

Efficient Evaluation of Generalized Tree-Pattern Queries with Same-Path Constraints

Xiaoying Wu¹, Dimitri Theodoratos¹, Stefanos Souldatos²,
Theodore Dalamagas², Timos Sellis²

¹New Jersey Institute of Technology, USA

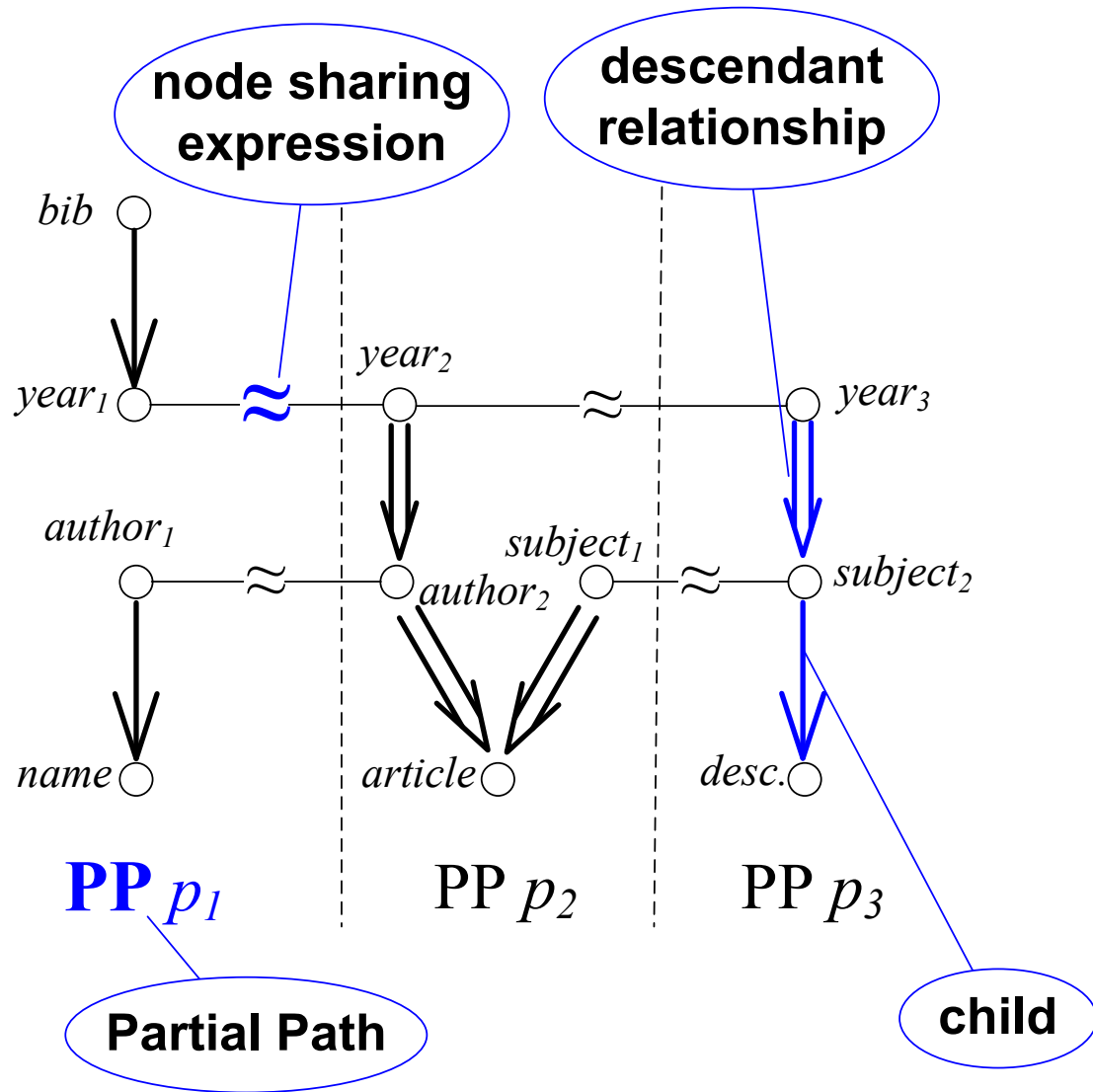
²National Technical University of Athens, Greece

