Matching Tree Patterns on Partial-trees
Optimizing Tree-Pattern Matching

Shachar Harussi
Supervision of Prof. Amir Averbuch

September 1, 2011
1 Motivation: Graph querying

2 Background: tree patterns

3 Partial trees - holistic divide and concur
Outline

1. Motivation: Graph querying

2. Background: tree patterns

3. Partial trees - holistic divide and concur
Motivation

Everything is a graph.
Motivation (cont.)

We need to query everything.

![Diagram showing tree patterns and partial trees with nodes and edges.](image-url)
Lets take a picture

- A picture is a graph.
Lets take a picture

- A picture is a graph.
Holistic approach

Given a graph pattern.
Holistic approach

- Given a graph pattern.
- And a graph data,
Holistic approach

- Given a graph pattern.
- And a graph data,
- The solution is $O(O)$. 

Shachar Harussi  Supervision of Prof. Amir Averbuch
Matching Tree Patterns on Partial-trees
But Holistic is hard

- **A Problem:** Holistic pattern matching is NP-hard.
- Even subgraph isomorphism problem [8] is hard.
But Holistic is hard

- A Problem: Holistic pattern matching is NP-hard.
- Even subgraph isomorphism problem [8] is hard.
- A Solution: divide and concur.
Local is easy

- Divide the pattern into local patterns $P_i$. 
Local is easy

- Divide the pattern a local patterns $P_i$. 
- And local data $D_i$. 

Shachar Harussi  Supervision of Prof. Amir Averbuch  
Matching Tree Patterns on Partial-trees
Local is easy

- Divide the pattern a local patterns $P_i$.
- And local data $D_i$.
- Partial solutions $O O$.
- Strings matching is fast $O(P_i \times D_i)$.
Local is easy

- **Divide** the pattern a
  local patterns $P_i$,
- And local data $D_i$.
- Partial solutions $O\ O$.
- Strings matching is fast
  $O(P_i \times D_i)$
- Join (**Concur**) Final
  solution $O\ O$. 

Shachar Harussi  Supervision of Prof. Amir Averbuch
Matching Tree Patterns on Partial-trees
So local approach is perfect?

The answer is NO
So local approach is perfect?

- The answer is **NO**
- The concur is a Pyrrhic victory - i.e. the join costs.
- But let's focus on trees.
Outline

1. Motivation: Graph querying
2. Background: tree patterns
3. Partial trees - holistic divide and concur
Tree Data Model

‘Tree pattern’ is a tree
The local approach problem

An example

**Query:** `/a/b[/c]/d`
The local approach problem

an example

query: ‘/a/b[/c]/d’

1. path1(’/a/b/c’):
   \((v_1, v_2, v_3), (v_1, v_4, v_5)\)
The local approach problem

an example

query: ‘/a/b[/c]/d’

1. path1(‘/a/b/c’):
   \((v_1, v_2, v_3), (v_1, v_4, v_5)\)

2. path2(‘/a/b/d’):
   \((v_1, v_4, v_6), (v_1, v_7, v_8)\)
The local approach problem
an example

query: '/a/b[/c]/d'
1. path1(''/a/b/c''):
   (v_{1}, v_{2}, v_{3}), (v_{1}, v_{4}, v_{5})
2. path2(''/a/b/d''):
   (v_{1}, v_{4}, v_{6}), (v_{1}, v_{7}, v_{8})
3. joins:
   (v_{1}, v_{2}, v_{3}, )
   (v_{1}, v_{4}, v_{5}, )
   (v_{1}, v_{4}, , v_{6})
   (v_{1}, v_{7}, , v_{8})
The local approach problem

an example

query: ‘/a/b[/c]/d’
1. **path1** (‘/a/b/c’):
   \((v_1, v_2, v_3), (v_1, v_4, v_5)\)
2. **path2** (‘/a/b/d’):
   \((v_1, v_4, v_6), (v_1, v_7, v_8)\)
3. **joins:**
   \((v_1, v_2, v_3, )\)
   \((v_1, v_4, v_5, )\)
   \((v_1, v_4, , v_6)\)
   \((v_1, v_7, , v_8)\)
4. **answer:**
   \((v_1, v_4, v_5, v_6)\)
Local structural-indexes
Local structural-indexes

