a set of statistical assumptions) to produce an estimate for the selectivity of P over the original data graph G.
2. **[Effective Graph-Synopsis Construction.]** Given a large XML data graph G and a space budget of B bytes, build a graph synopsis $\mathcal{S}(G)$ of G that effectively minimizes the approximation error in the selectivity estimates produced based on $\mathcal{S}(G)$ (and the given estimation framework) for complex path expressions over G.

4 XSKETCH Synopses

In this section, we present our detailed definitions for XSKETCH graph synopses and describe our estimation framework for evaluating path-expression selectivities over concise XSKETCHes. We then propose a set of localized refinement operations that allow us to increase the level of accuracy of the XSKETCH for certain portions of the data graph and discuss a metric for evaluating the quality of an XSKETCH synopsis. Finally, we consider the problem of effective XSKETCH construction: even though we demonstrate that the problem is, in general, $\mathcal{NP}$-hard, we propose an efficient construction heuristic that utilizes our XSKETCH- refinement operations in a greedy, incremental fashion.

4.1 XSKETCH Concepts and Definitions

Our proposed XSKETCH synopsis structures represent specific instantiations of the general graph-synopsis model discussed in Section 3. Two key concepts underlying XSKETCHes are those of *backward-* and *forward-stability* [18].

Definition 4.1 Let V, U be sets of elements in an XML data graph G. We say that V is *backward-stable (B-stable)* with respect to U, if and only if for each $v \in V$ there exists a $u \in U$ such that the edge (u, v) is in G. Similarly, U is said to be *forward-stable (F-stable)* with respect to V, if and only if for each $u \in U$ there exists a $v \in V$ such that the edge (u, v) is in G. Given a graph synopsis $\mathcal{S}(G)$ of G, we define a node w in the synopsis to be B-stable (F-stable) with respect to another synopsis node x if and only if $\texttt{extent}(w)$ is B-stable (resp., F-stable) with respect to $\texttt{extent}(x)$. ∎

Note that, by Definitions 3.1 and 4.1, a node in a graph synopsis $\mathcal{S}(G)$ can only be B-stable (F-stable) with respect to its parent (resp., child) nodes in $\mathcal{S}(G)$. Thus, a synopsis node w is B-stable with respect to its parent x if and only all data elements mapped to w have a parent data element mapped to x in $\mathcal{S}(G)$; in other words, the number of data elements in $\texttt{extent}(w)$ that are reached by an edge from data elements in $\texttt{extent}(x)$ in G is *exactly* $\texttt{count}(w) = |\texttt{extent}(w)|$. Similarly, the F-stability condition for a synopsis node w with respect to a child node x guarantees that the count field $\texttt{count}(w)$ is an exact estimate for the number of elements in w's extent that reach by an edge in G elements that map to x in $\mathcal{S}(G)$.

Example 4.1 Consider again the example label-split graph of Figure 2(a). We observe that all elements in `extent(MR)` have a parent element in `extent(A)` and, therefore, `MR` is B-stable with respect to `A`. As a result of this stability, the `count` associated with `MR` obviously yields an exact estimate for the number of elements reached by the path expression `A/MR`; in fact, since we can show that `A` is in turn B-Stable with respect to `IR` (all `Actor` elements are reached by an `IDREF` attribute), `MR`'s `count` gives an exact estimate for the selectivity of `IR/A/MR` as well. Node `IR`, on the other hand, is not B-stable with any of its parent nodes; thus, the `count` associated with `IR` does not give an exact selectivity estimate for any path expression that ends in `IR`.

We can make similar observations about forward stabilities. All elements, for example, in `extent(MR)` have a child element in `extent(IR)` and, therefore, `MR` is F-stable with respect to `IR`; as a result, `MR`'s `count` yields an exact selectivity estimate for the path expression `MR[IR]`. Note, however, that F-stability does not say anything about the number of elements in `IR` reached by elements in `MR`, it only guarantees that the path `MR/IR` exists for all elements in `MR`.

It is interesting to note that there is an obvious connection between the concepts of stability and graph bisimilarity. Essentially, stability can be seen as a *very localized* notion of bisimilarity since, for a given synopsis node, it only considers paths of length one to/from specific child/parent node(s) in the synopsis. In fact, stability plays a central role in known efficient algorithms for computing the bisimilarity partition of a graph (e.g., [18]), where the basic operation is to *stabilize* a subset in the partition with respect to other subsets and the final bisimilarity partition is reached when every subset is stable with respect to its neighboring subsets (e.g., parent subsets for B-bisimilarity). As will become clear later in this section, it is precisely this localized character of stability that we exploit in our XSKETCHes to ensure that the limited space available for the synopsis is judiciously allocated to those portions of the data graph where our estimation assumptions are particularly inappropriate. To allow for such localized refinements at different levels of resolution, our XSKETCH synopses augment our general graph-synopsis model with a 2-bit edge label that is used to indicate possible B-stability, F-stability, or both (i.e., B/F-stability) between neighboring nodes in the synopsis.

Definition 4.2 An 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 a 2-bit indicator whose value is defined as follows: (1)$\texttt{label}(u, v) = \{\mathbf{B}\}$, if v is B-stable with respect to u; (2)$\texttt{label}(u, v) = \{\mathbf{F}\}$, if u is F-stable with respect to v; (3)$\texttt{label}(u, v) = \{\mathbf{B}, \mathbf{F}\}$, if both (1) and (2) hold; and, (4)$\texttt{label}(u, v) = \phi$ (empty), otherwise. ∎

An example XSKETCH synopsis for the XML data graph in Figure 1(b) is depicted in Figure 2(c); note that, in this specific example, the XSKETCH is simply the label-split graph of Figure 2(a) augmented with the appropriate B/F-stability labels.

4.2 Estimation Framework for XSKETCHes

We now define our estimation framework for approximating the selectivity of complex path expressions over an XML data graph G based on a compact XSKETCH synopsis $\mathcal{XS}(G)$. The following theorem establishes the basis for our XSKETCH estimation process, demonstrating that the element counts estimated at the two endpoints of a label path in $\mathcal{XS}(G)$ are guaranteed to be *exact* as long as all the edges followed in $\mathcal{XS}(G)$ satisfy the appropriate stability conditions (a fact that we have already alluded to in Example 4.1).

Theorem 4.1 Let $\mathcal{XS}(G)$ be an XSKETCH synopsis for an XML data graph G, and let $v_1, \ldots, v_n$ be a directed path in $\mathcal{XS}(G)$.

1. If $\mathbf{B} \in \texttt{label}(v_i, v_{i+1})$ for each $i = 1, \ldots, n-1$, then *all* $\texttt{count}(v_n)$ elements corresponding to v_n are discovered by the label path $\texttt{label}(v_1)/\cdots/\texttt{label}(v_n)$ starting from some node in $\texttt{extent}(v_1)$ in G.
2. If $\mathbf{F} \in \texttt{label}(v_i, v_{i+1})$ for each $i = 1, \ldots, n-1$, then *all* $\texttt{count}(v_1)$ elements corresponding to v_1 reach at least one element in $\texttt{extent}(v_n)$ by the label path $\texttt{label}(v_1)/\cdots/\texttt{label}(v_n)$ in G. ∎

Theorem 4.1 ensures that the estimates obtained from an XSKETCH are accurate as long as all the edges traversed in the synopsis while parsing the path expression satisfy the appropriate stability constraints. (Remember that an XSKETCH with all edges labeled $\{\mathbf{B}, \mathbf{F}\}$ is exactly the perfect synopsis, i.e., the B/F-bisimilar graph.) Of course, given the hard space constraints that the XSKETCH synopsis must satisfy, it is impossible to guarantee such an ideal parsing for all possible path expressions over the data graph. In the remainder of this section, we introduce an estimation framework for approximating the selectivities of complex path expressions over XSKETCHes. As with any form of estimation that uses concise synopses (e.g., histograms or wavelets), our proposed framework also relies on a set of statistical (uniformity and independence) assumptions to compensate for the lack of detailed distribution information. We describe our XSKETCH-based estimation framework below, beginning with the easier case of simple path expressions and then considering the more general case of branching paths.