Partial Tree-Pattern Query Language

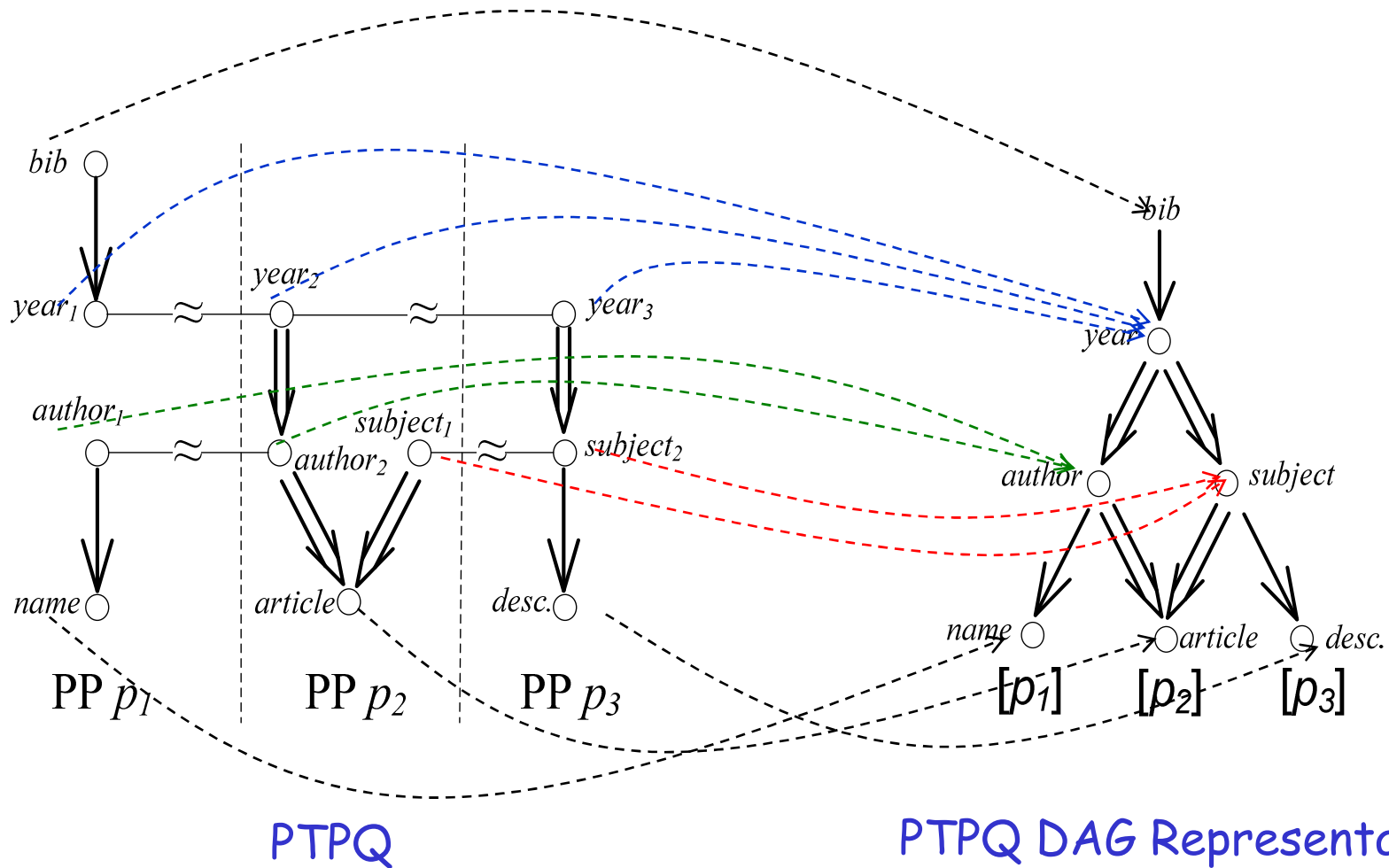
- ❖ A Partial Tree-Pattern Query (PTPQ)
 - Consists of {partial paths} + {node sharing expressions}
- ❖ Partial Paths (PP)
 - PPs comprise child and descendant relationships.
 - The child and descendant relationships among nodes in a PP *do not necessarily form a total order*.
 - The nodes of a PP are embedded to nodes on the *same XML tree path (same path constraint)*.
- ❖ Node sharing expressions (\approx)
 - A node sharing expression indicates that two nodes from different partial paths are *shared*.
 - They are embedded to the same XML tree node (*is-same-node* operator of XPath).

Partial Tree-Pattern Query Language

Query: find the articles authored by "Mary" and their subjects



An Annotated DAG Representation for PTPQs



A PTPQ can be represented by a DAG augmented with same path constraints.

Expressiveness of PTPQs

- ❖ PTPQs can express the fragment of XPath
 $XP\{ [], ./, //, |, ||, *, \approx \}$
- ❖ A PTPQ is equivalent to a *set* of TPQs
- ❖ PTPQs are flexible enough to allow a large range of queries :
 - keyword-style queries with no structure
 - keyword queries with arbitrary structural constraints
 - fully specified TPQs
- ❖ Advantages of PTPQs
 - useful for querying XML documents whose structure is complex or not fully known to the user,
 - useful for integrating XML data sources with different structures.

PTPQs Evaluation Challenges

- ❖ Given the large size of XML documents in current web applications, we cannot assume that the whole XML document is loaded in main memory for the evaluation of a query,
i.e., we need *non-main-memory* evaluation.
- ❖ PTPQs can only be represented as directed acyclic graphs (*dags*) annotated with *same-path constraints*.
- ❖ Unfortunately, existing non-main-memory evaluation algorithms focus almost exclusively on TPQs.
- ❖ Existing algorithms *cannot* be used directly or indirectly to compute PTPQs.

