Evaluating “find a path” reachability queries

P. Bouros[1], T. Dalamagas[2], S.Skiadopoulos[3], T. Sellis[1,2]

[1]National Technical University of Athens

[2]Institute for Management of Information Systems – R.C. Athena

[3]University of Peloponnese

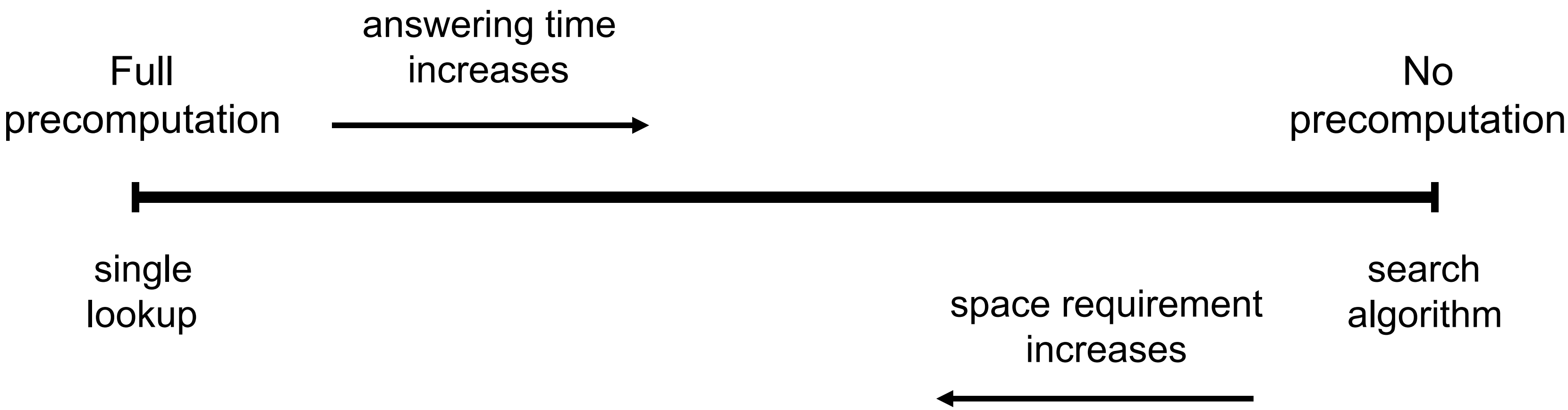
Outline

- 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

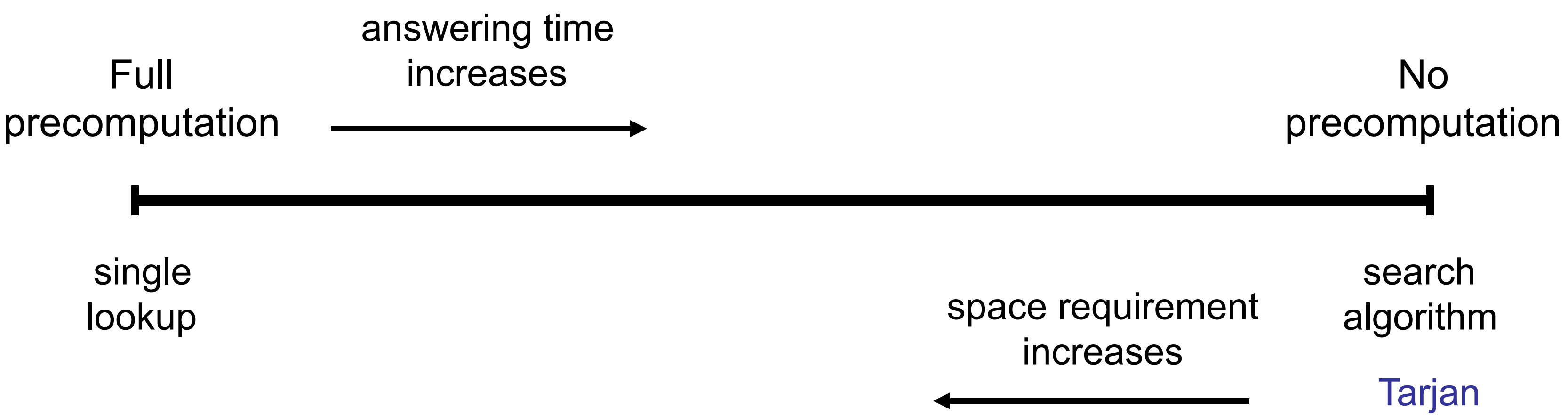
- Graphs, modelling complex problems
  - Spatial & road networks
  - Social networks
  - Semantic Web
- Important query type, reachability
  - "find a path" reachability query
    - Find a path between two given graph nodes