4.2.1 XSKETCH Estimates for Simple Paths

Let $\overline{\mathtt{l}} = \mathtt{l}_1/\cdots/\mathtt{l}_n$ denote a simple label path over an XML data graph G. We use $\mathtt{count}(\mathtt{l}_1/\cdots/\mathtt{l}_n)$ to denote the (estimated) number of data elements that are discovered by the path $\overline{\mathtt{l}}$ in G, i.e., the *selectivity* of $\overline{\mathtt{l}}$. Consider an XSKETCH synopsis $\mathcal{XS}(G)$ of the data, and let $\overline{v} = v_1/\cdots/v_n$ be a path in $\mathcal{XS}(G)$ such that, for each i, $\mathtt{label}(v_i) = \mathtt{l}_i$; we term such an XSKETCH path $\overline{v}$ an *embedding* of the label path $\overline{\mathtt{l}}$. An element e_n in $\mathtt{extent}(v_n)$ is discovered by embedding $\overline{v}$ if there exists a document path $e_1/e_2/\ldots/e_n$ such that $e_i \in \mathtt{extent}(v_i)$. It is obvious that if an element e is discovered by embedding $\overline{v}$, then it is also discovered by the corresponding path expression $\overline{l}$. If we use $\varepsilon(\overline{\mathtt{l}})$ to denote the set of all *distinct embeddings* of $\overline{\mathtt{l}}$ in our XSKETCH synopsis (i.e., embeddings that differ in at least one node in the path), then the selectivity of $\overline{\mathtt{l}}$ is estimated by summing the selectivities over all its distinct embeddings in our XSKETCH; that is, $\mathtt{count}(\overline{\mathtt{l}}) = \sum_{\overline{v}\in\varepsilon(\overline{\mathtt{l}})} \mathtt{count}(\overline{v})$, where $\mathtt{count}(\overline{v})$ denotes the estimated number of elements discovered by embedding $\overline{v}$. (Of course, we ensure that a synopsis node cannot contribute more than its total $\mathtt{count}$ to this estimate.)

Our selectivity estimation problem, therefore, essentially reduces to estimating the $\mathtt{count}$ of the data elements discovered by each distinct embedding of the label path in $\mathcal{XS}(G)$. This $\mathtt{count}$ can be expressed as $\mathtt{count}(v_1/\cdots/v_n) = \mathtt{count}(v_n) \times f(v_1/\cdots/v_n)$, where $f(v_1/\cdots/v_n)$ denotes the estimated *fraction* (i.e., empirical probability) of elements in $\mathtt{extent}(v_n)$ that are discovered by the embedding $v_1/\cdots/v_n$. By Theorem 4.1, when the embedding $v_1/\cdots/v_n$ follows along a chain of contiguous B-stable edges, we have $\mathtt{count}(v_1/\cdots/v_n) = \mathtt{count}(v_n)$ (i.e., $f(v_1/\cdots/v_n) = 1$), and the estimate for the embedding is *exact*. We now explain how our XSKETCH estimation framework deals with "breaks" in the stability chain of the $v_1/\cdots/v_n$ embedding to approximate the fraction $f(v_1/\cdots/v_n)$.

The first step in our estimation process is to parse the embedding into a sequence of *maximal, non-overlapping* B-stable sub-paths; that is, we break the embedding $\overline{v} = v_1/\cdots/v_n$ into a collection of m sub-paths $\overline{v_1} = v_1/\cdots/v_{k_1}$, $\overline{v_2} = v_{k_1+1}/\cdots/v_{k_2}$, …, $\overline{v_m} = v_{k_{m-1}+1}/\cdots/v_{k_m}$, where $k_0 = 0 < k_1 < k_2 < \cdots < n = k_m$, and each sub-path $\overline{v_i}$ is a maximal B-stable path, i.e., $\mathbf{B} \in \mathtt{label}(v_j, v_{j+1})$ for each $j = k_{i-1}+1, \cdots, k_i - 1$ and $\mathbf{B} \notin \mathtt{label}(v_{k_i}, v_{k_i+1})$. Thus, Theorem 4.1 can be applied to give exact estimates for each individual subpath $\overline{v_i}$. To obtain an estimate for the entire embedding, we employ the well-known *Chain Rule* from probability theory [10] and the fact that $f(v_{k_{m-1}+1}/\cdots/v_{k_m}) = 1$ to rewrite the required fraction as:

$$\begin{aligned} f(v_1/\cdots/v_n) &= f(v_{k_{m-1}+1}/\cdots/v_{k_m}) \times \prod_{i=1}^{m-1} f(v_{k_{i-1}+1}/\cdots/v_{k_i+1} \mid v_{k_i+1}/\cdots/v_{k_m}) \\ &= \prod_{i=1}^{m-1} f(v_{k_{i-1}+1}/\cdots/v_{k_i+1} \mid v_{k_i+1}/\cdots/v_{k_m}), \end{aligned}$$

where $f(\overline{u}/v \mid v/\overline{w})$ denotes the *conditional probability* that a data element in a synopsis node v is discovered by the embedding $\overline{u}/v$ given that there exists an embedding $v/\overline{w}$ "rooted" at that element. Applying the Chain Rule once again for each term in the above product, we have:

$$f(v_1/\cdots/v_n) = \prod_{i=1}^{m-1} \left(f(v_{k_i}/v_{k_i+1} \mid v_{k_{i+1}}/\cdots/v_{k_m}) \times f(v_{k_{i-1}+1}/\cdots/v_{k_i} \mid v_{k_i}/\cdots/v_{k_m}) \right).$$

The fact that $v_{k_{i-1}+1}/\cdots/v_{k_i}$ is a B-stable chain in $\mathcal{XS}(G)$ guarantees that $f(v_{k_{i-1}+1}/\cdots/v_{k_i} \mid v_{k_i}/\cdots/v_{k_m}) = 1$. To estimate each of the remaining $f()$ terms in the above product, since v_{k_i+1} does not satisfy a B-stability condition with respect to its parent v_{k_i}, we make two key assumptions.

A1. [Path Independence Assumption] Given a node v in $\mathcal{XS}(G)$, the distribution of incoming paths to v is *independent* of the distribution of outgoing paths from v; thus, $f(\overline{u}/v \mid v/\overline{w}) \approx f(\overline{u}/v)$.

A2. [Backward-Edge Uniformity Assumption] Given a node v in $\mathcal{XS}(G)$, the incoming edges to v from all parent nodes u of v such that v is *not* B-stable with respect to u are *uniformly distributed* across all such parents in proportion to their `count`s; that is, if we let $\mathcal{NB}(v)$ denote the set of all "non-B-stable" parents of v then the fraction of elements in $\texttt{extent}(v)$ that are reached by $u \in \mathcal{NB}(v)$ is approximately $\frac{\texttt{count}(u)}{\sum_{w \in \mathcal{NB}(v)} \texttt{count}(w)}$.

Applying the above two assumptions, we can now estimate the overall probability as follows:

$$f(v_1/\cdots/v_n) = \prod_{i=1}^{m-1} f(v_{k_i}/v_{k_i+1} \mid v_{k_{i+1}}/\cdots/v_{k_m}) \approx \prod_{i=1}^{m-1} f(v_{k_i}/v_{k_i+1}) \approx \prod_{i=1}^{m-1} \frac{\texttt{count}(v_{k_i})}{\sum_{w \in \mathcal{NB}(v_{k_i})} \texttt{count}(w)}.$$

Example 4.2 Consider the example XSKETCH shown in Figure 2 (c) and the simple path expression `Actor/MovieRef/-IDREF/Movie`. The path expression has a single embedding `A/MR/IR/M` and, thus, $f(\texttt{Actor/MovieRef/IDREF/Movie}) = f(\texttt{A/MR/IR/M})$. The maximal parsing of `A/MR/IR/M` yields two maximal sub-paths, `A/MR` and `IR/M`, contained in dashed lines in Figure 2 (c). The fraction f, therefore, can be expressed as follows:

$$f(\texttt{A/MR/IR/M}) = f(\texttt{IR/M}) \times f(\texttt{A/MR/IR}|\texttt{IR/M}) = f(\texttt{IR/M}) \times f(\texttt{MR/IR}|\texttt{IR/M}) \times f(\texttt{A/MR}|\texttt{MR/IR/M})$$

B-stability ensures that the first and last term probability terms are equal to 1; furthermore, using Path Independence we can approximate $f(\texttt{MR/IR}|\texttt{IR/M}) \approx f(\texttt{MR/IR})$ and use Backward-Edge Uniformity to compute the overall probability:

$$f(\texttt{A/MR/IR/M}) = f(\texttt{MR/IR}) = \frac{\texttt{count}(\texttt{MR})}{\texttt{count}(\texttt{MR}) + \texttt{count}(\texttt{AR})} = 0.5$$