- Tree representation that is: small, enables querying
Local structural-indexes

- Tree representation that is: small, enables querying
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
  - Cluster represent “Forward” knowledge
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
  - Cluster represent “Forward” knowledge
- Semi local: F&B index [9].
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
  - Cluster represent “Forward” knowledge
- Semi local: F&B index [9].
  - “Forward” and “Backward” knowledge
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
  - Cluster represent “Forward” knowledge
- Semi local: F&B index [9].
  - “Forward” and “Backward” knowledge
  - Unscalable
Local structural-indexes

- Tree representation that is: small, enables querying
- The current indexes (Dataguide [5], 1-index [3]) are:
  - Local processing
  - Based on clustering
  - Cluster represent “Forward” knowledge
- Semi local: F&B index [9].
  - “Forward” and “Backward” knowledge
  - Unscalable
  - Local processing
How to locally index a tree?

```
  v1
   a
  / \
 v2  v4
  b   b
 /    /
v3   v5
  c    c
```

```
 v7
  b
 /   /
 v6  v8
  d   d
```
How to locally index a tree?

Matching Tree Patterns on Partial-trees
The local approach problem

an example

query: ‘/a/b[/c]/d’
The local approach problem
an example

query: ‘/a/b[/c]/d’

1. path1(‘/a/b/c’):
   (v₁, v₂, v₃),  (v₁, v₄, v₃),  (v₁, v₇, v₃),
   (v₁, v₂, v₅),  (v₁, v₄, v₅),  (v₁, v₇, v₅),
The local approach problem

an example

query: ‘/a/b[/c]/d’

1. path1(‘/a/b/c’):
   \((v_1, v_2, v_3), (v_1, v_4, v_3), (v_1, v_7, v_3), (v_1, v_2, v_5), (v_1, v_4, v_5), (v_1, v_7, v_5)\)

2. path2(‘/a/b/d’):
   \((v_1, v_2, v_6), (v_1, v_4, v_6), (v_1, v_7, v_6), (v_1, v_2, v_8), (v_1, v_4, v_8), (v_1, v_7, v_8)\)
The local approach problem
an example

query: ‘/a/b[/c]/d’

1. path1(‘/a/b/c’):
   \((v_1, v_2, v_3), (v_1, v_4, v_3), (v_1, v_7, v_3), (v_1, v_2, v_5), (v_1, v_4, v_5), (v_1, v_7, v_5)\)

2. path2(‘/a/b/d’):
   \((v_1, v_2, v_6), (v_1, v_4, v_6), (v_1, v_7, v_6), (v_1, v_2, v_8), (v_1, v_4, v_8), (v_1, v_7, v_8)\)

3. answer:
   \((v_1, \ldots, v_8)\)

Yet Another Pyrrhic victory
Outline

1 Motivation: Graph querying

2 Background: tree patterns

3 Partial trees - holistic divide and concur
In the rest of this talk we

- Supplies a model for
  1. holistic structural-indexing.
  2. holistic lazy pattern matching.
- See experimental results.
Data Model - An example
Query Model: partial-trees

- Two kind of node:
  1. Tree nodes (single label);
  2. Subtree nodes (multiple labels).
Query Model: partial-trees

- Two kind of node:
  1. Tree nodes (single label);
  2. Subtree nodes (multiple labels).

- Same as XPath pattern ‘/a///c[///d]’
Query matching: partial-trees

- Embedding $T_p$ is obtained from $T$ by a series of edge contractions.
- Embedding function $f$ relates $T_p$ and $T$ nodes.
- Example $f : (v_1, a), (v_4, c), (v_9, d), (v_2, \ast) \ldots$
- $Solution(T, T_p) = \bigcup f$. 
Query matching: partial-trees (cont.)

Q: How we match patterns on structural indexes and physical data models?
A: We model the data as a partial-tree.

