Evaluating “find a path” reachability queries

P. Bouros\textsuperscript{1}, T. Dalamagas\textsuperscript{2}, S. Skiadopoulos\textsuperscript{3}, T. Sellis\textsuperscript{1,2}

\textsuperscript{1}National Technical University of Athens
\textsuperscript{2}Institute for Management of Information Systems – R.C. Athena
\textsuperscript{3}University of Peloponnese
Outline

• Introduction
• Related work
• Introduce path representation of a graph
• Present an index for path representations
• Extend depth-first search for answering “find a path” reachability queries
• Experimental study
• Conclusion and Future work
Introduction

• Graphs, modelling complex problems
  – Spatial & road networks
  – Social networks
  – Semantic Web

• Important query type, reachability
  – “find a path” reachability query
    • Find a path between two given graph nodes
Answering “find a path” reachability queries

- **Two extreme approaches**
  - **No precomputation**
    - Exploit graph edges
    - Search algorithm
  - **Full precomputation**
    - Store path information in TC of the graph
    - Single lookups

- Full precomputation increases answering time
- No precomputation increases space requirement

- Single lookup
- Space requirement increases
- Search algorithm
Answering “find a path” reachability queries

- No precomputation
  - Tarjan, single source path expression problem

Full precomputation → answering time increases → No precomputation

- Single lookup
- Space requirement increases
- Search algorithm
  - Tarjan single source path

- Precomputation
  - Encoding each path between any graph nodes
  - Graph segmentation - $\rho$-Index
  - Labeling schemes
    - Determine whether exists a path, but cannot identify it

No precomputation → answering time increases → Full precomputation
Answering “find a path” reachability queries

- No precomputation
  - Tarjan, single source path expression problem
- Precomputation
  - Agrawal et al., encode each path between any graph nodes
Answering “find a path” reachability queries

- No precomputation
  - Tarjan, single source path expression problem
- Precomputation
  - Agrawal et al., encode each path between any graph nodes
  - Barton and Zezula, graph segmentation - ρ-Index
Answering “find a path” reachability queries

- **No precomputation**
  - Tarjan, single source path expression problem
- **Precomputation**
  - Agrawal et al., encode each path between any graph nodes
  - Barton and Zezula, graph segmentation - ρ-Index
- **Labeling schemes**
  - Determine whether exists a path, but cannot identify it
Answering “find a path” reachability queries

• Our idea
  – Represent the graph as a set of paths
  – Each path contains precomputed answers
  – Precompute and store part of path information in TC of the graph
Answering “find a path” reachability queries

- Our idea
  - Represent the graph as a set of paths
  - Each path contains precomputed answers
  - Precompute and store part of path information in TC of the graph
- In the middle
  - No need to compute TC
In brief

• Propose a novel representation of a graph as a set of paths (path representation)
• Present an index for providing efficient access in representation (P-Index)
• Extend depth-first search to work with paths in answering “find a path” reachability queries (pdfs)
• Preliminary experimental evaluation
Graph – Representations - Indices

G(V,E)

- edges E(G)
  - Adjacency list
- paths P(G)
  - P-Index

graph
representation
index
Path representation

- **Set of paths**
  - Stores part of path information in TC of a graph
  - Combines graph edges to efficiently answer “find a path” reachability queries
  - Preserves reachability information
    - Each graph edge is contained in at least one path
    - Construct graph by merging paths

- **Not unique**
Path representation – Example

G
Path representation – Example

G

<table>
<thead>
<tr>
<th>p1</th>
<th>(A,B,C,E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
<td>(C,D,B,F)</td>
</tr>
<tr>
<td>p3</td>
<td>(C,H)</td>
</tr>
<tr>
<td>p4</td>
<td>(D,K)</td>
</tr>
</tbody>
</table>

\[ P(G) = \{p1,p2,p3,p4\} \]
Path representation – Example

\[ P(G) = \{p1, p2, p3, p4\} \]

<table>
<thead>
<tr>
<th>p1</th>
<th>(A, B, C, E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
<td>(C, D, B, F)</td>
</tr>
<tr>
<td>p3</td>
<td>(C, H)</td>
</tr>
<tr>
<td>p4</td>
<td>(D, K)</td>
</tr>
</tbody>
</table>
P-Index