4.2.2 XSKETCH Estimates for Branching Paths

Let $\overline{\mathtt{l}} = \mathtt{l}_1/\cdots/\mathtt{l}_n[\mathtt{l}_{n+1}/\cdots/\mathtt{l}_{n+k}]/\mathtt{l}_{n+k+1}/\cdots/\mathtt{l}_{n+k+m}$ denote a branching label path (i.e., "twig") over an XML data graph G. As previously, we use $\texttt{count}(\overline{\mathtt{l}})$ to denote the estimated selectivity of the twig $\overline{\mathtt{l}}$ in G, and estimate this as $\sum_{\overline{v} \in \varepsilon(\overline{\mathtt{l}})} \texttt{count}(\overline{v})$, where $\varepsilon(\overline{\mathtt{l}})$ is the set of all distinct embeddings $\overline{v} = v_1/\cdots/v_n\ [v_{n+1}/\cdots/v_{n+k}]/v_{n+k+1}/\cdots/v_{n+k+m}$ in our XSKETCH synopsis $\mathcal{XS}(G)$. Once again, the selectivity count for each such twig embedding is estimated as $\texttt{count}(\overline{v}) = \texttt{count}(v_{n+k+m}) \times f(\overline{v})$, with $f(\overline{v})$ denoting the fraction of elements in $\texttt{extent}(v_{n+k+m})$ that are discovered by the embedding.

Based on Theorem 4.1, it is easy to see that $\texttt{count}(v_{n+k+m})$ is actually an exact estimate for the desired embedding selectivity as long as (1) the path chains $v_1/\cdots/v_n$ and $v_n/v_{n+k+1}/\cdots/v_{n+k+m}$ are both B-stable, and (2) the path chain $v_n/v_{n+1}/\cdots/v_{n+k}$ is F-stable. We now describe how our XSKETCH estimation process handles "breaks" in these stability chains. Using the Chain Rule, we can rewrite the required selectivity estimate for the embedding as follows:

$$\begin{aligned} f(\overline{v}) \quad = \quad & f(v_1/\cdots/v_n) \times f(\exists v_n/v_{n+1}/\cdots/v_{n+k} \mid v_1/\cdots/v_n) \times \\ & f(v_n/v_{n+k+1}/\cdots/v_{n+k+m} \mid (v_1/\cdots/v_n \wedge \exists v_n/v_{n+1}/\cdots/v_{n+k})), \end{aligned} \tag{1}$$

where the notation $f(\exists v/\overline{u})$ has been introduced to denote the fraction of data elements in the extent of v that are at the root of *at least one* $v/\overline{u}$ embedding, thus capturing the "existential" semantics of conditional branches in a branching XPath expression (Section 2). Note that the last term in the above equation captures the *correlation* between the conditional branch $v_n/v_{n+1}/\cdots/v_{n+k}$ and the path branch $v_n/v_{n+k+1}/\cdots/v_{n+k+m}$. To simplify the above expression, our estimation process makes the following independence assumption.

A3. [Branch-Independence Assumption] Given a node v reached by some path $\overline{w}/v$ in $\mathcal{XS}(G)$, outgoing paths from v are *conditionally independent* of the existence of other outgoing paths, given the originating path $\overline{w}/v$; in other words, for any two distinct outgoing paths from v, say $v/\overline{u}$ and $v/\overline{t}$, we have $f(v/\overline{t} \mid \overline{w}/v \wedge \exists v/\overline{u}) \approx f(v/\overline{t} \mid \overline{w}/v)$.

Using Assumption (A3), we can simplify Equation (1) to:

$$f(\overline{v}) \approx f(v_1/\cdots/v_n) \times f(\exists v_n/v_{n+1}/\cdots/v_{n+k} \mid v_1/\cdots/v_n) \times f(v_n/v_{n+k+1}/\cdots/v_{n+k+m} \mid v_1/\cdots/v_n). \tag{2}$$

The first and third term in the above product represent simple path estimates that can be estimated using the process described in the previous section. We now focus on the estimation of the second term in Equation (2) that involves the conditional branch of the branching path expression.

Clearly, by Theorem 4.1, if $v_n/v_{n+1}/\cdots/v_{n+k}$ is an F-stable chain in $\mathcal{XS}(G)$ then *all* elements in $\texttt{extent}(v_n)$ (regardless of the arrival path to v_n) are at the root of at least one $v_n/v_{n+1}/\cdots/v_{n+k}$ path, so the second term in (2) is exactly 1. To deal with "breaks" in the F-stability chain of $v_n/v_{n+1}/\cdots/v_{n+k}$, our estimation framework uses a methodology similar to that used for the simple-paths case. First, we parse the conditional branch $v_n/v_{n+1}/\cdots/v_{n+k}$ into a sequence of *maximal, non-overlapping* F-stable sub-paths; that is, we break it into a collection of m sub-paths $\overline{v_1} = v_n/\cdots/v_{k_1}$, $\overline{v_2} = v_{k_1+1}/\cdots/v_{k_2}$, $\ldots$, $\overline{v_m} = v_{k_{m-1}+1}/\cdots/v_{k_m}$, where $k_0 = n-1 < k_1 < k_2 < \cdots < n+k = k_m$, and each sub-path $\overline{v_i}$ is a maximal F-stable path, i.e., $\mathbf{F} \in \texttt{label}(v_j, v_{j+1})$ for each $j = k_{i-1}+1, \cdots, k_i - 1$ and $\mathbf{F} \notin \texttt{label}(v_{k_i}, v_{k_i+1})$. Next, we apply the Chain Rule and the fact that (by F-stability) $f(\exists v_n/\cdots/v_{k_1} \mid v_1/\cdots/v_n) = 1$ to rewrite the required fraction as follows:

$$
\begin{aligned}
f(\exists v_n/v_{n+1}/\cdots/v_{n+k} \mid v_1/\cdots/v_n) \;&=\; f(\exists v_n/\cdots/v_{k_1} \mid v_1/\cdots/v_n) \times \prod_{i=2}^{m} f(\exists v_{k_{i-1}}/\cdots/v_{k_i} \mid v_1/\cdots/v_n \wedge \exists v_n/\cdots/v_{k_{i-1}}) \\
&=\; \prod_{i=2}^{m} f(\exists v_{k_{i-1}}/\cdots/v_{k_i} \mid v_1/\cdots/v_n \wedge \exists v_n/\cdots/v_{k_{i-1}}).
\end{aligned}
$$

Note that the conditioning expression $v_1/\cdots/v_n \wedge \exists v_n/\cdots/v_{k_{i-1}}$ in each of the terms in the above product simply states that each node $v_{k_{i-1}}$ is reachable starting from v_1, so it can be abbreviated to simply $v_1/\cdots/v_{k_{i-1}}$. This observation and an additional application of the Chain Rule gives:

$$
\begin{aligned}
f(\exists v_n/v_{n+1}/\cdots/v_{n+k} \mid v_1/\cdots/v_n) \;&= \prod_{i=2}^{m} f(\exists v_{k_{i-1}}/\cdots/v_{k_i} \mid v_1/\cdots/v_{k_{i-1}}) \\
&= \prod_{i=2}^{m} f(\exists v_{k_{i-1}}/v_{k_{i-1}+1} \mid v_1/\cdots/v_{k_{i-1}}) \times f(\exists v_{k_{i-1}+1}/\cdots/v_{k_i} \mid v_1/\cdots/v_{k_{i-1}}/v_{k_{i-1}+1}) \\
&= \prod_{i=2}^{m} f(\exists v_{k_{i-1}}/v_{k_{i-1}+1} \mid v_1/\cdots/v_{k_{i-1}}),
\end{aligned}
$$

since $v_{k_{i-1}+1}/\cdots/v_{k_i}$ is an F-stable chain. Given the lack of the F-stability condition between $v_{k_{i-1}}$ and $v_{k_{i-1}+1}$, we utilize our Path Independence Assumption (A1) (i.e., incoming and outgoing paths at $v_{k_{i-1}}$ are independent) to write $f(\exists v_{k_{i-1}}/v_{k_{i-1}+1} \,|v_1/\cdots/v_{k_{i-1}}) \approx f(\exists v_{k_{i-1}}/v_{k_{i-1}+1})$. To estimate the term $f(\exists v_{k_{i-1}}/v_{k_{i-1}+1})$ (i.e., the fraction of data elements in the extent of $v_{k_{i-1}}$ that have a child in the extent of $v_{k_{i-1}+1}$), our XSKETCH estimation framework relies on one final statistical assumption.

