Twig Query Processing over Graph-Structured XML Data

Zografoula Vagena, Mirella M. Moro, Vassilis J. Tsotras

WebDB 2004
Outline

- Motivation
- Reachability in general graphs
- Reachability in DAGs
- Twig Processing in Digraphs
- Performance Evaluation
- Conclusions
Motivation

- Graph structures play an important role in XML
  - XML data modeled as graphs
  - Structural summaries have graph structures
- Tree Pattern Queries (twigs) have been proposed to enable selection of elements based on their structural characteristics
Example (1): XML data

Twig Query Processing over
Graph-Structured Data
Example (2): Structural Index

Twig Query Processing over Graph-Structured Data
Twig Query Example (XPath semantics)

- For each bib element $e_1$:
  - check if there exists an article element $a_1$, descendant of $e_1$, that has at least one title descendant
  - if no abort
  - otherwise, identify all author elements that are descendants of $e_1$

- For each such author element $au_1$:
  - identify and return all name elements that are descendants of $au_1$
Problem

• State of the art query processing techniques, applied on the data directly have assumed tree structures and are not applicable for graphs

• Navigation-based query processing techniques proposed for structural summaries, are optimized for path queries and for main memory structures
Our Solution

- Augment each node with information that identifies its place within an XML document, if possible in constant time
- Explore this information to design efficient algorithms that holistically answer the query
Labeling Scheme for Digraphs (1)

*Descendant Relationship* between two nodes is satisfied if there is a directed path between the nodes.

Transitive Closure could be utilized. However, too space consuming.
2-hop cover

Let $G = (V, E)$ be a directed graph. For every $u, v \in V$, let $P_{uv}$ be a collection of paths from $u$ to $v$ and $P = \{P_{uv}\}$. A hop is defined to be a pair $(h, u)$, where $h$ is a path in $G$ and $u \in V$ is one of the endpoints of $h$.

A collection of hops $H$ is said to be a 2-hop cover of $P$ if for every $u, v \in V$, such that $P_{uv} \neq \emptyset$, there is a path $p \in P_{uv}$ and two hops $(h_1, u) \in H$ and $(h_2, v) \in H$, such that $p = h_1 h_2$, i.e. $p$ is the concatenation of $h_1$ and $h_2$.

The size of the cover $H$, is equal to the number of hops in $H$. 

Twig Query Processing over Graph-Structured Data

[95x509]Labeling Scheme for Digraphs (2)
[746x17]10
[8x17]7/10/07
Labeling Scheme for Digraphs (3)

2-hop labels

Let $G = (V, E)$ be a directed graph. A 2-hop reachability labeling of $G$ assigns to each vertex $v \in V$ a label $L(v) = (L_{in}(v), L_{out}(v))$, such that $L_{in}(v), L_{out}(v) \subseteq V$ and there is a path from every $x \in L_{in}$ to $v$ and from $v$ to every $y \in L_{out}$. Furthermore, for any two vertices $u, v \in V$, we should have:

$$v \rightarrow u \text{ iff } L_{out}(v) \cap L_{in}(u) \neq \emptyset$$

The labeling size is defined to be $\sum_{v \in V} |L_{in}(v)| + |L_{in}(v)|$. 

Twig Query Processing over Graph-Structured Data
Labeling Scheme for Digraphs (4)

- The set of paths can be the set of all shortest paths between each pair of nodes in the graph.
- Having a 2-hop cover for a graph, the 2-hop labels for the nodes of the graph can be created by mapping a hop with handle \( v \) to an item in the label of \( v \).
- Having two nodes \( u, v \) with labels \((L_{\text{in}}(u), L_{\text{out}}(u))\) and \((L_{\text{in}}(u), L_{\text{out}}(u))\), reachability can be checked efficiently either in a sort or in a hash fashion.
Labeling Scheme for DAGs (1)

- With 2-hop covers, the time to identify structural relationships between nodes is not constant anymore.
- In case the graph is a DAG, more space efficient labeling schemes exist.
Labeling Scheme for DAGs (2)

Theorem: Let $G = (V, E)$ be a planar, directed and acyclic graph with one source and one sink. For each node $v \in V$, a label $L$ consisting of two numbers $a_v$ and $b_v$ is assigned to it. Then for every pair of nodes $u$ and $v \in V$ with labels $(a_u, b_u)$ and $(a_v, b_v)$ the following holds:

$$u \rightarrow v \text{ iff } a_u < a_v \text{ and } b_u < b_v$$

One can always find such a label.
Labeling Scheme for DAGs (3)

Construction Algorithm

1. initialize counter to \( n + 1 \)

2. perform left depth first traversal
   - push node into stack when first visited
   - pop node when all starting from edges have been examined
   - assign value of counter to node when popping it and increment counter

3. reinitialize counter to \( n + 1 \)

4. repeat 2 for right depth first traversal
Labeling Scheme for DAGs: Example

Data Graph

Right Depth Traversal

Left Depth Traversal

Twig Query Processing over Graph-Structured Data
Twig Processing in XML DAGs (1)

1. Accesses nodes according to their sequence within a depth first traversal of the tree

2. Employs an FSM to identify matching nodes
   - divide twig query into constituent path queries
   - share common prefixes to decrease computation
   - convert query to equivalent DFA
   - use DFA to identify path instances

3. Merges path instances on their common prefixes to create twig instances
   - would benefit if path instances from previous phase were produced in root-to-leaf order
Twig Processing in XML DAGs (2)

- A runtime stack is utilized to buffer previous states of the DFA and to allow backtracking to those states whenever possible.

- An element stack is associated with each query node to buffer document nodes of the same tag:
  - each node in the stack points to its nearest source in the parent stack. That way, the set of stack creates a compact encoding of partial and total results.
  - additional buffering is introduced to produce results in root-to-leaf order.
Twig Processing Example (1)

(a) Query paths and simplified DFA

(b) Document

Twig Query Processing over Graph-Structured Data
Twig Processing Example (2)

(c) Runtime stack
(d) Element stacks, left path
(e) Results

Twig Query Processing over
Graph-Structured Data
Experimental Setup (1)

Projected Xmark DTD

Twig Query Processing over Graph-Structured Data
Experimental Setup (2)

Queries

Twig Query Processing over Graph-Structured Data
Experimental Results

Query Performance

Twig Query Processing over Graph-Structured Data
Conclusions

• Twig Processing techniques for graph structured XML data using:
  – 2-hop labels for reachability in general digraphs
  – 2-node labels for reachability in DAGs

• Preliminary Performance Evaluation

• Future work:
  – size of labels in real world, disk-based, implementation of graph structures
  – specialized labels for other common graph structures