• Consider graph $G(V,E)$ and its path representation $P(G)$
  – For each node $v$ in $V$ retain $\text{paths}[v]$ list of paths in $P(G)$ containing $v$
  – $P\text{-Index}(G) = \{\text{paths}[v_i]\}$, for each $v_i$ in $V$
P-Index – Example

\[ G \]

\[
\begin{array}{c|c}
 p1 & (A,B,C,E) \\
p2 & (C,D,B,F) \\
p3 & (C,H) \\
p4 & (D,K) \\
\end{array}
\]
P-Index – Example

P-Index(G)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>p1, p2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>p1, p2, p3</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>p2, p4</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>p1</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>p2</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>p3</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>p4</td>
<td></td>
</tr>
</tbody>
</table>

P(G)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>(A, B, C, E)</td>
</tr>
<tr>
<td>p2</td>
<td>(C, D, B, F)</td>
</tr>
<tr>
<td>p3</td>
<td>(C, H)</td>
</tr>
<tr>
<td>p4</td>
<td>(D, K)</td>
</tr>
</tbody>
</table>
P-Index – Example

**P(G)**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>(A,B,C,E)</td>
</tr>
<tr>
<td>p2</td>
<td>(C,D,B,F)</td>
</tr>
<tr>
<td>p3</td>
<td>(C,H)</td>
</tr>
<tr>
<td>p4</td>
<td>(D,K)</td>
</tr>
</tbody>
</table>

**P-Index(G)**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>p1</td>
</tr>
<tr>
<td>B</td>
<td>p1, p2</td>
</tr>
<tr>
<td>C</td>
<td>p1, p2, p3</td>
</tr>
<tr>
<td>D</td>
<td>p2, p4</td>
</tr>
<tr>
<td>E</td>
<td>p1</td>
</tr>
<tr>
<td>F</td>
<td>p2</td>
</tr>
<tr>
<td>H</td>
<td>p3</td>
</tr>
<tr>
<td>K</td>
<td>p4</td>
</tr>
</tbody>
</table>
Algorithm pdfs

- Answers “find a path” reachability queries
- Extends depth-first search to work with paths
  - For each node, visit
    - Not only its children
    - Also, its successors in paths of P(G)
- Input: graph G(V,E), P(G), P-Index(G)
  - Current path stack `curPath`
- Method:
  - If exists path in P(G) where source before target
  - While `curPath` not empty
    - Read top node u of `curPath`
    - Read a path p containing top u – If no path left, pop u
    - Else for each node v in p after u
      - Case 1: if exists path in P(G) where v before target then FOUND path
      - Case 2: if `visited[v]=FALSE` then push it in `curPath`, `visited[v]=TRUE`
      - Case 3: if `visited[v]=TRUE` then ignore rest of nodes in p
Query: FindAPath(B,K)
pdfs – Example

G

A

D

F

K

E

H

• p1 contains B

P(G)

<table>
<thead>
<tr>
<th>p1</th>
<th>(A,B,C,E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
<td>(C,D,B,F)</td>
</tr>
<tr>
<td>p3</td>
<td>(C,H)</td>
</tr>
<tr>
<td>p4</td>
<td>(D,K)</td>
</tr>
</tbody>
</table>

A  | p1 |
B  | p1, p2|
C  | p1, p2, p3|
D  | p2, p4|
E  | p1 |
F  | p2 |
H  | p3 |
K  | p4 |
• visit C,E
• no path in P(G) contains either C or E before target K
E contained in p1 at the end
### P-Index(G)

- E contained in p1 at the end
- pop E

<table>
<thead>
<tr>
<th>Node</th>
<th>Path(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>p1</td>
</tr>
<tr>
<td>B</td>
<td>p1, p2</td>
</tr>
<tr>
<td>C</td>
<td>p1, p2, p3</td>
</tr>
<tr>
<td>D</td>
<td>p2, p4</td>
</tr>
<tr>
<td>E</td>
<td>p1</td>
</tr>
<tr>
<td>F</td>
<td>p2</td>
</tr>
<tr>
<td>H</td>
<td>p3</td>
</tr>
<tr>
<td>K</td>
<td>p4</td>
</tr>
</tbody>
</table>
pdfs – Example

P(G)

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(A,B,C,E)</td>
<td>(C,D,B,F)</td>
<td>(C,H)</td>
<td>(D,K)</td>
</tr>
</tbody>
</table>

P-Index(G)