A4. [Forward-Edge Uniformity Assumption] Given a node v in $\mathcal{XS}(G)$, the outgoing edges from v to all children u of v such that v is *not* F-stable with respect to u are *uniformly distributed* across all such children in proportion to their $\texttt{count}$s, and the total number of such edges is *at most* equal to the total of these $\texttt{count}$s; that is, if we let $\mathcal{NF}(v)$ be the set of all "non-F-stable" children of v and $s = \sum_{w \in \mathcal{NF}(v)} \texttt{count}(w)$, then the fraction of elements in $\texttt{extent}(v)$ that reach $u \in \mathcal{NF}(v)$ is approximately $\texttt{count}(u)/\max\{s, \texttt{count}(v)\}$.

Note that the above uniformity assumption (A4) is slightly different from its "backward" analog (A2), due to the $\max\{\}$ in the normalizing constant for the fractions. The key intuition for this differentiation is as follows. Backward-Edge Uniformity is basically estimating "up" from a given synopsis node and, since the node has parents, every node in its extent also *must* have parents in the data (root XML elements are grouped separately). Forward-Edge Uniformity tries to estimate "down" from a node, and the situation is not symmetric as not every element in the node's extent has to have children. Omitting the $\max\{\}$ from the denominator in (A4) and using only the summation over children would essentially force every element in the node's extent to have a child which seems an excessively strong assumption to make (especially for nodes with very large counts compared to their child counts).

4.3 XSKETCH Refinement Operations

Our estimation framework relies on four key statistical (uniformity and independence) assumptions (A1)-(A4) for approximating the selectivity of complex path expressions over concise XSKETCHes. Clearly, depending on the validity of these four assumptions in the actual data graph, an XSKETCH synopsis that relies on uniformity and/or independence to approximate the data-graph distribution will yield path-selectivity estimates of varying accuracy. To construct an effective XSKETCH for a given space budget, we need to be able to appropriately *refine* the synopsis structure for regions of the data graph where our uniformity and independence assumptions fail, since these regions are likely to result in high estimation errors. (Again, the relational analog would be allocating more buckets to "difficult" regions of the data distribution during histogram construction [22, 23].)

In this section, we introduce three such refinement operations for XSKETCH synopses. Our operations act locally to refine portions of the synopsis where any one of our estimation assumptions (A1)-(A4) does not hold, in order to improve estimation accuracy. At an abstract level, each refinement operation uses a partitioning criterion to *split* an XSKETCH node u into a set of new nodes $\{u_i\}$, such that $\cup_i \texttt{extent}(u_i) = \texttt{extent}(u)$ and $\texttt{extent}(u_i) \cap \texttt{extent}(u_j) = \phi$ for all $i \neq j$. Our node-partitioning criteria aim to either completely eliminate some uniformity/independence assumption(s) for the new synopsis nodes $\{u_i\}$, or to at least make such assumptions much more realistic for the new nodes $\{u_i\}$ than the old node u (similar to histogram-bucket splits). Thus, successive refinements evolve the synopsis to a larger and more precise structure. We define three different refinement operations for XSKETCH nodes, namely `b-stabilize`, `f-stabilize`, and `b-split`. We briefly describe each operation below and then discuss how they attack the uniformity and independence assumptions of our estimation framework. Due to space constraints, more details and pseudo-code descriptions for our XSKETCH refinement operations can be found in the full paper [21].

• `b-stabilize`($\mathcal{XS}(G)$, u, v): Here v is a parent of node u in the $\mathcal{XS}(G)$ synopsis, and **B** $\notin$ `label`(v, u). Clearly, when estimating the selectivity of any path embedding in $\mathcal{XS}(G)$ that contains the edge (v, u), this edge constitutes a breakpoint in the parsing of the embedding into maximal B-stable subpaths. This essentially forces the application of Path Independence (A1) and Backward-Edge Uniformity (A2) in order to estimate $f(v/u)$, i.e., the fraction of data elements in u that descend from v (Section 4.2). A `b-stabilize` operation eliminates the need for such assumptions by refining the u node into two element partitions with the same label one of which is B-stable with respect to v. In effect, `b-stabilize` separates those data elements in u that are reached through v into a new XSKETCH node u_1, and substitutes (v, u) with a new edge (v, u_1) where `label`(v, u_1) = `label`$(v, u) \cup \{$**B**$\}$ and $f(v/u_1)$=1.

• `f-stabilize`($\mathcal{XS}(G)$, u, w): The `f-stabilize` operation represents the "forward" equivalent of `b-stabilize`. Here u is a parent of node w in the $\mathcal{XS}(G)$ synopsis, and **F** $\notin$ `label`(v, u). When estimating the selectivity of a branching-path embedding whose branch contains the edge (u, w), the break in the F-stability chain mandates the use of Path Independence (A1) and Forward-Edge Uniformity (A3) in order to estimate $f(\exists u/w)$, i.e., the fraction of data elements in u that have a child in w (Section 4.2). The `f-stabilize` operation separates out exactly those elements of u in a new synopsis node u_1, so that `label`(u_1, w) = `label`$(u, w) \cup \{$**F**$\}$ and f($\exists u_1/w$)=1.

• `b-split`($\mathcal{XS}(G)$, u, $\{v_i\}$): Here u is a node in the $\mathcal{XS}(G)$ synopsis and $\{v_i\}$ is the set of parent nodes of u such that **B** $\notin$ `label`(v_i, u) (i.e., u is not B-stable with respect to any v_i). When estimating the selectivity of any path embedding in $\mathcal{XS}(G)$ that contains any of the (v_i, u) edges, the break in the B-stability chain forces the use of Backward-Edge Uniformity (A2) in order to estimate the fraction $f(v_i/u)$ of elements in u that descend from elements in v_i. Such a scenario is depicted in Figure 3(a) where the node u in the XSKETCH is shown along with an example histogram that summarizes the exact count-distribution information for the number of children in u per element in each parent v_i. The application of Backward-Edge Uniformity essentially approximates the count distribution for each parent with a single average, indicated by the dashed line in our example histogram. As our example figure shows, this may result in a poor approximation when the count-distribution of the $\{v_i\}$ parents is somewhat skewed. Our `b-split` operation tries to intelligently partition the elements in u across two new XSKETCH nodes so that (1) the set of parents $\{v_i\}$ of u is also partitioned across the two new nodes, and (2) the Backward-Edge Uniformity assumption *within each partition* gives a much more accurate approximation. A possible `b-split` for our example summary is shown in Figure 3(b) where, without loss of generality, the parent set $\{v_i\}$ has been partitioned into $\{v_1, \ldots, v_k\}$ and

$\{v_{k+1}, \ldots, v_n\}$. The key idea is that, by intelligently partitioning based on count information, `b-split` manages to produce much more uniform count histograms and, thus, substantially improve the accuracy of the average approximation within each of the resulting nodes u_1 and u_2. The benefit of the `b-split` operation, therefore, is that it does not lift Backward-Edge Uniformity but, instead, tries to make the summary fit it much better.

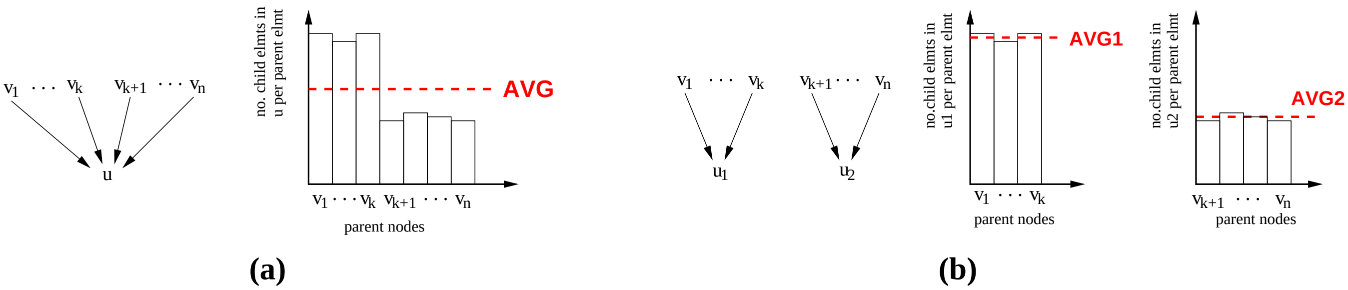


Figure 3: The `b-split` operation: (a) Original "bad" summary. (b) After applying `b-split` at node u.