XML Query Evaluation Model

❖ XML Inverted Lists Model

- The data is *indexed* and the position of every node in the XML document tree is encoded.
- An *inverted list* is built on every node label.
- The nodes of the relevant inverted lists are read in the *pre-order* of their appearance in the XML tree.

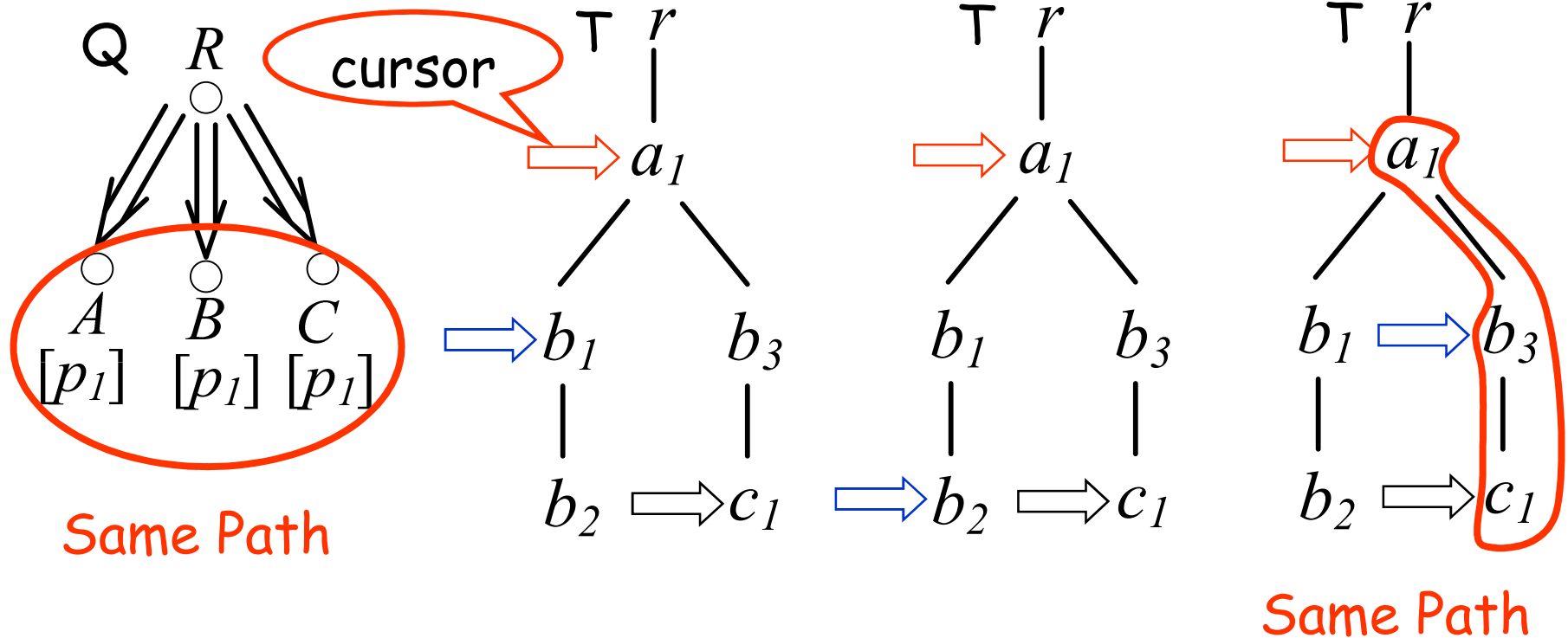
Algorithm PartialTreeStack

- ❖ It is a holistic stack-based algorithm
 - It takes into account the dag form of PTPQs and avoids redundant processing of subdags having multiple parents.
 - It exploits multiple pointers of stack entries.
 - It utilizes a topological ordering of the nodes in the query dag to match dag patterns directly to the XML tree.