Answering "find a path" reachability queries



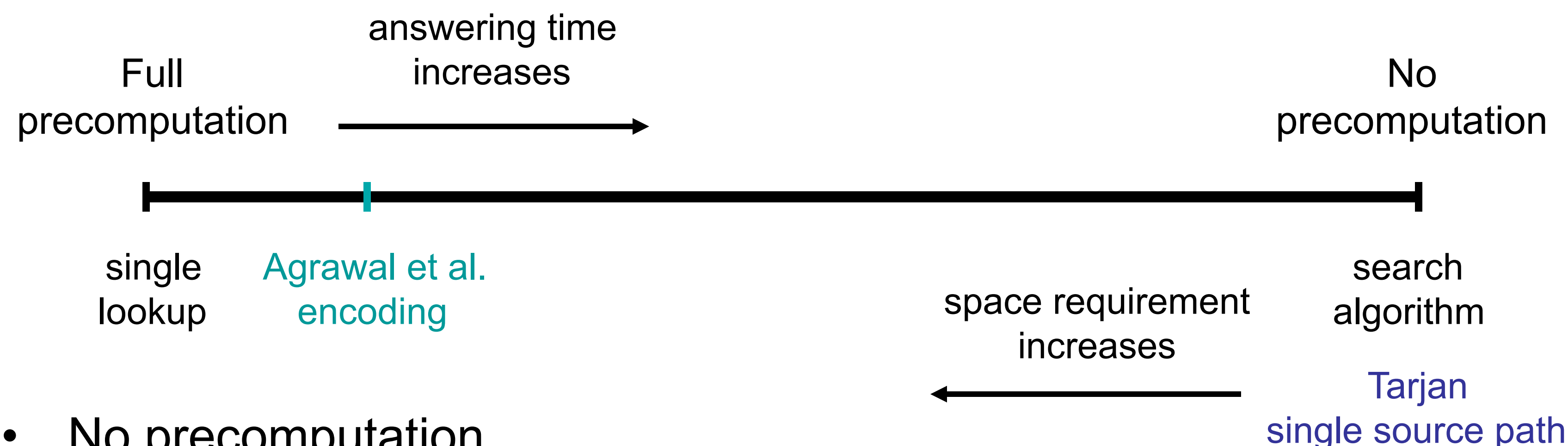
- Two extreme approaches
  - No precomputation
    - Exploit graph edges
    - Search algorithm
  - Full precomption
    - Store path information in TC of the graph
    - Single lookups

Answering “find a path” reachability queries



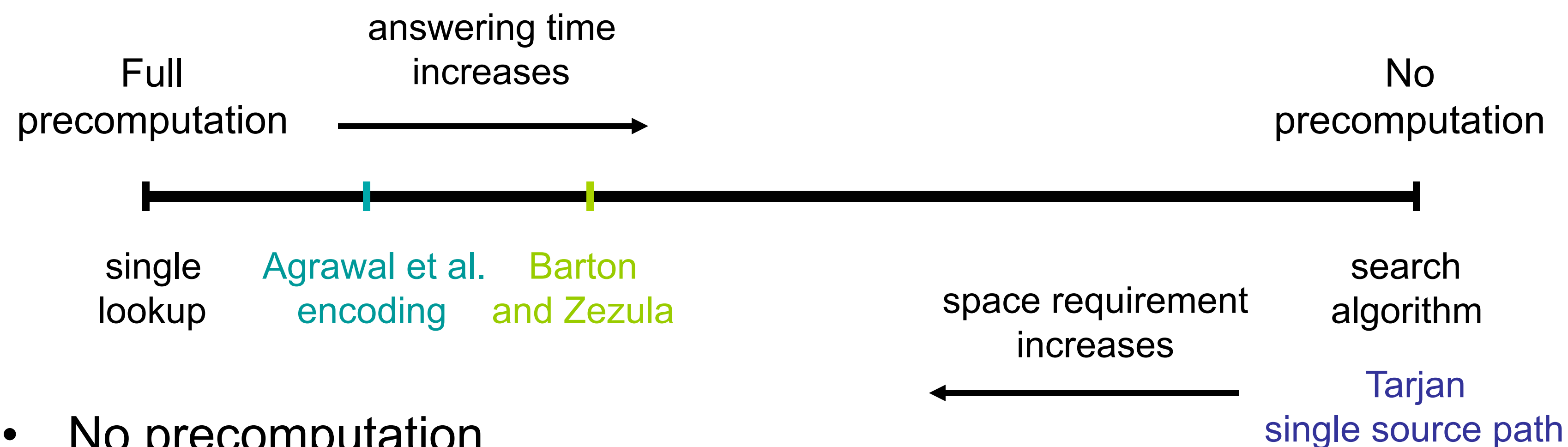
- No precomputation
  - Tarjan, single source path expression problem