Having defined our XSKETCH-refinement operations, we now discuss how they attack each of the four assumptions in our estimation framework to improve estimation accuracy.

• *Path Independence (A1)* is applied across edges that are not stable during path parsing and is lifted with the help of `b-stabilize` and `f-stabilize` operations that introduce more B-stable and F-stable edges in the summary. As an example, consider the label-split graph shown in Figure 4(a); this is the same graph as in Figure 2(a), except that we have annotated the B-stable edges. Consider the path expression `A/MR/ID/A`, which has a corresponding embedding in the summary. Since **B** $\notin$ `label`(`MR`/`ID`), the maximal parsing of the embedding yields the two subpaths `A/MR` and `IR/A` which are then combined using Path Independence; this assumption, however, is clearly false, since the overall path does not exist in the original document graph (Figure 2(a)). An application of `b-stabilize`(`IR`,`MR`) gives the summary shown in Figure 4(b) where `IR` has been split and the false paths have been eliminated. Forward stabilizations operate in a similar manner, eliminating false paths and lifting Path Independence when estimating the selectivity of branch predicates. We do not discuss this further, as it is completely symmetric to `b-stabilize`.

Figure 4: (a) Original label-split summary graph. (b) After applying `b-stabilize`(`IR, MR`).

• *Backward-Edge Uniformity (A2)* is applied at B-stability breakpoints during path parsing to estimate the percentage of elements that descend from a parent extent. A `b-stabilize` operation removes such breakpoints and, thus, lifts this assumption; furthermore, `b-split` operations, as we have shown, also directly target Backward-Edge Uniformity.

• *Forward-Edge Uniformity (A4)* is applied at F-stability breakpoints during the parsing of conditional branches. Thus, it is directly attacked by `f-stabilize` operations, which remove such breakpoints by forcing graph edges to become F-stable. For instance, consider the data graph shown in Figure 5(a) where we have two `Actor` elements, one containing 90 `Interview`

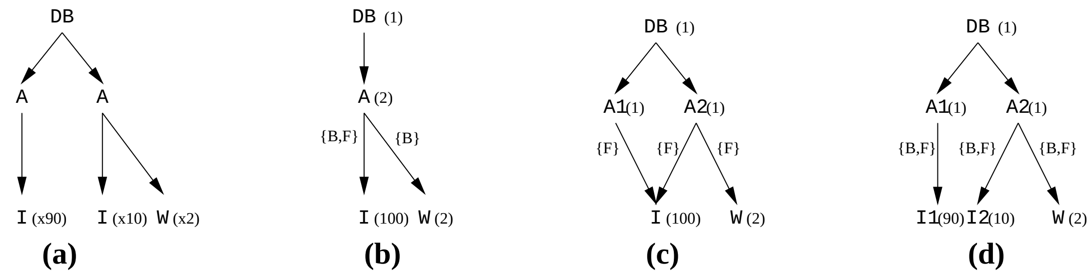


Figure 5: (a) XML data graph. (b) Label-split graph. (c) `f-stabilize` operation. (d) `b-stabilize` operation.

elements, and one containing 10 `Interview` elements and 2 `WebLink` elements. Figure 5(b) shows the label-split graph summary. Under Forward-Edge Uniformity, the estimate for path expression `A[W]` is 2, since we distribute the two `W` elements among the two `A` elements. An application of `f-stabilize(A,W)` yields the summary shown in Figure 5(c), where `A1` represents exactly those `A` elements that have at least one `W` child. Clearly, with this refinement, we no longer need to apply the uniformity assumption for `A[W]`, since the corresponding edge is now F-stable.

• *Branch Independence (A3)* is applied at the branching point(s) of a path expression in order to decouple the selectivity estimates of the branch and the simple path. Although there is no single refinement operation in our framework that targets this assumption explicitly, we can actually handle it effectively with a combination of refinements. We illustrate this point with a simple example. Consider again the data graph shown in Figure 5(a). It is obvious that there is a strong correlation between the branch on `W` and the number of `I` elements. The label-split graph of Figure 5(b) fails to capture this correlation: using our estimation assumptions, we compute the count of `A[W]/I` as 100. Applying `f-stabilize(A,W)` leads to the summary in Figure 5(c), that accurately captures the `A[W]` correlation but requires Backward-Edge Uniformity for computing `A/I`. Since this uniformity assumption is inaccurate (counts are skewed and non-uniform), we can apply a `b-stabilize(I,A1)` operation and produce the final summary of Figure 5(d). Note that the new summary does not require the Branch Independence assumption for `A[W]/I`, since all relevant edges at the branching point are stable, so the estimate is exact. Although this is a fairly simple scenario, it gives the gist of how applying different refinement operations can improve the quality of XSKETCH estimates with respect to an inaccurate Branch-Independence assumption.

4.4 Evaluating XSKETCH Quality

The previous section introduced a number of operations that refine the summary with respect to the assumptions of the estimation framework. In this section, we propose a concrete *goodness metric* for evaluating the *quality* of an XSKETCH synopsis, so that we can decide (a) whether a refined summary is "better" than the original, and (b) which refinement(s) lead to a more accurate summary.

The document graph G of an XML database essentially defines a distribution C of element counts among all possible path expressions. An XSKETCH synopsis summarizes the path structure of G and, therefore, defines another distribution C_X that approximates the true distribution. Thus, the quality of the synopsis should be evaluated on how well the approximate distribution C_X fits the true distribution C. "Goodness-of-fit" problems have been heavily researched in statistics and a number of statistical tests have been proposed for evaluating the goodness of an approximation (e.g., the chi-squared or G^2 tests [6]); the validity of these tests, however, typically relies on strict requirements (e.g., on the minimum count for each "cell" of the distribution) that are not always easy to meet in practice. Our approach is more practical and is based on the *average absolute relative error* between the estimated and real counts over a representative sample L of (branching) path expressions in $\mathcal{XS}_G$:

$$\texttt{goodness}(\mathcal{XS}) = \frac{1}{|L|} \times \sum_{\bar{\texttt{l}} \in L} \frac{|\texttt{count}_G(\bar{\texttt{l}}) - \texttt{count}_{\mathcal{XS}}(\bar{\texttt{l}})|}{max\{\texttt{count}_G(\bar{\texttt{l}}), s\}}$$

This metric represents the absolute error of the estimate relative to the true count of a path expression and is therefore a good indicator of the quality of a summary. We use the parameter s as a sanity bound in order to alleviate the inordinately high percentages of low-count path expressions and handle the case of negative path expressions, i.e., expressions that have a zero count on G. In addition, we ensure that the sample L is biased toward high-count path expressions, which are more likely to contribute significantly to the overall error. More details on the sanity bound s and the sampling method for L can be found in the full paper [21].

4.5 XSKETCH Construction Algorithm

Given the selectivity-estimation framework of Section 4.2, we now address the difficult problem of building an XSKETCH synopsis that effectively summarizes a large XML data graph within a given space budget. In many respects, XSKETCH construction is similar to other *statistical-model inference* problems, where the goal is to infer an "optimal" statistical model (e.g., Bayesian or Markov network) from an underlying data set. Most such problems have been shown to be computationally hard and can be solved exactly only by exhaustive search [19]. As the following theorem demonstrates, our effective XSKETCH construction problem is also computationally intractable; thus, it is unlikely that we can build accuracy-optimal XSKETCHes in an efficient manner.