Algorithm PartialTreeStack

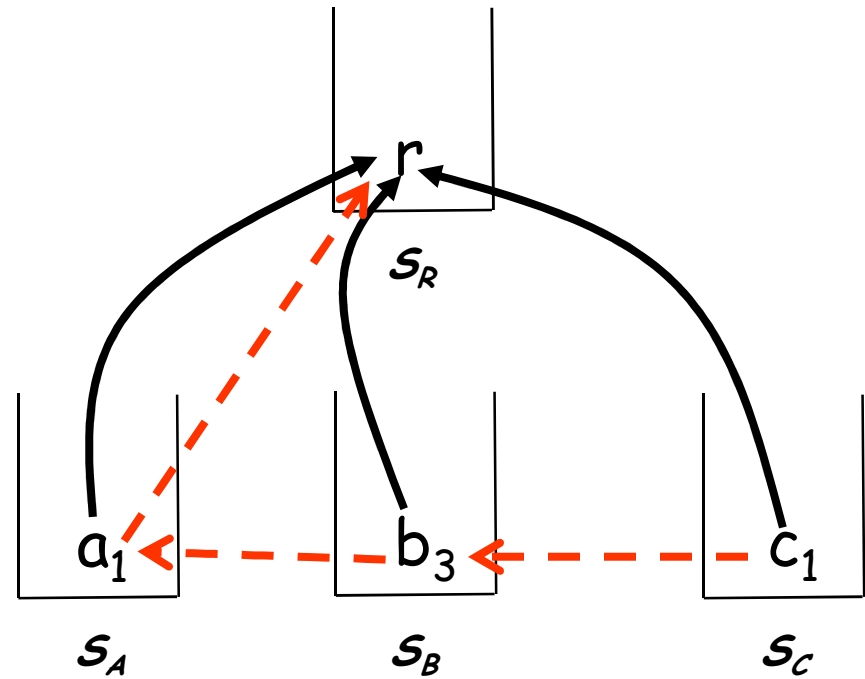
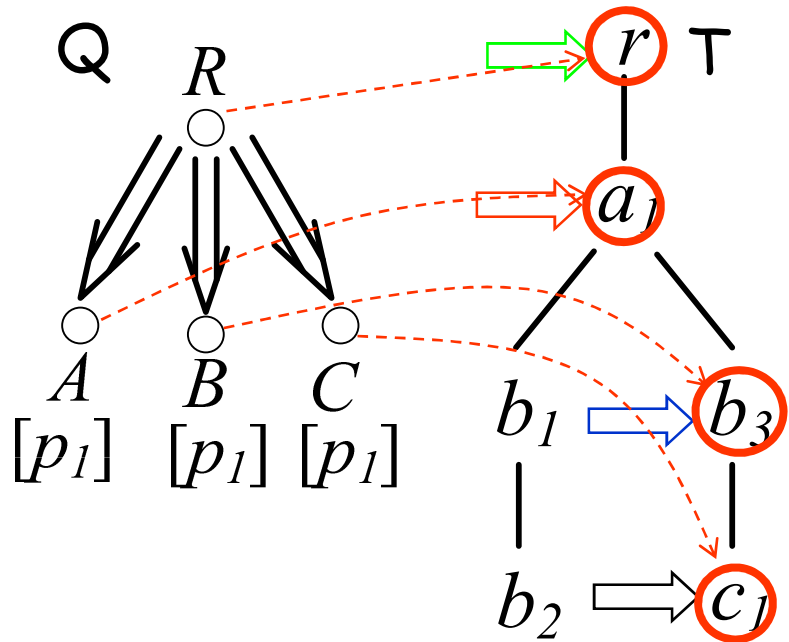
- ❖ **First Phase:** compute only solutions to individual partial paths of the query that are *guaranteed* to be in a final solution of the query.
- ❖ **Second phase,** the partial path solutions are merge-joined to compute the answer of the query.

Algorithm PartialTreeStack



A sequence of cursor movements result in matches of nodes A , B and C of Q that lie on the same path

Algorithm PartialTreeStack



The solution of Q on T: $ra_1b_3c_1$

Populating of the query stacks

Algorithm PartialTreeStack

- ❖ Correctness: Algorithm PartialTreeStack correctly evaluates a PTPQ Q on an XML tree T .
- ❖ Complexity:
 - Space complexity: $O(\text{recurDepth} \times |Q|)$.
 - Time complexity: $O(\text{input} \times |Q| \times P + \text{output} \times N)$
- ❖ $|Q|$: the size of Q
- ❖ N : the number of nodes of Q
- ❖ P : the number of partial paths of Q
- ❖ recurDepth : the recursion depth of Q on T
- ❖ input : the sum of sizes of the input streams
- ❖ output : the size of the answer of Q on T

