Matching Tree Patterns on Partial-trees

Optimizing Tree-Pattern Matching

Shachar Harussi
Supervision of Prof. Amir Averbuch

September 1, 2011

TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

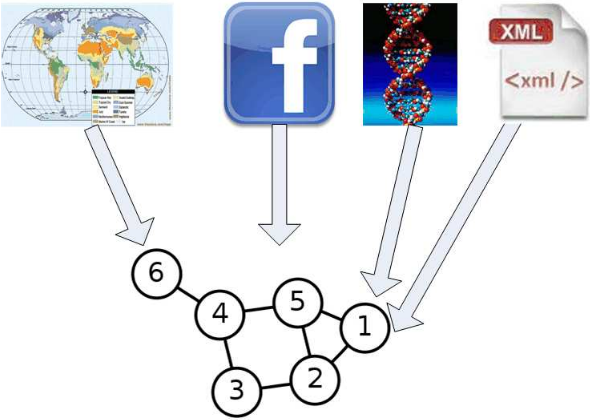
scaledb

Outline

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

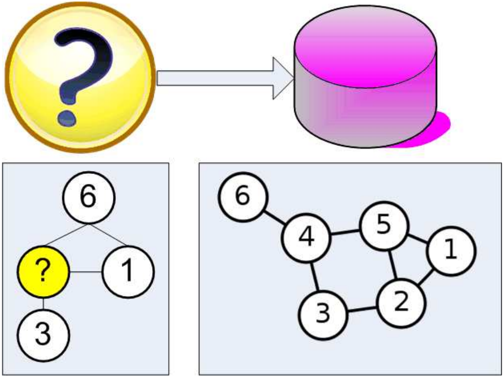
Motivation

Everything is a graph.

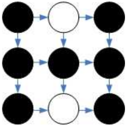


Motivation(cont.)

We need to query everything.

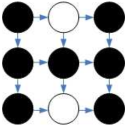


Lets take a picture

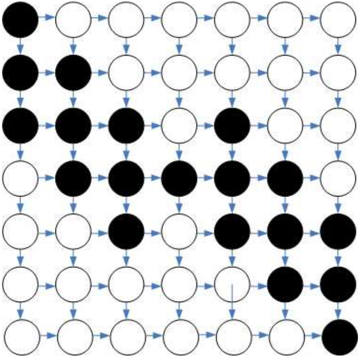


- A picture is a graph.

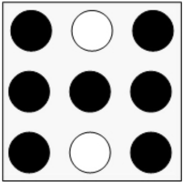
Lets take a picture



- A picture is a graph.

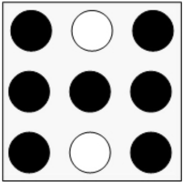


Holistic approach

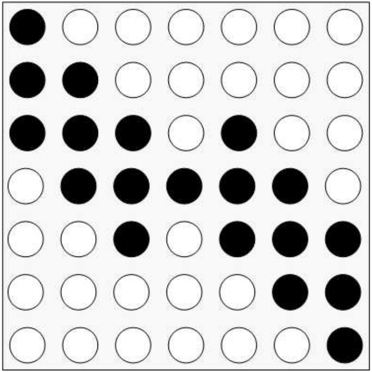


- Given a graph pattern.

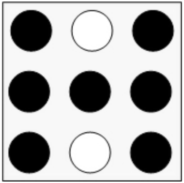
Holistic approach



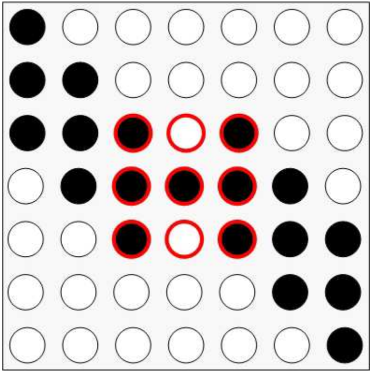
- Given a graph pattern.
- And a graph data,



Holistic approach



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



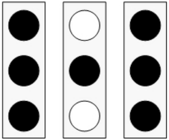
But Holistic is hard

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

But Holistic is hard

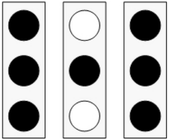
- A Problem: Holistic pattern matching is NP-hard.
- Even subgraph isomorphism problem [8] is hard.
- A Solution: divide and concur.