Theorem 4.2 Let $\mathcal{Q}$ be a fixed set of path expressions to be evaluated over an XML data graph G. The problem of building an XSKETCH synopsis $\mathcal{XS}(G)$ of G with at most K nodes that minimizes the mean-squared error in the selectivity estimates of path expressions in $\mathcal{Q}$ is $\mathcal{NP}$-hard. ∎

Given the intractability of the XSKETCH construction problem[2] we now propose a computationally-efficient heuristic algorithm for building XSKETCH synopses. Our algorithm (termed BUILDXSKETCH) is based on a *greedy, forward-selection* paradigm that, essentially, starts out with the coarsest possible synopsis model for G (i.e., the Label-Split Graph $\mathcal{S}_0(G)$) and incrementally adds more complexity using the localized refinement operations discussed in Section 4.3. Our greedy 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 synopsis. In practice, rather than examining single-step refinements over all possible nodes in the XSKETCH, we only consider refinements over a representative, small *sample V* of the synopsis nodes; once again, this node sample V is biased towards nodes with high counts in order to better capture frequent portions of the data graph. The details can be found in the full paper [21].

5 Experimental Study

In this section we present the results of an extensive empirical study that we have conducted using our novel XSKETCH synopses proposed in this paper. The objective of this study is twofold: (1) to establish the effectiveness of XSKETCHes as summaries for graph-structured XML documents, and (2) to demonstrate the benefits of our methodology compared to earlier approaches for the estimation of simple path expressions over tree XML documents. Our experiments consider a wide range of queries over synthetic and real-life data and our findings can be summarized as follows.

• **Improved Estimates.** In all the data sets that we have considered, XSKETCHes produce accurate estimates with low error for both positive and negative (i.e., zero count) queries. Our experiments also demonstrate that XSKETCHes are efficient summaries for tree XML documents, outperforming earlier approaches for simple path expression estimation.

• **Reduced space requirements.** XSKETCHes capture the most important path and branching correlations in the underlying data set using only a small fraction (1% to 5%) of the space required by the perfect B/F-bisimilar summary and reduce estimation errors substantially compared to the (crude) label-split graph.

• **Small sample sizes are effective.** Effective XSKETCH synopses can be efficiently constructed using only a small sample

[2]Even though our reduction uses the mean-squared error metric, we conjecture that the problem remains $\mathcal{NP}$-hard for other error metrics as well.

of the XML document paths (1000 paths) and a small sample of the summary nodes (10% of the total number) for synopsis evaluation and refinement. The quality of the resulting synopsis improves with the size of the sample, but our experiments show that, even with a very small number of paths and nodes, XSKETCHes are able to reduce the estimation error substantially.

Thus, our experimental results validate the thesis of this paper that XSKETCHes are efficient summary structures for accurately estimating the selectivity of complex path expressions over general, graph-structured XML databases.

5.1 Experimental Testbed and Methodology

Techniques. We consider two estimation techniques in our study.

• XSKETCHes: We have implemented the XSKETCH framework that we have presented in this paper. We use the forward selection algorithm for constructing our synopses with different settings for the node sample (V) and path sample (P) parameters. Specifically, we vary V between 1%, 5%, and 10% of the total number of nodes in the summary, and we use a sample P of 100, 500, and 1000 paths.

• Markov Tables (MTs): Aboulnaga et al. [1] introduced a number of summary structures for estimating the selectivity of simple path expressions over *tree-structured* XML documents. We compare XSKETCHes against 2nd order Suffix-* Markov Tables, that were shown to be the most accurate synopses on the real-life data sets tested in [1].

Data Sets. We use two real-life data sets and one synthetic data set in our experiments.

• IMDB: This is a real-life, graph-structured data set from the Internet Movie Database (`www.imdb.com`). It is generated by crawling the IMDB database and selecting a subset of the nodes until a desired size is reached. The key characteristic of this data set is a high number of cycles and it represents highly irregular data.

• XMark: This a synthetic, graph-structured data set from the XML Benchmark [24], containing information on the activities of an E-commerce web site. We use the benchmark data generator with a 0.1 scale to generate the corresponding document (10MB in size). The data does not contain cycles but the high number of `idref` references make it quite irregular.

• DBLP: This is a real-life, tree-structured data set that contains bibliographic data from the DB&LP database (`www.-informatik.uni-trier.de/~ley/db`). It does not contain any cycles and is relatively regular in structure.

	IMDB	XMark	DBLP
Number of elements	102,755	206,131	1,399,766
Nodes in Label-Split Graph	123	84	601
Nodes in B/F-Bisimilar Graph	49,181	197,508	5,884
Size of Label-Split Graph	5.7 KB	3.7 KB	17 KB
Size of B-Bisimilar Graph	436 KB	1.8 MB	117 KB
Size of B/F-Bisimilar Graph	1.5 MB	6.2 MB	117 KB

Table 1: Characteristics of the Three Data Sets.

	IMDB	XMark	DBLP
Simple Paths	1,125	511	2,614
Branching Paths	1,351	1,940	-

Table 2: Average Query Result Sizes.

Table 1 summarizes the main characteristics of the data sets in terms of the sizes of the corresponding B-bisimilar graph, B/F-bisimilar graph, and label-split graph. As expected, the DBLP data set has the smallest perfect summary since it is the most regular data set of the three; in addition, the B-bisimilar graph for this data set is identical to the B/F-bisimilar graph since we are dealing with a tree-structured document. IMDB and XMark, on the other hand, have large perfect summaries thus motivating the need for concise synopses. Note that the sizes reported do not include the space needed to store the actual text of the element labels; each label is hashed to a unique integer and the mapping is stored in a separate structure that is not part of the summary.

Query Workload. We evaluate the accuracy of the generated summaries against a workload consisting of 1000 *positive* path expressions, i.e., paths that have a non-zero count. The workload is created by sampling the B/F-bisimilar graph of each data set and generating both simple and branching path expressions. More specifically, we generate 600 simple path expressions, 300 branching expressions with one branch of length 1 to 3, and 100 expressions with two branches, each one with length between

1 and 3. This mix is representative of what we expect to see in a real-life workload of queries against an XML database: most queries will typically involve simple path expressions, a smaller number will contain one-branch path expressions, and a few will reference more complex paths. In the experiments where we consider summaries for estimating simple paths only, we use a modified workload that consists entirely of simple path expressions. In all cases, the length of the path expression (without branches) is distributed between 2 and 5 and the sample is biased toward high counts in the B/F-bisimilar graph nodes. As a result, the generated paths follow the distribution of the data, with high-count elements being referenced more frequently in the query set. Table 2 summarizes the average result size for our query workloads for all the data sets considered in our study. Of course, the sample of path expressions in our query workloads is completely unrelated to the samples that our XSKETCH construction algorithm uses during the building of the synopses.

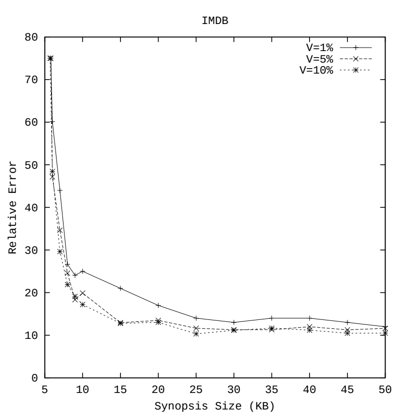
We have also experimented with a *negative* workload, i.e., path expressions that do not discover any elements in the data graph. Our results have shown that XSKETCH summaries consistently produce close to zero estimates with negligible error and therefore we omit this workload from our presentation.

Answer-Quality Metrics. We evaluate the constructed summaries using the quality metric that we introduced in Section 4.4. This metric represents the average absolute relative error combined with a sanity bound s in order to avoid the effect of inordinately high relative-error contributions from low-count queries. In our evaluation, we set the sanity bound s to the 10-percentile of the true counts of path expressions in the workload.

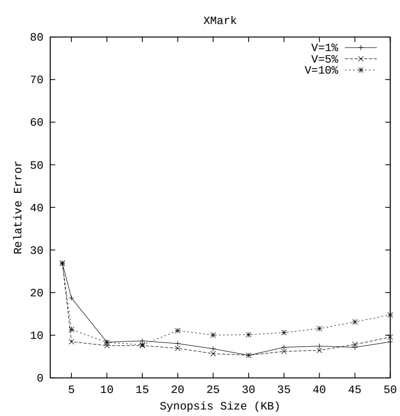
5.2 Experimental Results

XSKETCH **Performance for Branching Paths.** This set of experiments evaluates the estimation accuracy of XSKETCHes for branching path expressions over graph-structured XML data. We vary three key XSKETCH parameters, namely the allotted space budget for the synopsis, the size of the node sample V, and the size of the path sample P used during XSKETCH construction.