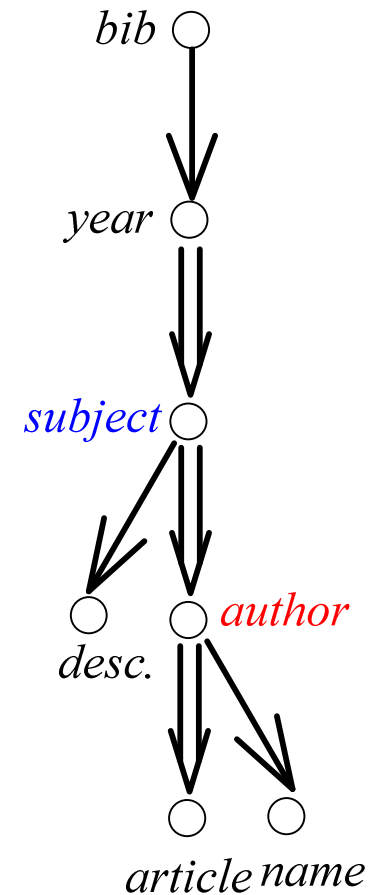
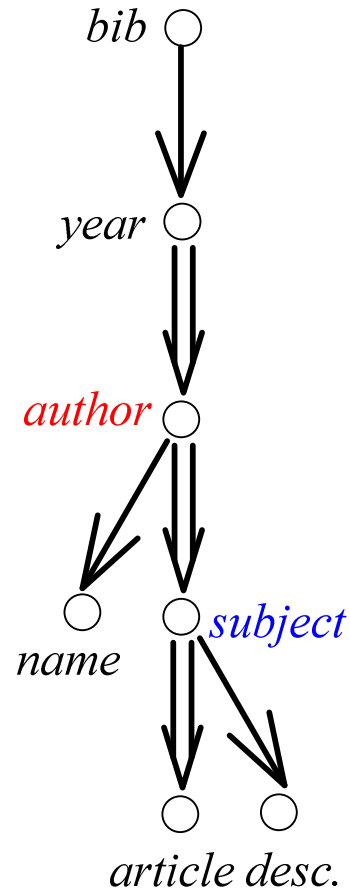
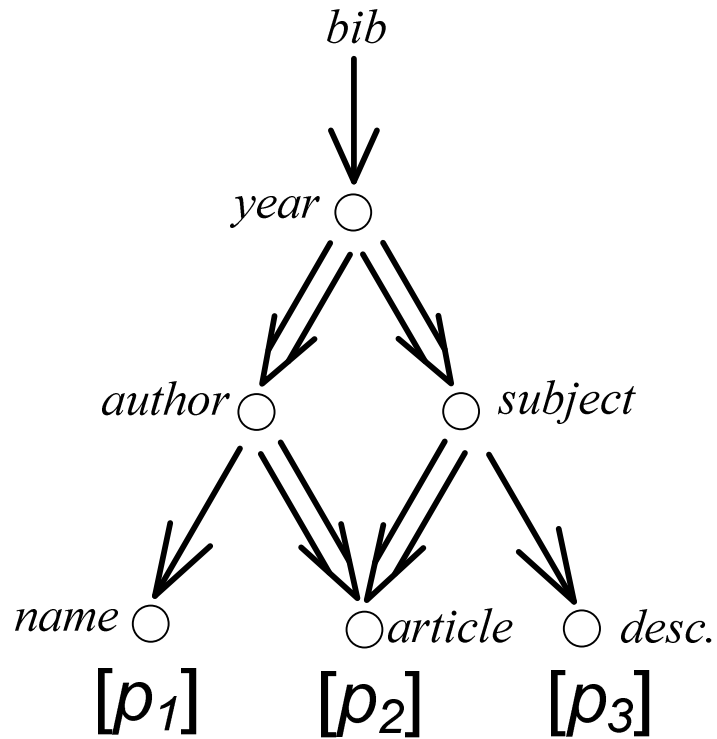
Comparison Algorithm: TPQGen

❖ Given a PTPQ Q , Algorithm TPQGen

- generates a set of TPQs which is equivalent to Q .
- uses the state-of-the-art algorithm TwigStack[1] to evaluate them.
- unions the results to produce the answer of Q .

[1] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: optimal XML pattern matching. In *SIGMOD*, 2002.