Answering "find a path" reachability queries



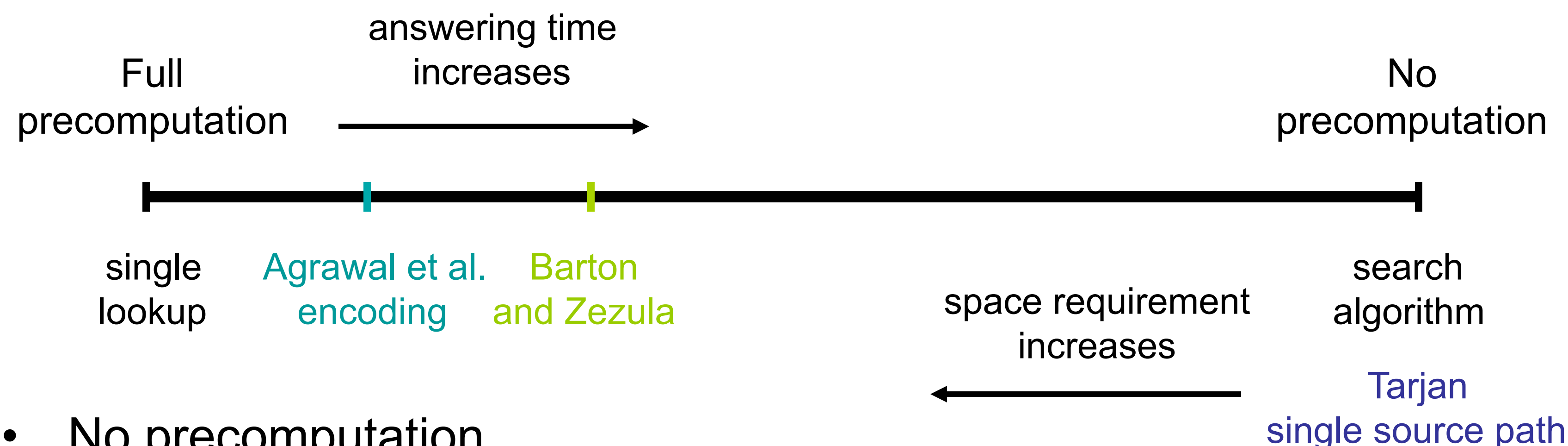
- No precomputation
  - Tarjan, single source path expression problem
- Precomputation
  - Agrawal et al., encode each path between any graph nodes

Answering "find a path" reachability queries



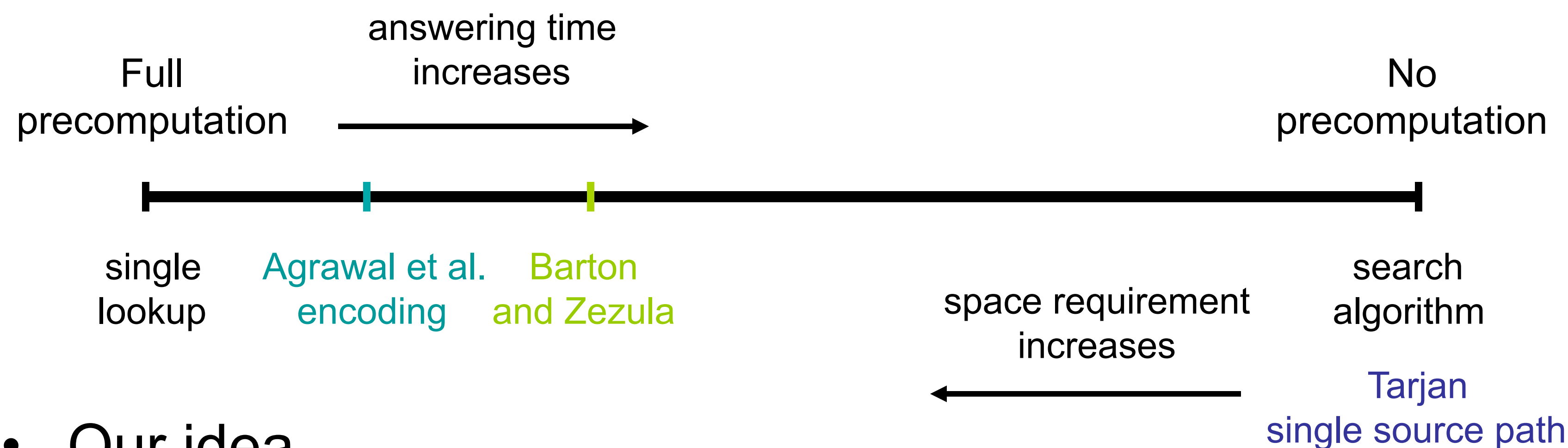
- No precomputation
  - Tarjan, single source path expression problem
- Precomputation
  - Agrawal et al., encode each path between any graph nodes
  - Barton and Zezula, graph segmentation - ρ-Index