In the first set of experiments, we study the effect of the size of the node sample V on XSKETCH quality. The V parameter determines the number of XSKETCH-node refinements that the construction algorithm considers at each step and essentially defines the portion of the construction search space that our forward-selection algorithm examines. We keep the size of the path sample P fixed to 1000, and vary V as a percentage of the total number of nodes in the (current) synopsis. We experiment with three values for the node-sample size: 1%, 5%, and 10%.



(a) IMDB



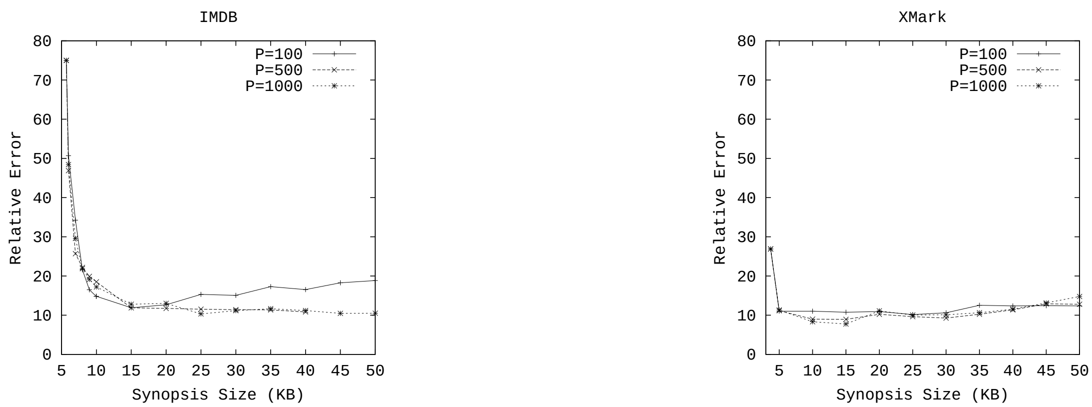
(b) XMark

Figure 6: Estimation error varying V: (a) IMDB data set, (b) XMark data set.

Figure 6 depicts the estimation error of XSKETCHes for the IMDB and XMark data sets as a function of the synopsis size for the three different values of the node-sample size V. 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 clearly indicate that XSKETCHes constitute an efficient and accurate summarization method for graph-structured documents. Even with a small node-sample size of V=10%

for synopsis refinements and an allotted space of 25–30 KBytes, the estimation error drops to 10% and is substantially lower than the error of the coarsest summary, the label-split graph. This is most noticeable in the IMDB data set that is the most irregular of the two: the starting summary yields an average error of 70%, which rapidly drops to 10% after the first iterations of the construction algorithm. Furthermore, XSKETCHes achieve a low estimation error while using only a very small fraction of the space required by the corresponding "perfect" B/F-bisimilar graph for both data sets (Table 1). Overall, our XSKETCH synopses yield accurate selectivity estimates with low space overhead and can be efficiently constructed using only a small sample of the nodes for refinement.

The next set of experiments studies the effect of the size of the path sample P on the estimation accuracy of the constructed XSKETCH summary. The P parameter determines the number of sampled paths against which each refinement is evaluated; in essence, it represents the size of the training set of our forward-selection construction algorithm and can affect the quality of the generated synopsis. We keep the node-sample size V fixed to 10% of the total nodes in the summary and we experiment with three sizes for the path-sample size P: 100, 500, and 1000.



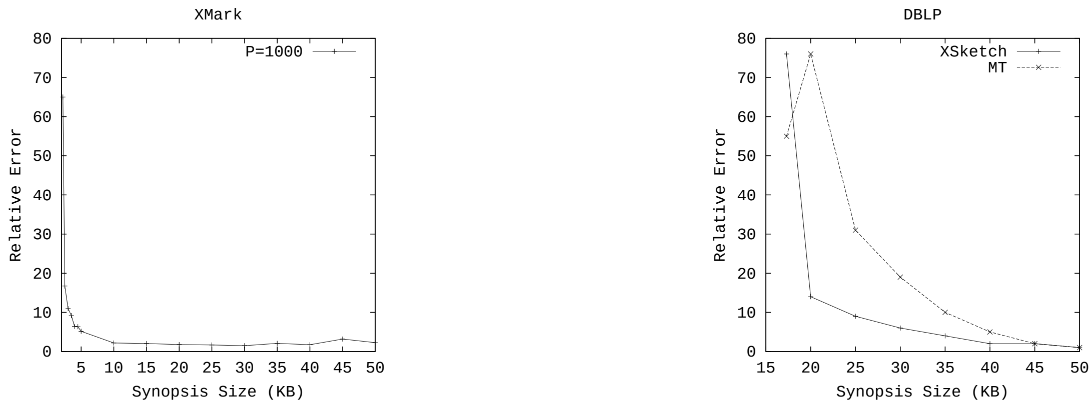
(a) IMDB (b) XMark

Figure 7: Estimation error varying P: (a) IMDB data set, (b) XMark data set.

Figure 7 depicts the XSKETCH estimation error for the IMDB and XMark data sets as a function of the synopsis size for the three different values of the path-sample size P. The results clearly show that, in all cases, our XSKETCH construction algorithm captures effectively the important statistical characteristics of the underlying path and branch distribution in the data. Once again, the XSKETCH estimation error is reduced significantly during the first few iterations and improves gradually thereafter. We observe that the convergence of our algorithm is faster for larger path sample sizes, especially in the case of the more irregular IMDB data set, where the small sample size of 100 exhibits a more "unstable" behavior with increased errors. In effect, the limited size of the training set prevents the algorithm from identifying the important correlations in the structure of the graph and leads to the application of less effective refinements at each step. Still, the overall conclusion is that reasonable path-sample sizes are adequate to construct accurate synopses, and an XSKETCH that occupies only 25–30 KBytes (0.5%–2% of the space required by the corresponding "perfect" B/F-bisimilar graph) is sufficient to guarantee low selectivity-estimation errors.

In both sets of experiments reported above, we observe that the estimation error drops substantially during the first iterations of our XSKETCH construction algorithm, followed by a gradual but less steep decrease afterwards. It is evident that XSKETCH construction captures the most important path and branching correlations early in the build process and then refines the synopsis with respect to "less dominant" structural dependencies. We also note a number of minor fluctuations in the estimation error as the summary size grows; this is due to the fact that we evaluate the summary against a different path sample than the one used by the construction algorithm. Still, the overall picture is that estimation error is reduced for larger summary sizes.

XSKETCH **Performance for Simple Paths.** In this set of experiments, we focus on the simpler problem of estimating the selectivity for *simple* (i.e., non-branching) path queries. Since our synopsis does not need to handle branches in path expressions, our estimation framework no longer requires the corresponding assumptions, namely Branch Independence and Forward-Edge Uniformity; as a result, we do not need to consider `f-stabilize` operations in our XSKETCH construction algorithm, since they are not relevant to the estimation of simple path expressions. We set the path-sample size $P = 1000$ and the node-sample size $V = 10\%$, and we evaluate the quality of our XSKETCH synopses against a workload of simple path queries. In the interest of space, we only present our results for the XMark data set; our findings for the IMDB data set are similar and can be found in the full version of this paper [21].



(a) XMark (b) DBLP

Figure 8: Estimation error for simple path expressions: (a) XMark data set, (b) DBLP data set.

Figure 8 (a) depicts the XSKETCH estimation error as a function of the synopsis size for a query workload of simple path expressions on the XMark data set. Our results show that XSKETCHes are very efficient and accurate in estimating the selectivities of simple path expressions over graph-structured XML data. Using a small space budget of 30-50 KBytes, XSKETCHes yield a negligible estimation error of about 1%. It is interesting to observe that the initial estimation error of the label-split graph (75%) is higher than in the case of branching path expressions (26%). This is mainly due to the additional selectivity factors that branches introduce in the estimation formulas; as a result, the estimated number of elements for each expression drops (since the number of small multiplicative factors increases) and the observed relative error is not as large. Nevertheless, our XSKETCH construction algorithm is once again able to capture the most important path correlations in the data during the first few steps of the build process, thus reducing the estimation error substantially for small XSKETCH sizes.