Matching a partial-tree pattern on a partial-tree

\[
solution(T_p, T_p) \triangleq \bigcup_T \text{solution}(T, T_p) \, ^{-1} \circ \text{solution}(T, T_p)
\]
Query matching: partial-trees (cont.)

- **Q**: How we match patterns on structural indexes and physical data models?
- **A**: We model the data as a partial-tree.
- **Fast**: $O(|\overline{T_p}| \times |T_p|)$.

Matching a partial-tree pattern on a partial-tree

$$solution(\overline{T_p}, T_p) \triangleq \bigcup_T solution(T, \overline{T_p})^{-1} \circ solution(T, T_p)$$


Structural Indexing: partial-trees

An holistic safe Index:

1. Offline: Embed $T$ into an index $\overline{T_p}$
2. Online: $Solution(\overline{T_p}, T_p)$
Index example: offline phase
Index example: online phase

```
*  
<table>
<thead>
<tr>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>a₁,₂,₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>b, c₃-₅</td>
</tr>
<tr>
<td>b, c, d₇-₉</td>
</tr>
</tbody>
</table>
```
Index example: online phase
## Experimental results: index

<table>
<thead>
<tr>
<th>Data</th>
<th>K</th>
<th>Average coverage (%)</th>
<th>Average improvement (%)</th>
<th>Average gain</th>
<th>Maximal improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>2</td>
<td>57</td>
<td>95</td>
<td>2887</td>
<td>11.44</td>
</tr>
<tr>
<td>DBLP</td>
<td>3</td>
<td>86</td>
<td>87</td>
<td>8095</td>
<td>38.37</td>
</tr>
<tr>
<td>DBLP</td>
<td>4</td>
<td>95</td>
<td>81</td>
<td>16424</td>
<td>11.18</td>
</tr>
<tr>
<td>DBLP</td>
<td>5</td>
<td>9</td>
<td>72</td>
<td>20283</td>
<td>0.19</td>
</tr>
<tr>
<td>DBLP</td>
<td>6</td>
<td>100</td>
<td>58</td>
<td>53726</td>
<td>0.15</td>
</tr>
<tr>
<td>DBLP</td>
<td>7</td>
<td>100</td>
<td>44</td>
<td>60168</td>
<td>0.15</td>
</tr>
<tr>
<td>DBLP</td>
<td>8</td>
<td>100</td>
<td>47</td>
<td>55841</td>
<td>0.16</td>
</tr>
<tr>
<td>XMark</td>
<td>2</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>XMark</td>
<td>3</td>
<td>28</td>
<td>93</td>
<td>1936</td>
<td>58.2</td>
</tr>
<tr>
<td>XMark</td>
<td>4</td>
<td>46</td>
<td>78</td>
<td>5717</td>
<td>1.27</td>
</tr>
<tr>
<td>XMark</td>
<td>5</td>
<td>17</td>
<td>92</td>
<td>1810</td>
<td>1.16</td>
</tr>
<tr>
<td>XMark</td>
<td>6</td>
<td>22</td>
<td>96</td>
<td>627</td>
<td>8.9</td>
</tr>
<tr>
<td>XMark</td>
<td>7</td>
<td>28</td>
<td>87</td>
<td>3640</td>
<td>0.42</td>
</tr>
<tr>
<td>XMark</td>
<td>8</td>
<td>60</td>
<td>73</td>
<td>9184</td>
<td>0.53</td>
</tr>
</tbody>
</table>
Holistic matching data model: region encoding

Term | \((Doc, first, last, level)\)
--- | ---
‘a’ | \((1, 1, 9, 1), (1, 2, 5, 2), (1, 6, 9, 2)\)
‘b’ | \((1, 3, 5, 3), (1, 7, 9, 3)\)
‘c’ | \((1, 4, 4, 4), (1, 5, 5, 4), (1, 8, 8, 4)\)
‘d’ | \((1, 9, 9, 4)\)
Holistic matching data model: region encoding