Answering "find a path" reachability queries



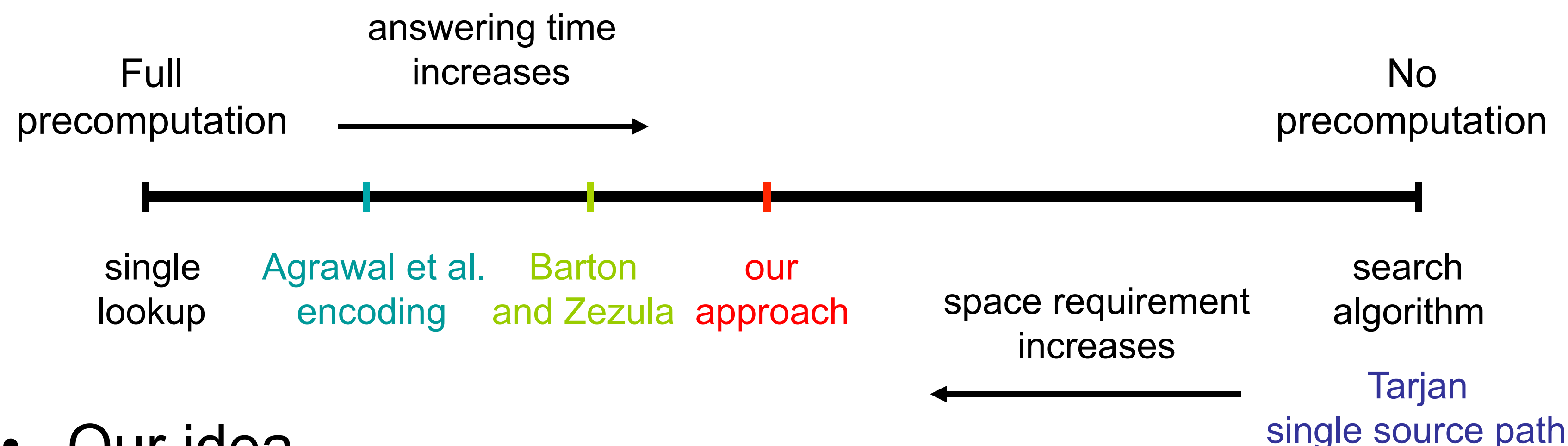
- No precomputation
  - Tarjan, single source path expression problem
- Precomputation
  - Agrawal et al., encode each path between any graph nodes
  - Barton and Zezula, graph segmentation - ρ-Index
- Labeling schemes
  - Determine whether exists a path, but cannot identify it