In our final experiment, we compare our XSKETCH synopses against the Markov Table (MT) summaries of Aboulnaga et al. [1] on simple path expressions over *tree-structured* data. We use the 2nd Order Suffix-* Markov Table that was shown to have the best performance on the real-life data sets tested in [1], and we compare against XSKETCH summaries constructed with the path-sample and node-sample sizes fixed at $P = 1000$ and $V = 1\%$, respectively. Figure 8 (b) shows the estimation error of the two methods as a function of the synopsis size on the (tree-structured) DBLP data set. XSKETCHes are more efficient in capturing the key structural dependencies using the limited storage space and consistently provide estimates with very low error after a few iterations of the construction algorithm: with a small space budget of 30KBytes, XSKETCHes have an estimation error of 6% compared to 19% for MT summaries. The construction of MT summaries is based on the summarization of low-frequency paths with special *-paths, that approximate the pruned frequencies with an average and thus enforce a uniformity assumption. In addition, the MT-estimation model is based on a "Markovian memory" assumption in order to compute the count of a path from the counts of shorter paths (of length up to 2). The MT-construction algorithm , however, prunes paths in a greedy fashion based solely on their frequency and does not consider the validity of the two assumptions with respect to the underlying path distribution; as a result, it can make sub-optimal decisions that lead to less accurate summaries even when more storage space is allocated. This is evident in our results, where the estimation error for MT summaries suddenly jumps when the synopsis size increases from 17KB to 20KB. Our approach, on the other hand, is more methodical as XSKETCH construction directly takes into account the structural dependencies that exist in the paths of the XML data graph.

6 Extensions: Incorporating Value-Distribution Information

Thus far, we have focused on the problem of producing a synopsis for the *structure* of an XML data graph, in order to accurately estimate the number of distinct nodes/elements that are reached through a branching label-path expression. In general, the elements of an XML data graph can also have associated *values*, and path expressions can involve predicates on these element values. Figure 9 (a) depicts an example document graph where each **M**`ovie` element has a **B**`oxOffice` child element that contains the box-office sales of the movie. Each value is represented in the document graph as a separate node that is pointed to by the corresponding element node. A possible path expression on this graph, for example, could be `Actor/MovieRef/IDREF/Movie[BoxOffice>100000]`, selecting all `Movie` elements reachable by `Actor/MovieRef/IDREF` that have a `BoxOffice` element with a value greater than 100,000. In this section, we present some preliminary ideas on how our XSKETCH-based methodology can be extended to handle complex path expressions that involve such value predicates.

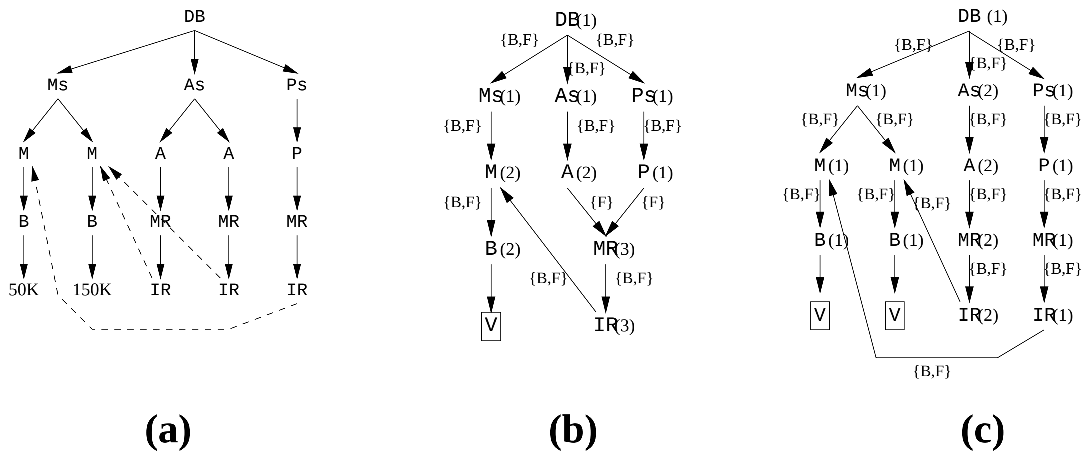


Figure 9: Capturing value-distribution information in XSKETCHes.

Our proposed approach to the problem is similar in concept with our methodology for capturing correlations between paths and branches. More specifically, we introduce special nodes in the XSKETCH summary that capture the *distribution of values* under the elements of a single XSKETCH node. The nature of these special value-distribution nodes depends on the specific data type of the element values; for example, value-distribution nodes can represent histograms [22] for numeric values or pruned-count suffix trees [2] for string values. In addition, these value summaries can be multi-dimensional in the general case, thus capturing the correlation between values under the parent summary node and values in different parts of the document graph. Returning to our example, Figure 9 (b) shows the label split graph where a special node `V` represents a single-dimensional value summary (e.g., a histogram) of sales under the `BoxOffice` elements. This node implies a single distribution of values for all `BoxOffice` elements, regardless of the specific label paths that reach them and independent of other values in the document. To some extent, therefore, we are applying a value-path independence assumption for values in order to compensate for the lack of other information. The idea is to define the appropriate operations which the construction algorithm can use to refine the synopsis with respect to this new set of statistical assumptions whenever they are not valid. As an example, assume that `BoxOffice` sales depend on whether the corresponding movie is reachable from `Actor`s or `Producer`s. We can apply a sequence of `b-stabilize` operations on nodes `MR`, `IR`, `M`, and `B`, in order to split the `B` elements based on the incoming path (i.e., from `Actor` or `Producer`) and thus force the split of the original distribution node in two new value-nodes, one for each new `B` node (Figure 9(c)). The two new distribution nodes summarize `BoxOffice` sales under different incoming paths and the resulting XSKETCH synopsis captures the correlation between values and paths more accurately.

7 Conclusions

In this paper, we have presented the key concepts and algorithms underlying XSKETCHes, a novel class of statistical synopsis structures for general, graph-structured XML data. Our XSKETCH synopses exploit localized graph stability to accurately capture, in limited space, the key correlations in the path and branching distribution of large XML data graphs. We have developed a systematic estimation framework for XSKETCHes that relies on well-founded assumptions to compensate for the lack of detailed information in our synopses. We have also demonstrated that the problem of building an accuracy-optimal XSKETCH is $\mathcal{NP}$-hard, and we have proposed an efficient construction heuristic that employs localized refinement operations to evolve an initial, coarse summary to a larger and more accurate synopsis. Extensive experimental results with synthetic and real-life data sets 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*, Sept. 2001.

[2] H.V. Jagadish, Raymond T. Ng, and Divesh Srivastava. "Substring Selectivity Estimation". In *Proc. of the 18th ACM Symposium on Principles of Database Systems*, May 1999.

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

[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 (`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] R. Christensen. *"Log-Linear Models and Logistic Regression"*. Springer-Verlag New York, Inc. (Springer Series in Statistics), 1997.

[7] J. Clark. "XSL Transformations (XSLT), Version 1.0". W3C Recom- mendation (`www.w3.org/TR/xslt/`), November 1999.

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

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

[10] W. Feller. *"An Introduction to Probability Theory and its Applications – Vol. I"*. John Wiley & Sons, 1968. (3rd Edition).

[11] B. Fox. "Discrete Optimization Via Marginal Analysis". *Management Science*, 13(3), November 1966.

[12] Y. Kanemasa H. Ishikawa, K. Kubota. XQL: A query language for xml data. In *QL98 - The Query Languages Workshop*, 1998.

[13] 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*, Feb. 2002.

[14] F.M. Malvestuto. "Approximating Discrete Probability Distributions with Decomposable Models". *IEEE Trans. on Systems, Man, and Cybernetics*, 21(5), September 1991.

[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 Engineering Bulletin*, 24(2), 2001.

[18] R. Paige and R.E. Tarjan. "Three Partition Refinement Algorithms". *SIAM Journal on Computing*, 16(6), December 1987.

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

[20] N. Polyzotis, 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. "Statistical Synopses for Graph- Structured XML Databases". Bell Labs Tech. Memo., Dec. 2001.

[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] V. Poosala, Y.E. Ioannidis, P.J. Haas, and E.J. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". In *Proc. of the 1996 ACM SIGMOD Intl. Conf. on Management of Data*, June 1996.

[24] A.R. Schmidt, F. Waas, M.L. Kersten, D. Florescu, I. Manolescu, M.J. Carey, and R. Busse. "The XML Benchmark Project". Tech. Report, CWI, 2001.

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