Algorithm TPQGen

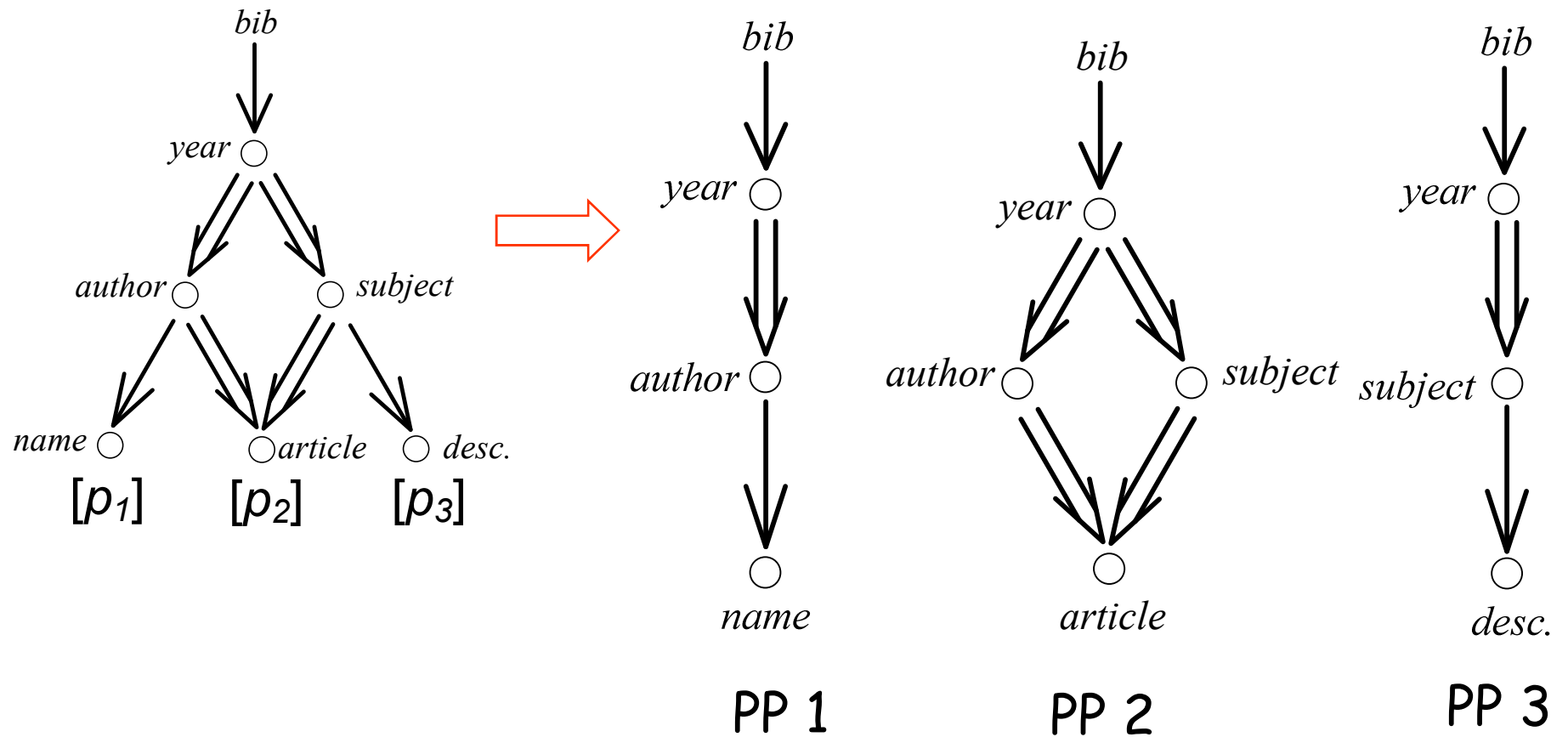


Comparison Algorithm: PartialPathJoin

❖ Given a PTPQ Q , Algorithm PartialPathJoin

- uses PartialPathStack-R to evaluate the corresponding partial path queries
- merge-joins the results on the common nodes (nodes participating in the node sharing expressions) to produce the answer of the PTPQ.

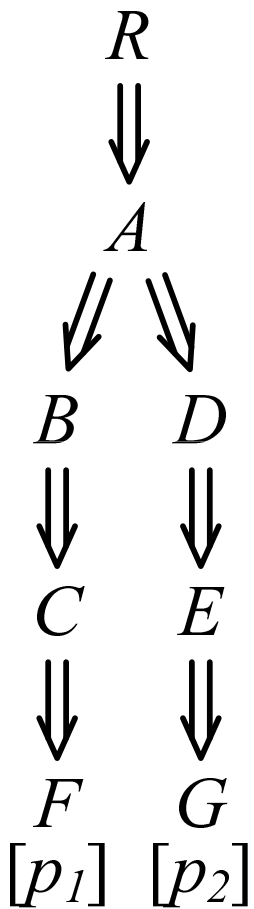
Algorithm PartialPathJoin



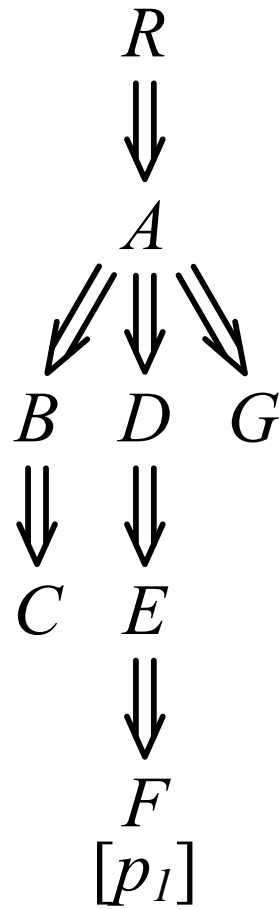
Experiments: datasets

Dataset	#labels	#nodes	height
Treebank	250	2.5 million	36
Synthetic	8	1.5 million	16