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

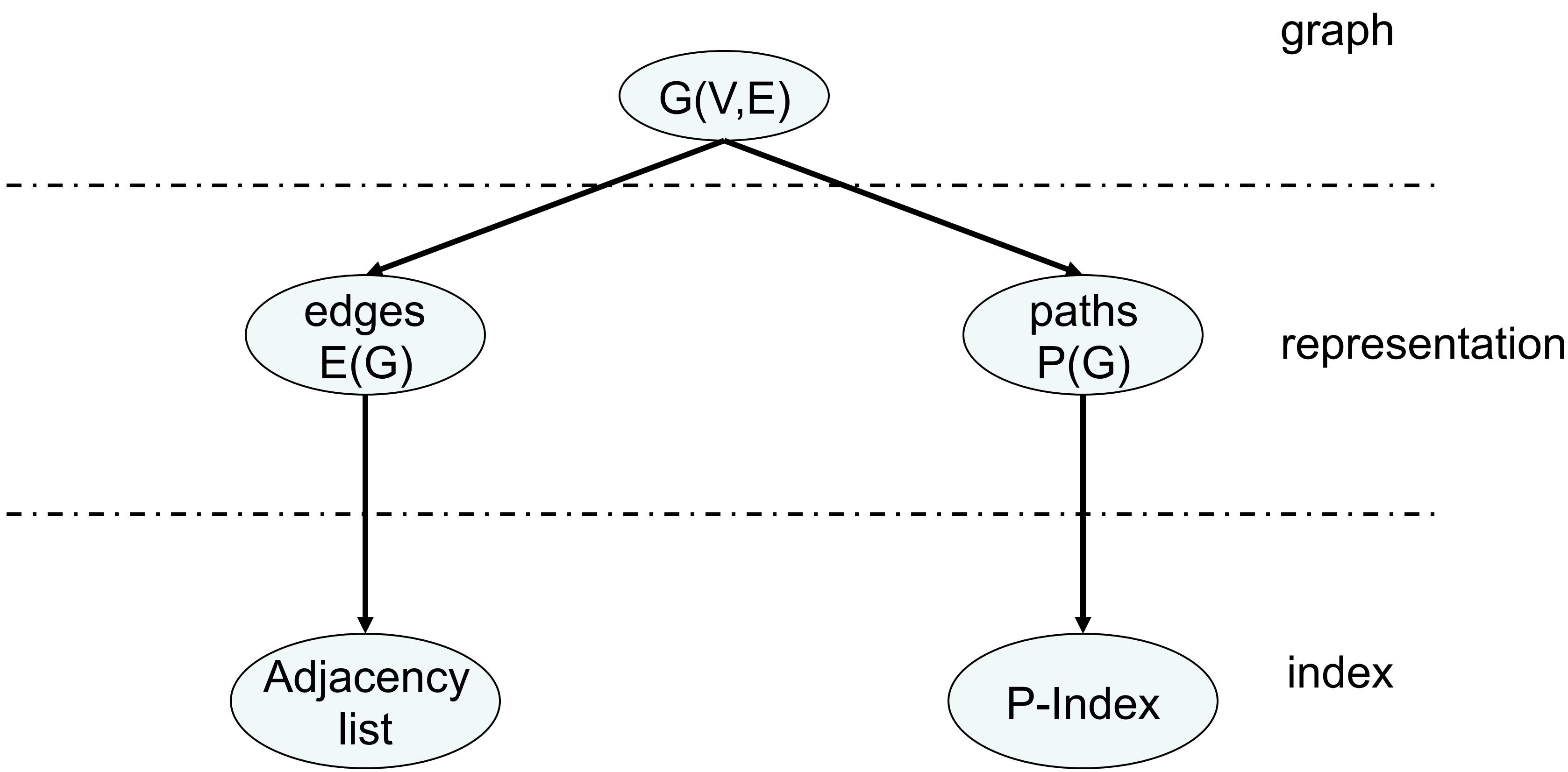


- Our idea
  - Represent the graph as a set of paths
  - Each path contains precomputed answers
  - Precompute and store part of path information in TC of the graph
- In the middle
  - No need to compute TC

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

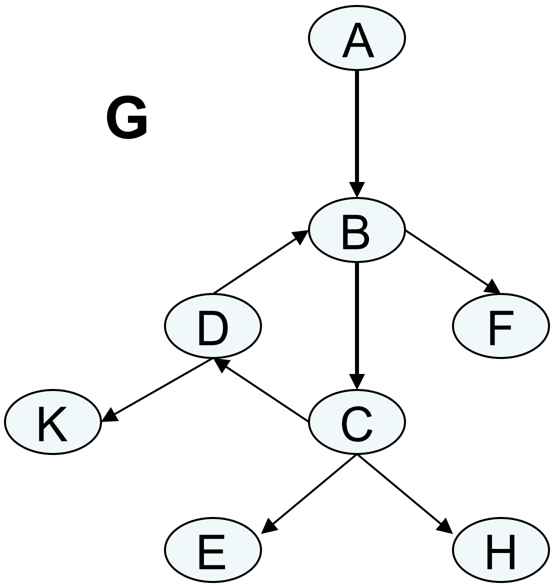
Graph – Representations - Indices



Path representation

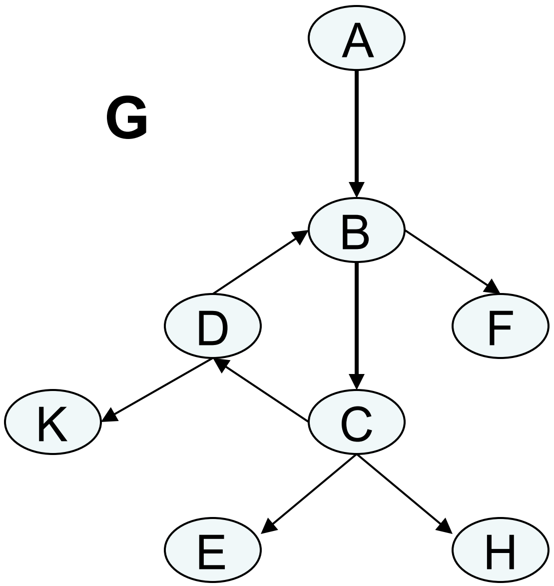
- Set of paths
  - Stores part of path information in TC of a graph
  - Combines graph edges to efficiently answer "find a path" reachability queries
  - Preserves reachability information
    - Each graph edge is contained in at least one path
    - Construct graph by merging paths
- Not unique