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p1, p2</th>
<th>p1, p2, p3</th>
<th>p2, p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>p1</td>
<td>p1, p2</td>
<td>p1, p2, p3</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>p2, p4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>p1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>p2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>p3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>p4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• p1 contains C
• But E already visited, next path
pdfs – Example

![Graph Diagram]

**P(G)**

| p1 | (A,B,C,E) |
| p2 | (C,D,B,F) |
| p3 | (C,H)     |
| p4 | (D,K)     |

**P-Index(G)**

- A: p1
- B: p1, p2
- C: p1, p2, p3
- D: p2, p4
- E: p1
- F: p2
- H: p3
- K: p4

**Notes:**
- p1 contains C
- But E already visited, next path
- p2 contains C
• p1 contains C
• But E already visited, next path
• p2 contains C
• consider D, exists path in P(G) containing D before target K: p4
FOUND path from B to K (B,C,D,K)
Experimental study

• Sets of random graphs

• Construct path representations
  – Traverse graph in depth-first manner starting from several nodes
  – Terminate when all graph edges included
  – Promote construction of long paths
    • Reusing graph edges

• Experimental parameters
  – Graph nodes $|V|$: $10^4$, $5*10^4$, $10^5$, $5*10^5$, $10^6$
  – Avg degree $d = |E|/|V|$: 2, 3, 4, 5, 10
  – Max length of paths in $P(G)$ $L_{max}$: 10, 20, 30, 40, 50
Varying max path length

- Graph G: $|V|=100,000$ & $d=4$, 5 different path representations
- 1,000 “find a path” queries
- As $L_{\text{max}}$ increases
Varying max path length

- Graph G: $|V|=100,000$ & $d=4$, 5 different path representations
- 1,000 “find a path” queries
- As $L_{max}$ increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
Varying max path length

- Graph G: |V|=100,000 & d=4, 5 different path representations
- 1,000 “find a path” queries
- As $L_{\text{max}}$ increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
  - Storage requirements increase
Varying max path length

- Graph G: $|V|=100,000$ & $d=4$, 5 different path representations
- 1,000 “find a path” queries
- As $L_{max}$ increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
  - Storage requirements increase

![Graph showing average execution time vs maximum path length](image1)

![Graph showing storage overhead vs maximum path length](image2)

~2.5 more edges
Varying max path length

- **Graph G**: $|V|=100,000$ & $d=4$, 5 different path representations
- 1,000 “find a path” queries
- As $L_{\text{max}}$ increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node $u$ before target
  - Storage requirements increase
Varying avg degree

- Initial graph $G$: $|V| = 100,000$ & $d=2$ & $L_{\text{max}} = 30$
  - Progressively add edges
- 1,000 “find a path” queries
- More dense graph
  - Larger number of long paths
  - Fewer short paths
Varying number of graph nodes

- 5 graphs: $d=4 \& L_{max}=30$
- 1,000 “find a path” queries
- $|V|$ increases
  - Paths have fewer common nodes
  - Less possibly exists a path in $P(G)$ where node $u$ before target
Conclusions and Future work

• Conclusions
  – Propose a novel representation of a graph as a set of paths
  – Present P-Index
  – Extend depth-first search to work with paths in answering “find a path” reachability queries
  – Preliminary experimental evaluation

• Future work
  – Answer “find a path” with length constraint reachability queries
  – Updates
  – Introduce cost model for path representation
    • Construction of the set of paths
    • Answering queries cost
    • Updating representation cost
Questions ?
Evaluating “find a path” reachability queries

Additional slides
Answering queries – Basic idea

Find a path from S to T
Answering queries – Basic idea

S contained in $p_1, p_2, p_3$
Answering queries – Basic idea

Consider $p_2$ part – X last node
S contained in $p_4, p_5$
Answering queries – Basic idea

Consider $p_5$ part – $Z$ last node
Z only contained in $p_5$ – backtrack to Y
Answering queries – Basic idea

Y contained in $p_6$
Answering queries – Basic idea

Consider $p_6$ part – FOUND target $T$
Varying number of graph nodes

![Graph showing the relationship between number of nodes and average number of nodes processed. The x-axis represents the number of nodes, while the y-axis shows the average number of nodes processed. Two lines are depicted: one for dfs and another for pdfs. The dfs line shows a linear increase, while the pdfs line remains relatively flat.]