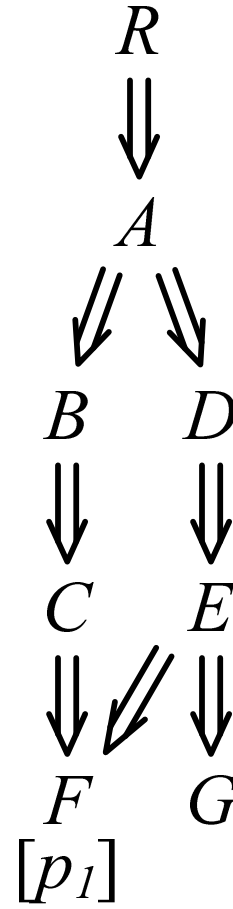
Experiments: queries used



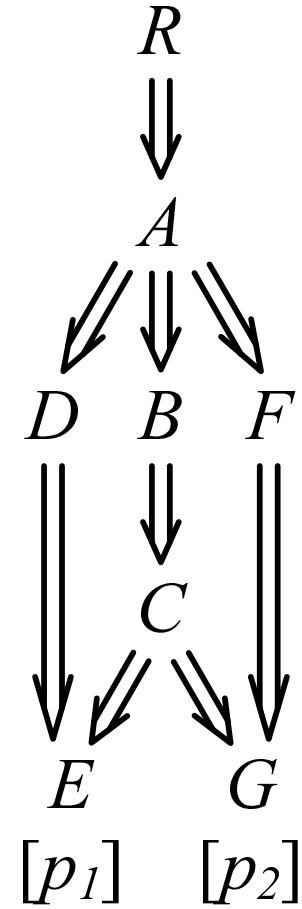
EQ1



EQ2

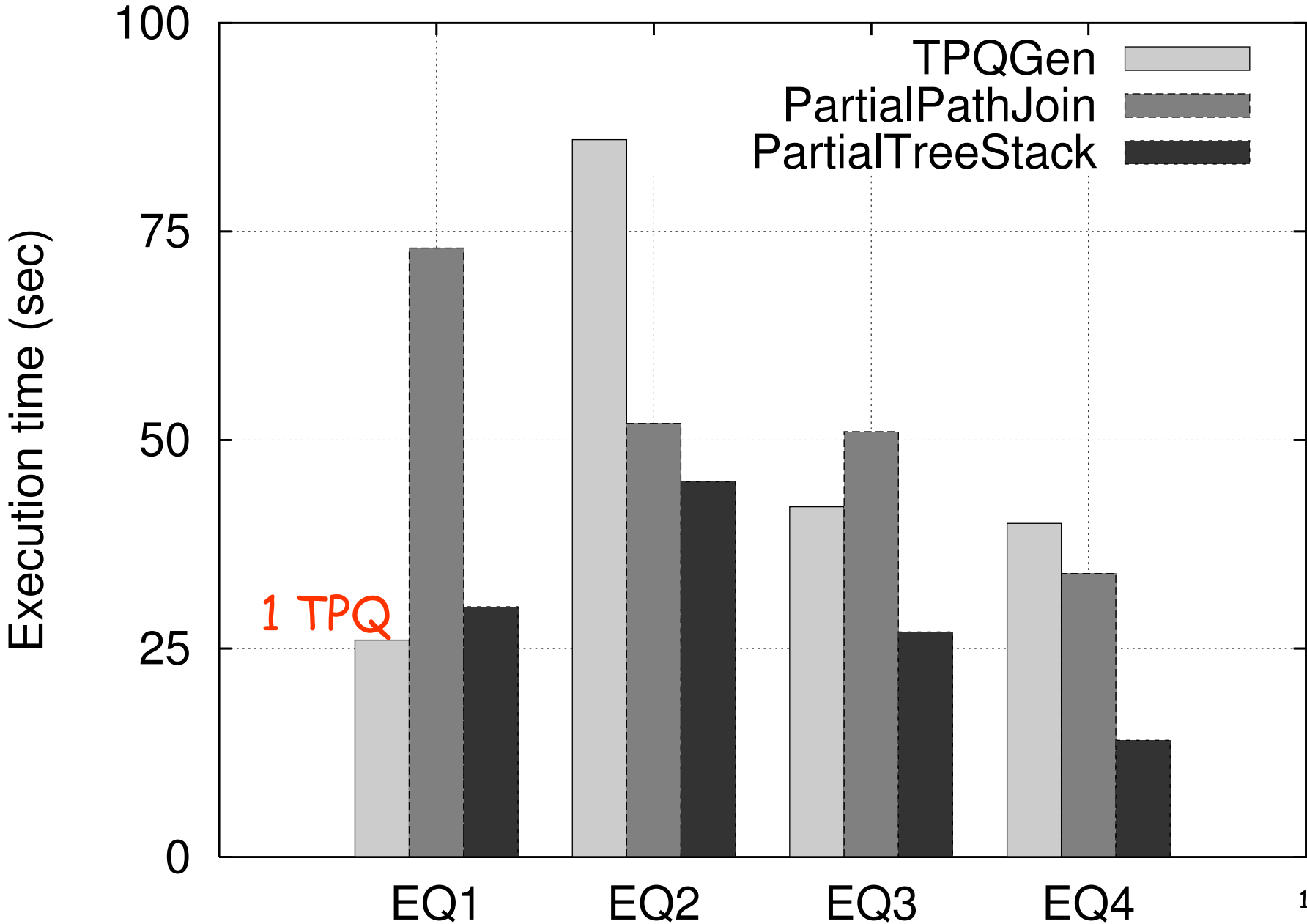


EQ3

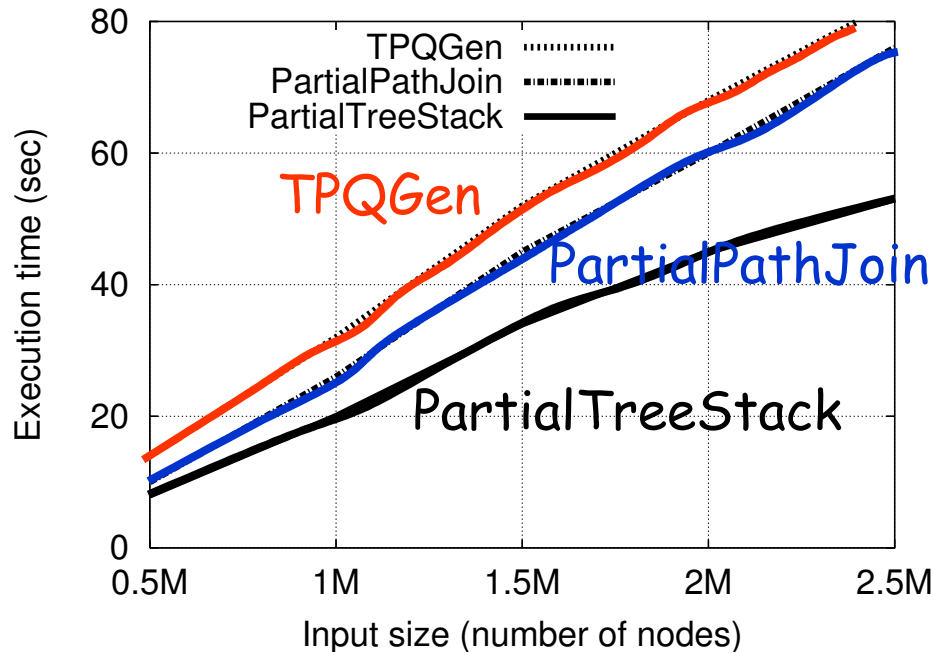


EQ4

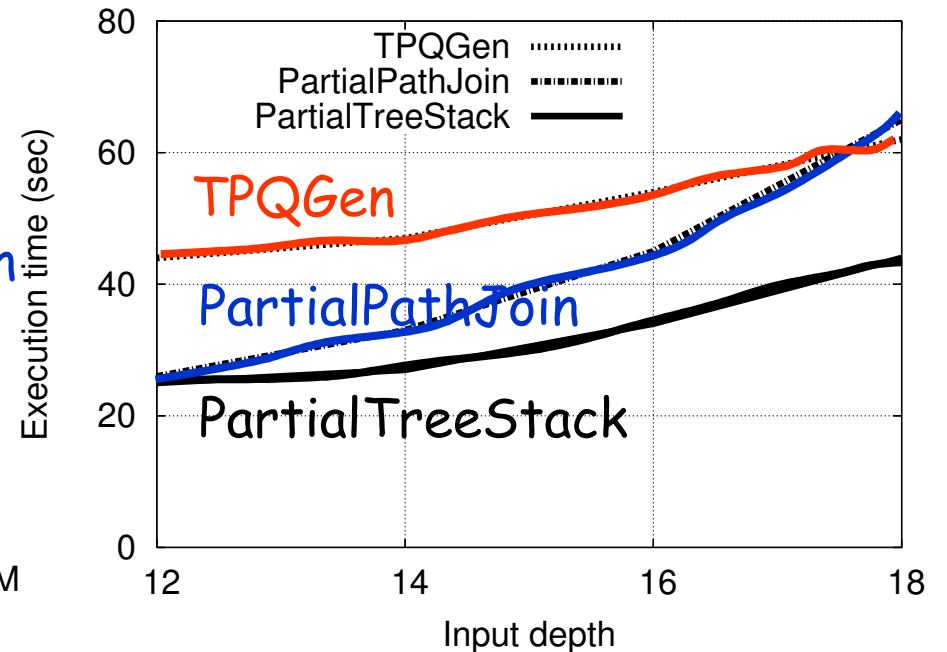
Query Execution Time on Treebank



Scalability on Synthetic Dataset



Execution time of EQ3 on synthetic dataset with increasing size



Execution time of EQ3 on synthetic dataset with increasing depth (height)

Conclusion

- ❖ We consider a query language called Partial Tree-Pattern Query Language (PTPQ) that is more expressive and flexible than Tree-Pattern Queries (TPQ).
- ❖ We have addressed the problem of efficient non-main-memory evaluation PTPQs on the XML inverted lists evaluation model.

Thank you

Please email your questions to:

Xiaoying wu
xw43@njit.edu