Path representation – Example



Path representation – Example

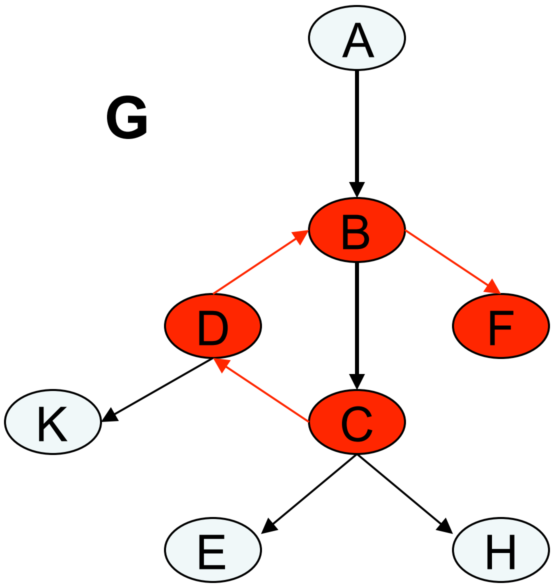
G



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

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

Path representation – Example



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

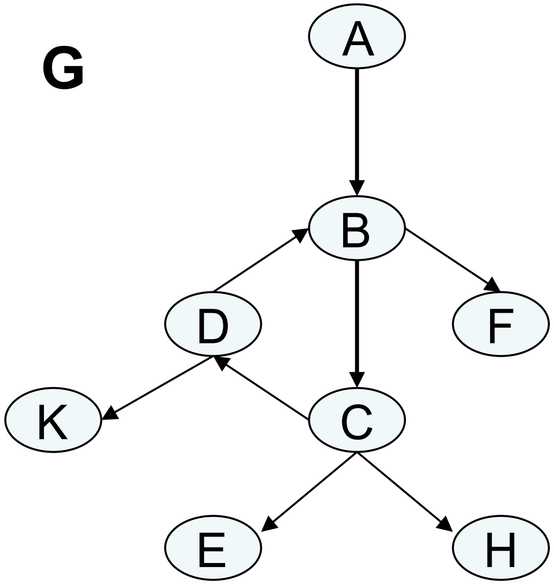
P(G) = {p1,p2,p3,p4}

P-Index

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

P-Index – Example

G

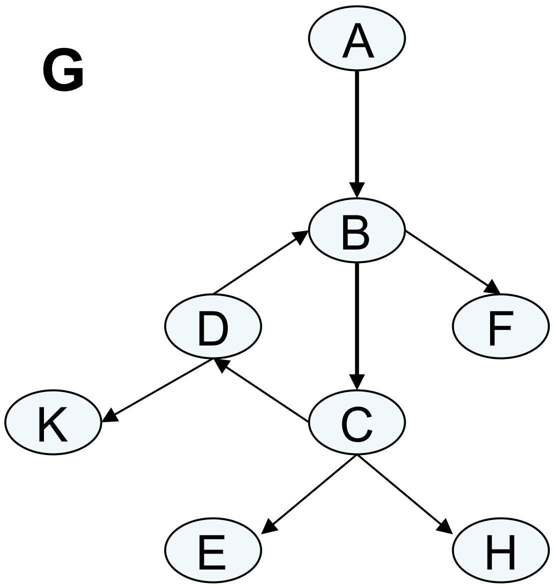


P(G)

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

P-Index – Example

G



P(G)

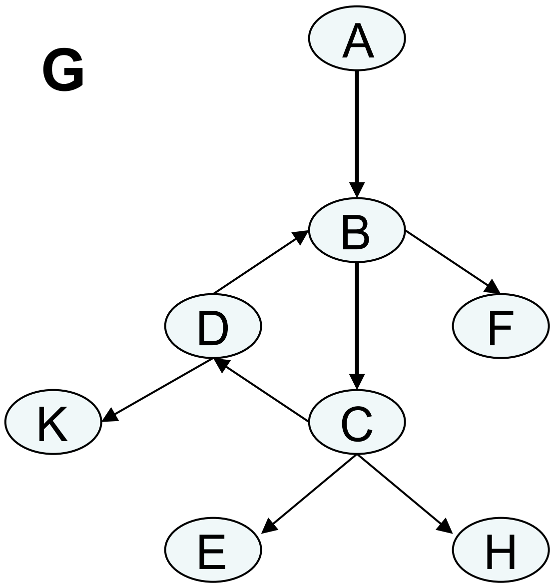
p1	(A,B,C,E)
p2	(C,D,B,F)
p3	(C,H)
p4	(D,K)

P-Index(G)

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

P-Index – Example

G



P(G)

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

P-Index(G)

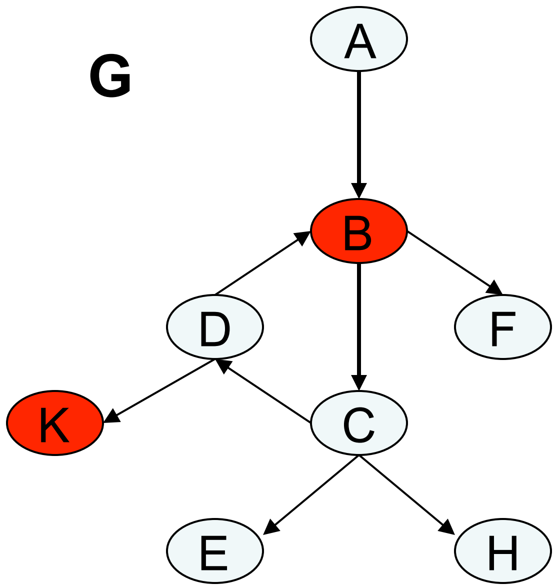
A	p1
B	p1, p2
C	p1, p2, p3
D	p2, p4
E	p1
F	p2
H	p3
K	p4

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

pdfs – Example

G



Query: FindAPath(B,K)