Local is easy

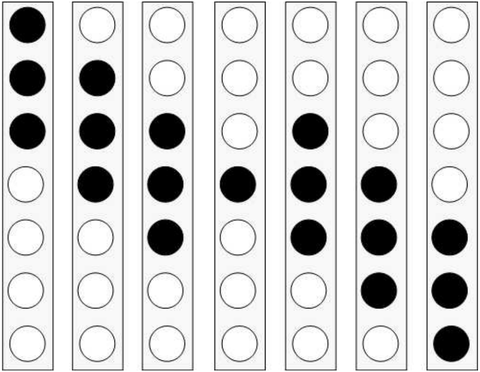


- Divide the pattern a local patterns P_i,

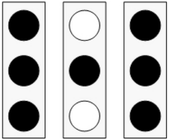
Local is easy



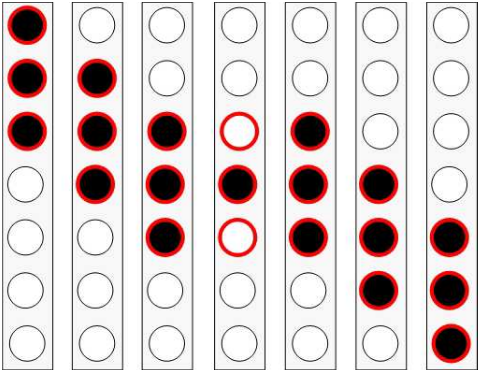
- Divide the pattern a local patterns P_i,
- And local data D_i.



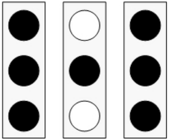
Local is easy



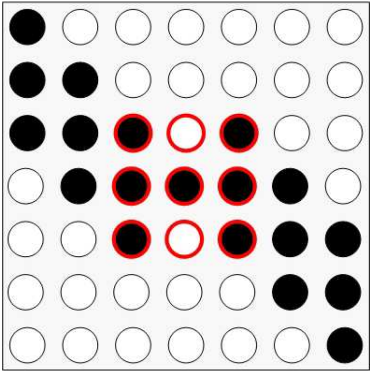
- Divide the pattern a local patterns P_i,
- And local data D_i.
- Partial solutions O O.
- Strings matching is fast $O(P_i \times D_i)$



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.



So local approach is perfect ?

- The answer is NO

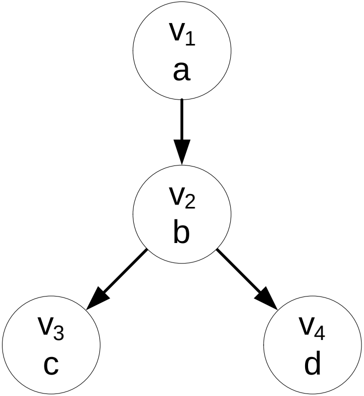
So local approach is perfect ?

- The answer is NO
- The concur is a Pyrrhic victory - i.e. the join costs.
- But lets focus on trees.

Outline

Tree Data Model

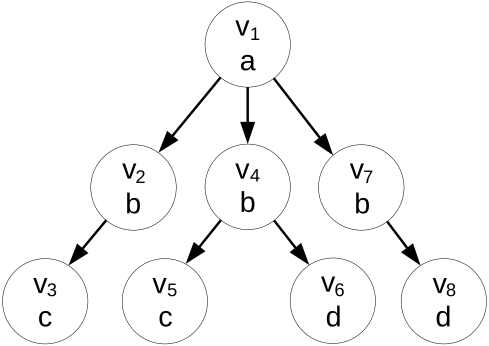
'**Tree pattern**' is a tree



The local approach problem

an example

query: '/a/b[/c]/d'