<table>
<thead>
<tr>
<th>Term</th>
<th>(Doc, first, last, level)</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘a’</td>
<td>(1, 1, 9, 1), (1, 2, 5, 2), (1, 6, 9, 2)</td>
</tr>
<tr>
<td>‘b’</td>
<td>(1, 3, 5, 3), (1, 7, 9, 3)</td>
</tr>
<tr>
<td>‘c’</td>
<td>(1, 4, 4, 4), (1, 5, 5, 4), (1, 8, 8, 4)</td>
</tr>
<tr>
<td>‘d’</td>
<td>(1, 9, 9, 4)</td>
</tr>
</tbody>
</table>
Holistic matching: history

Holistic matching was developed by following ideas:

1. Algebraic approach: Binary joins.
2. Algebraic approach: Path joins.
3. Holistic approach: TwigStack \([4]\), Twig\(^2\) Stack.
4. Holistic approach: TwigTA
   - Theoretic foundations.
   - Controls the ‘Laziness’.
   - Predicts unmatched nodes before extraction.
   - Scalable.
Lazy holistic matching: partial-trees

The matching algorithm list of region-encodings of nodes. The algorithm iteratively does the following:

1. Extracts set of node-encodings $S$.
2. Translates $S$ into a partial tree $\overline{T_p}$.
3. Performs $Solution(\overline{T_p}, T_p)$ and refines a set of intermediate holistic solutions.
Matching example: iteration I - translates S into $T_p$
Matching example: iteration I - Solution($\overline{T_p}, T_p$)
Matching example: iteration 1 - \textit{Solution}(\overline{T_p}, T_p)
Matching example: iteration II
Matching example: iteration II
Experimental results: matching

- **TwigTA**[6] Vs. **TwigStack**[4].
- **TwigTA** prunes up to 99% of the nodes.
Summary

We:

- Understood local vs. holistic tree pattern matching.
- Learn about partial-tree pattern model and applied it for:
  - Holistic structural-indexing;
  - Holistic lazy pattern-matching.
Future Work

- What about graph data, graph patterns?
- What about other domains: RDF, streaming, image mining?
XML tasks and applications - accomplished in my PhD Thesis


¹ Optimizing XML Processing, 2010
XML tasks and applications - accomplished in my PhD Thesis

**Paper**


---

1 Optimizing XML Processing, 2010
XML tasks and applications - accomplished in my PhD Thesis

1

S. Harrusi, A. Averbuch.
Structural-indexes as automata: holistic approach.
Submitted to VLDB journal.

1 Optimizing XML Processing, 2010
XML tasks and applications - accomplished in my PhD Thesis


1 Optimizing XML Processing, 2010

Shachar Harussi  Supervision of Prof. Amir Averbuch
Matching Tree Patterns on Partial-trees
XML tasks and applications - accomplished in my PhD Thesis

S. Harrusi, A. Averbuch. XML streaming parsers on next generation SIM cards. Submitted to journal of data Management.

1 Optimize XML Processing, 2010
XML tasks and applications - accomplished in my PhD Thesis


\(^1\)Optimizing XML Processing, 2010
XML tasks and applications - accomplished in my PhD Thesis

1 Optimizing XML Processing, 2010

paper

The End!
C.Koch.
Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree automata-based approach.

M.Frick, M.Grohe, and C.Koch.
Query evaluation on compressed trees (extended abstract).

Tova Milo and Dan Suciu.
Index structures for path expressions, 1997.

N.Bruno, N.Koudas, and D.Srivastava.
Holistic twig joins: optimal XML pattern matching.

R.Goldman and J.Widom.
Dataguides: Enabling query formulation and optimization in semistructured databases.


S.Harrusi and A.Averbuch.
Tree automata based holistic twig pattern matching, i: Methodology.

T.Schwentick.
Automata for XML – a survey.

J. R. Ullmann.
An algorithm for subgraph isomorphism.

Wei Wang, Haifeng Jiang, Hongzhi Wang, Xuemin Lin, Hongjun Lu, and Jianzhong Li.
Efficient processing of XML path queries using the disk-based F&B index.