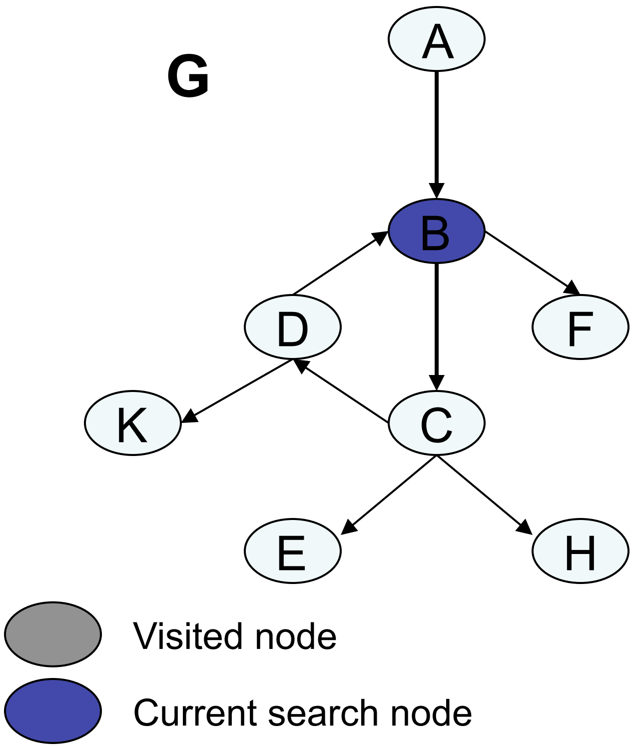
P(G)

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

pdfs – Example



• p1 contains B

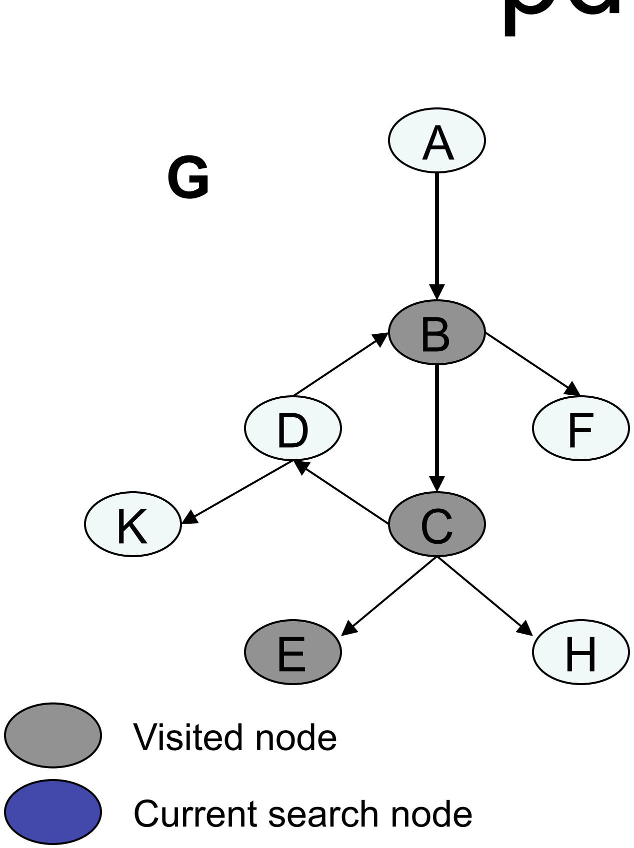
P(G)

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

pdfs – Example



- visit C,E
- no path in P(G) contains either C or E before target K

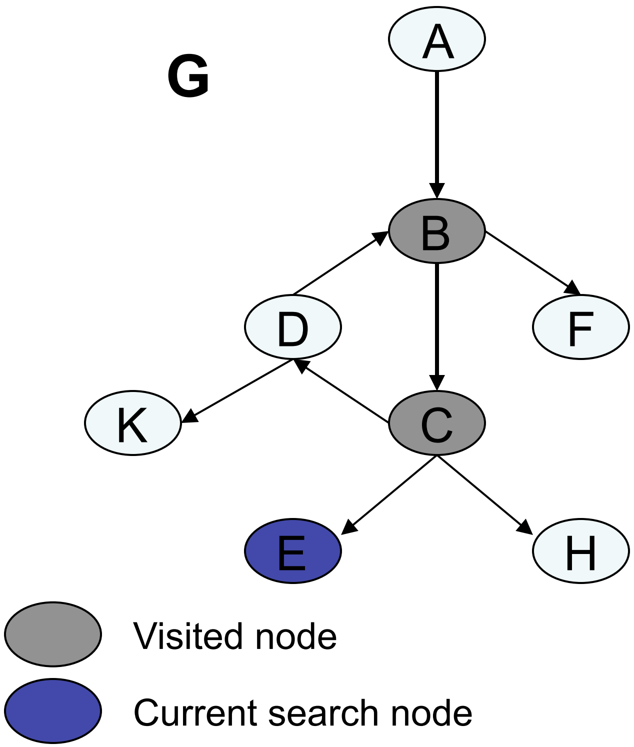
P(G)