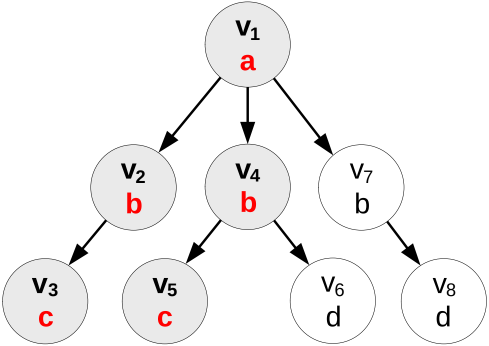
The local approach problem

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

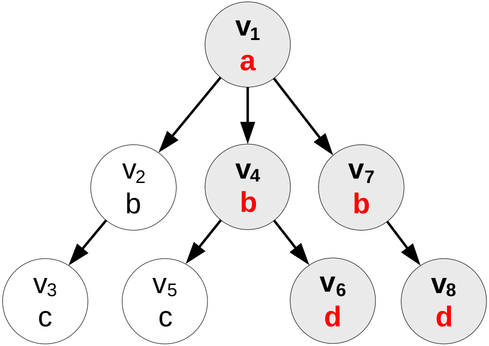
query: '/a/b[/c]/d'

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)

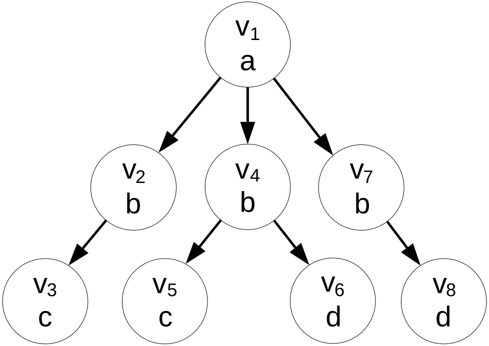
3.joins:

$(v_1, v_2, v_3, \quad)$

$(v_1, v_4, v_5, \quad)$

$(v_1, v_4, \quad , v_6)$

$(v_1, v_7, \quad , v_8)$



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:

$(v_1, v_2, v_3, \quad)$

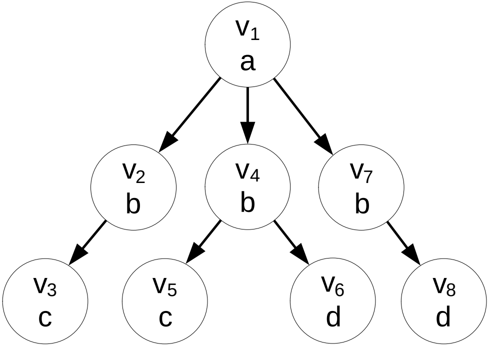
$(v_1, v_4, v_5, \quad)$

$(v_1, v_4, \quad, v_6)$

$(v_1, v_7, \quad, v_8)$

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

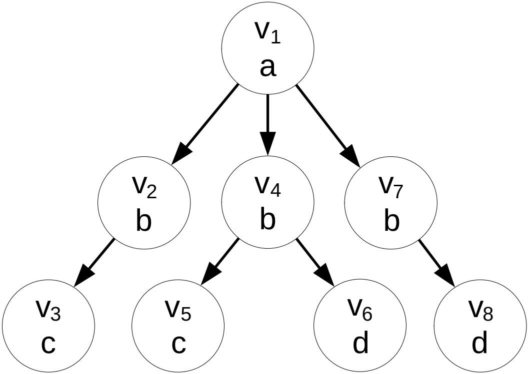
Local structural-indexes

- 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

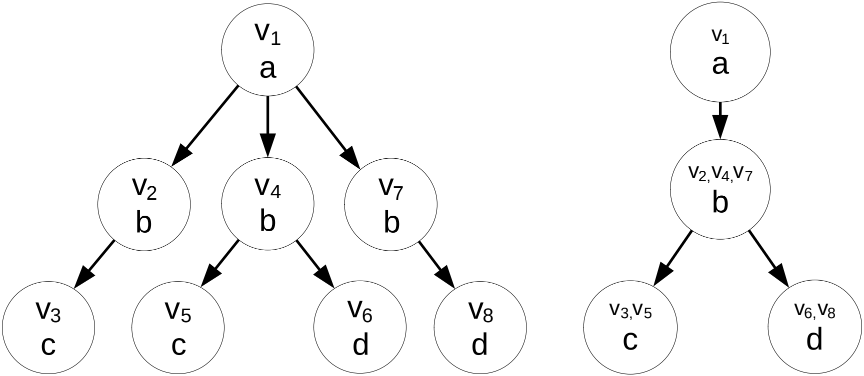
Local structural-indexes

- 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?



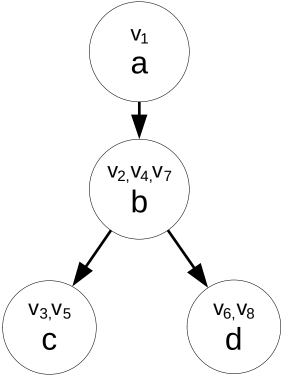
How to locally index a tree?



The local approach problem

an example

query: '/a/b[/c]/d'



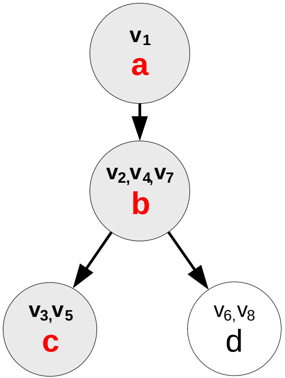
The local approach problem

an example

query: '/a/b[/c]/d'

1. path1('/a/b/c'):

(v_1, v_2, v_3), (v_1, v_4, v_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),



The local approach problem

an example

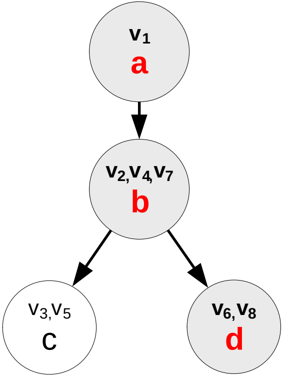
query: '/a/b[/c]/d'

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),

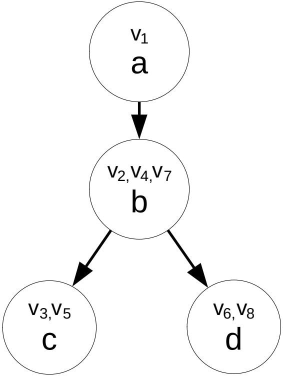
2. path2('/a/b/d'):

(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

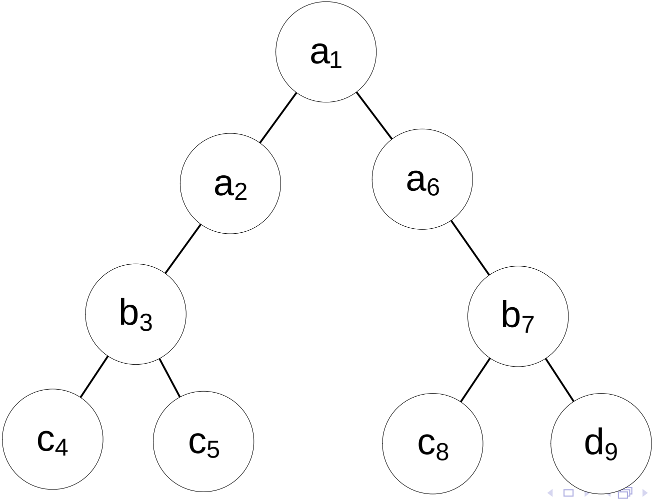


Outline

In the rest of this talk we

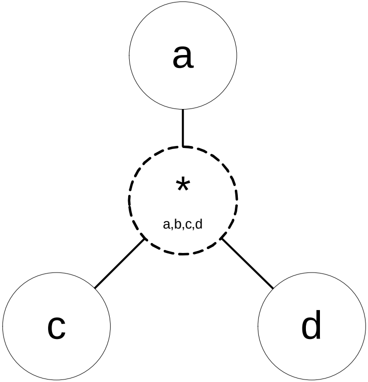
- Supplies a model for
  1. holistic structural-indexing.
  2. holistic lazy pattern matching.
- See experimental results.

Data Model - An example



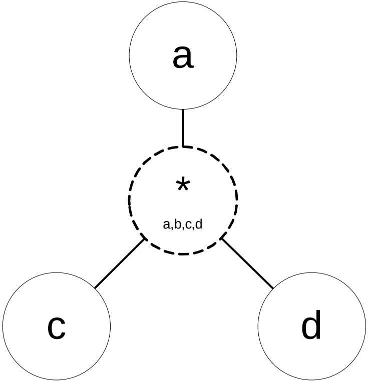
Query Model: partial-trees

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



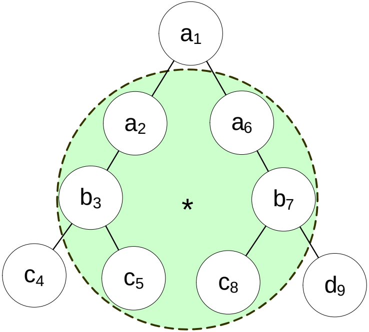
Query Model: partial-trees

- 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), $(v_2,\star)\ldots$
- $Solution(T, T_p) = \bigcup f$.



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 patrial-tree

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

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 patrial-tree

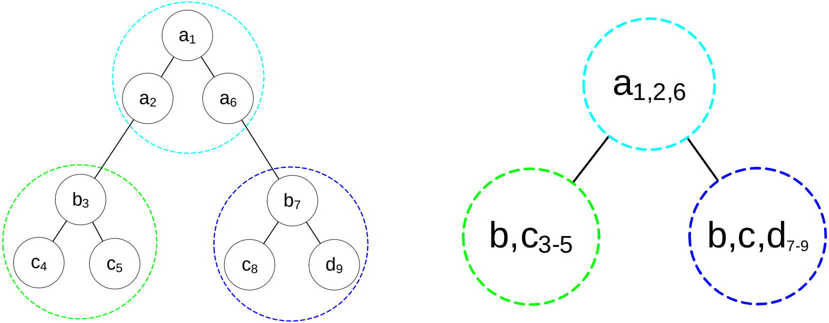
$$solution(\overline{T_p}, T_p) \triangleq \bigcup_T solution(T, \overline{T_p})^{-1} \circ solution(T, T_p)$$

Structural Indexing: partial-trees

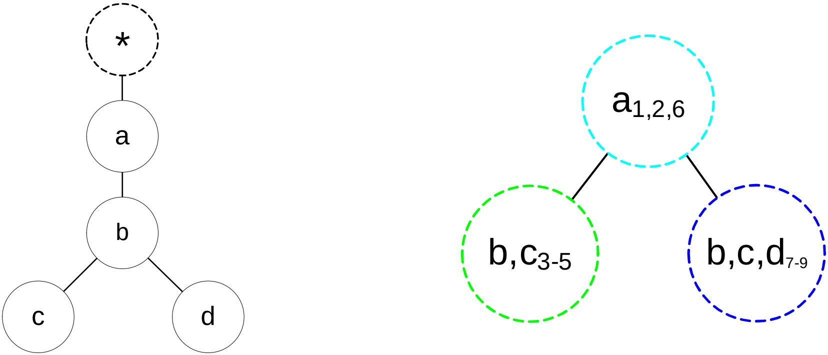
An holistic safe Index:

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

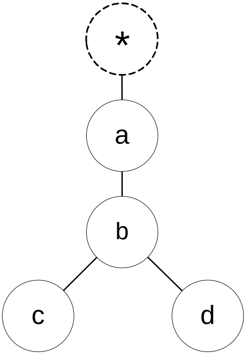
Index example: offline phase

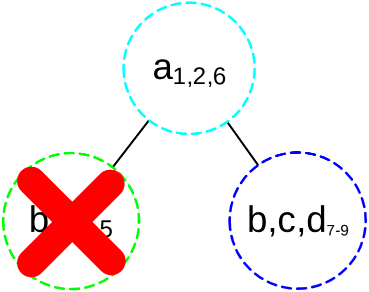


Index example: online phase



Index example: online phase

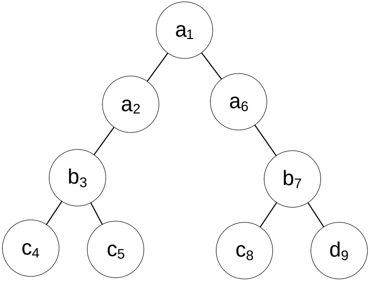




Experimental results: index

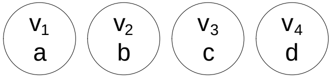
Data	K	Average coverage (%)	Average improvement (%)	Average gain	Maximal improvement (%)
DBLP	2	57	95	2887	11.44
DBLP	3	86	87	8095	38.37
DBLP	4	95	81	16424	11.18
DBLP	5	9	72	20283	0.19
DBLP	6	100	58	53726	0.15
DBLP	7	100	44	60168	0.15
DBLP	8	100	47	55841	0.16
XMark	2	0	100	0	100
XMark	3	28	93	1936	58.2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

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



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: history

Holistic matching was developed by following ideas:

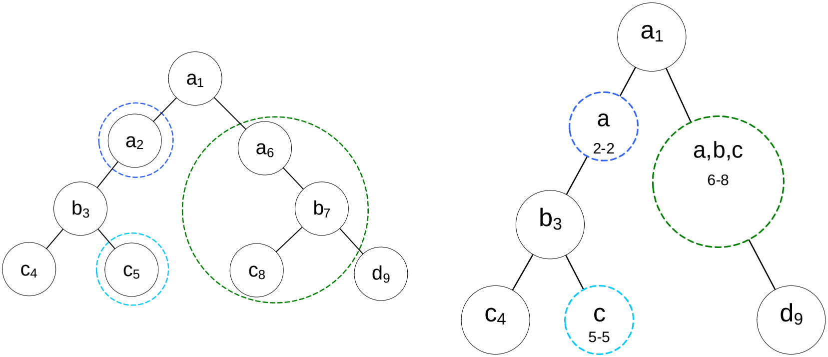
1. Algebraic approach: Binary joins.
2. Algebraic approach: Path joins.
3. Holistic approach: *TwigStack* [4], *Twig*^{2}*Stack*.
4. Holistic approach: *TwigTA*
   - Theoretic foundations.
   - Controls the ‘Laziness’.
   - Predicts unmatched nodes before extraction.
   - Scalable.

Lazy holistic matching: partial-trees

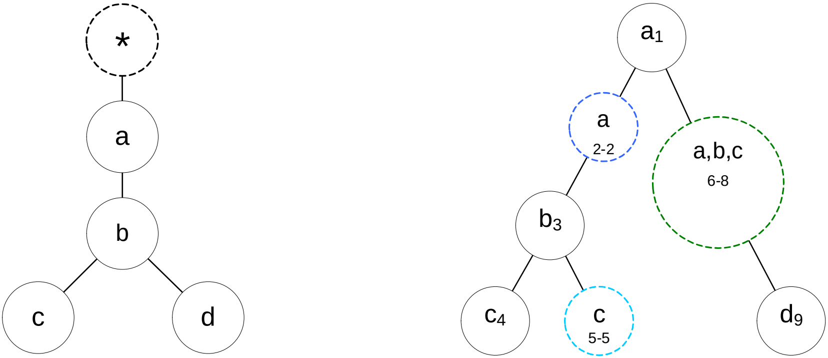
The matching algorithm list of region-encodings of nodes. The algorithm iteratively does the following:

1. Extracts set of node-encodings S.
2. Translates S into a partial tree $\overline{T_p}$.
3. Preforms $Solution(\overline{T_p}, T_p)$ and refines a set of intermediate holistic solutions.

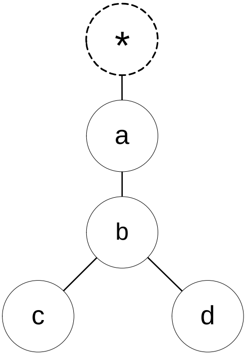
Matching example: iteration I - translates S into $\overline{T_p}$

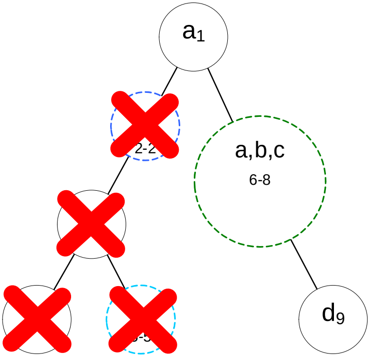


Matching example: iteration I - $Solution(\overline{T_p}, T_p)$

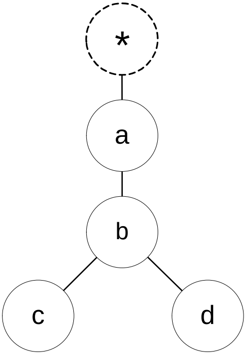


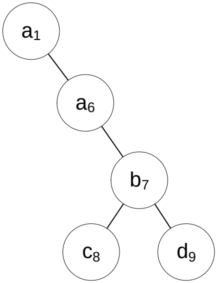
Matching example: iteration I - $Solution(\overline{T_p}, T_p)$



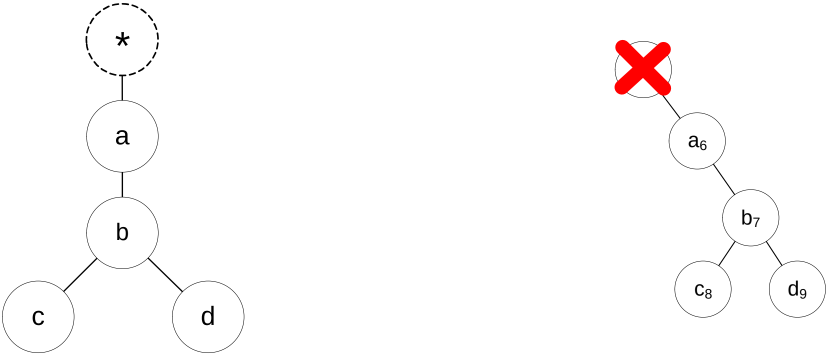


Matching example: iteration II



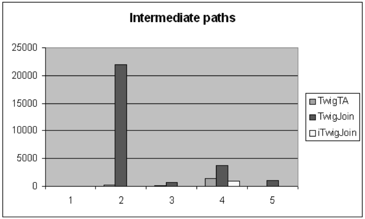


Matching example: iteration II



Experimental results: matching

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



Summary

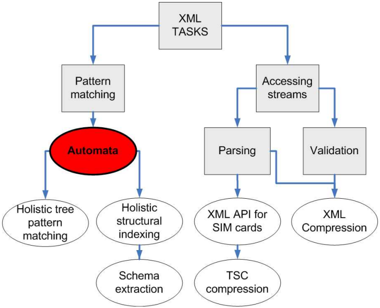
We:

- 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]

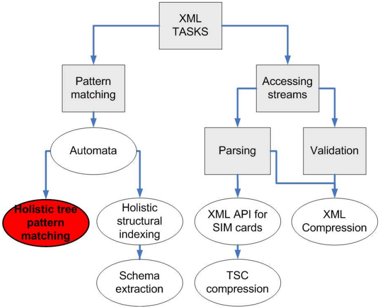


paper

S.Harrusi, A.Averbuch. Tree automata based holistic twig pattern matching: TA methodology. VLDB 2011.

[1] Optimizing XML Processing, 2010

XML tasks and applications - accomplished in my PhD Thesis [1]

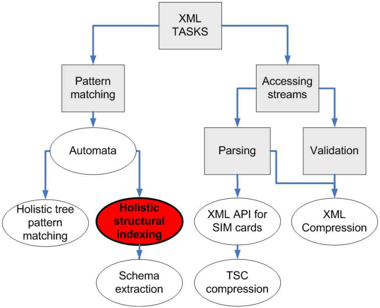


paper

S.Harrusi, A.Averbuch. Tree automata based holistic twig pattern matching: the TwigTA algorithm. ECCOMAS 2011.

[1] Optimizing XML Processing, 2010

XML tasks and applications - accomplished in my PhD Thesis [1]

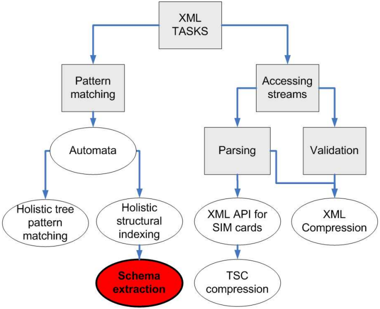


paper

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]

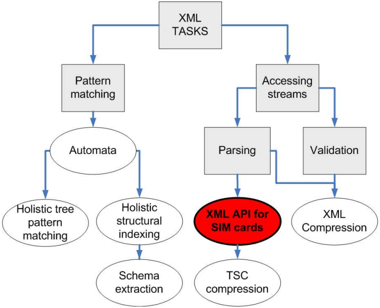


paper

S.Amiel, S.Harrusi, A.Averbuch. Semi-structured ordered tree language induction - XML Schema extraction. Submitted to VLDB journal.

[1] Optimizing XML Processing, 2010

XML tasks and applications - accomplished in my PhD Thesis [1]

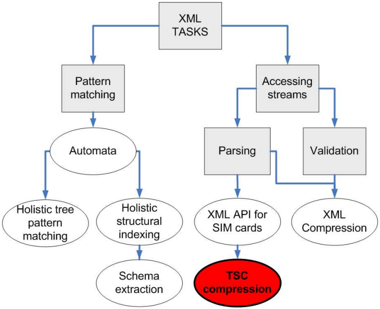


paper

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

[1] Optimizing XML Processing, 2010

XML tasks and applications - accomplished in my PhD Thesis [1]

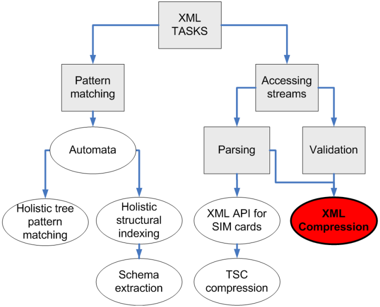


paper

S.Harrusi,A.Averbuch, N.Rabin A fast compact prefix encoding for pattern matching on limited resources device. Proceedings of Data Compression Conference (DCC 2010), Snowbird, Utah, March, 2010.

[1] Optimizing XML Processing, 2010

XML tasks and applications - accomplished in my PhD Thesis [1]



paper

S.Harrusi, A.Averbuch, A.Yehudai, XML syntax conscious compression, Data Compression Conference (DCC 2006), Snowbird, Utah, March 28 - 30, 2006, Proceedings of IEEE, pp. 402–411.

[1] Optimizing XML Processing, 2010

The End!

C.Koch.
Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree automata-based approach.
In *VLDB*, pages 249–260, 2003.

M.Frick, M.Grohe, and C.Koch.
Query evaluation on compressed trees (extended abstract).
LICS, pages 188–195, 2003.

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.
In *Proceedings of SIGMOD*, pages 310–321, 2002.

R.Goldman and J.Widom.

Dataguides: Enabling query formulation and optimization in semistructured databases.
In *Proceedings of VLDB*, pages 436–445, 1997.

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

T.Schwentick.
Automata for XML – a survey.
Journal of Computer and System Sciences, 73:289–315, 2007.

J. R. Ullmann.
An algorithm for subgraph isomorphism.
J. ACM, 23:31–42, January 1976.

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.
In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 145–156. VLDB Endowment, 2005.