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

pdfs – Example



- E contained in p1 at the end

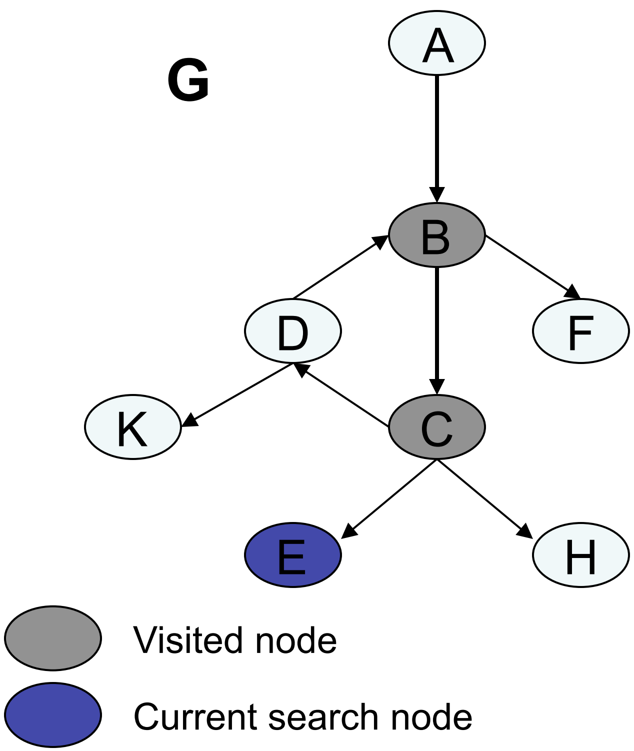
P(G)

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

pdfs – Example



- E contained in p1 at the end
- pop E

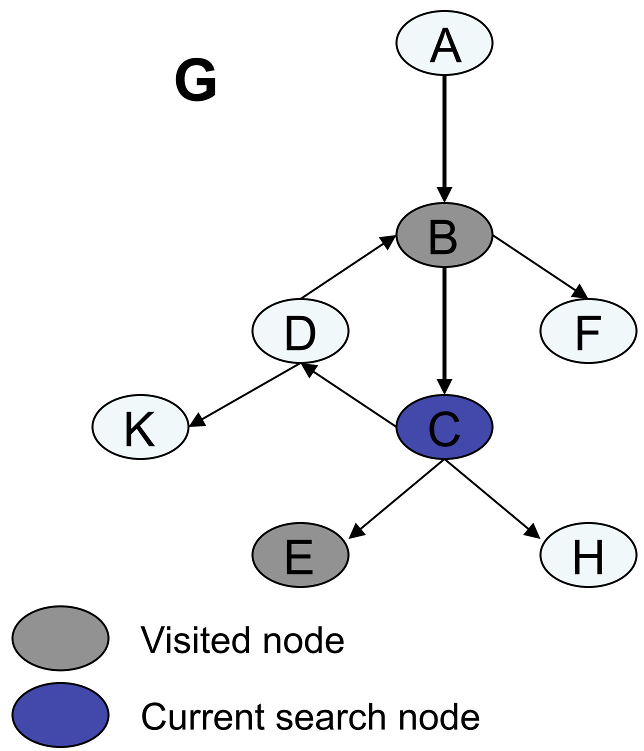
P(G)

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

pdfs – Example



- p1 contains C
- But E already visited, next path

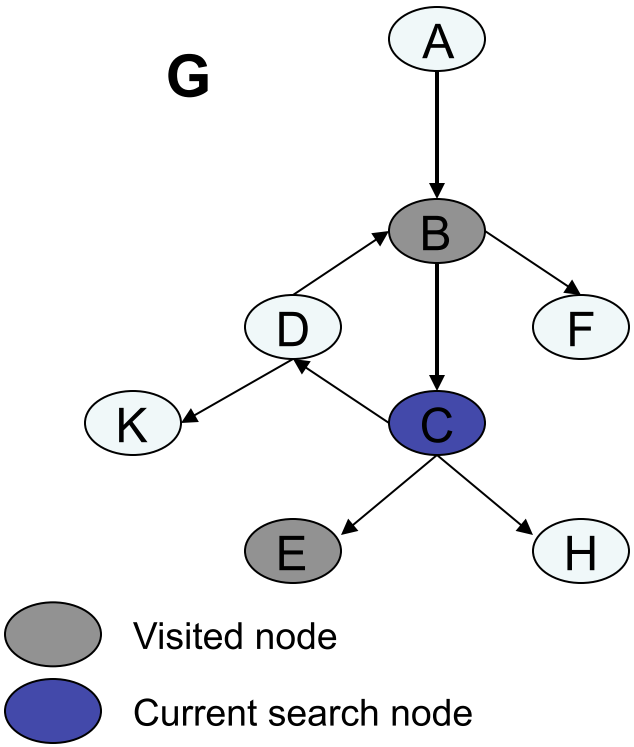
P(G)

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

pdfs – Example



- p1 contains C
- But E already visited, next path
- p2 contains C

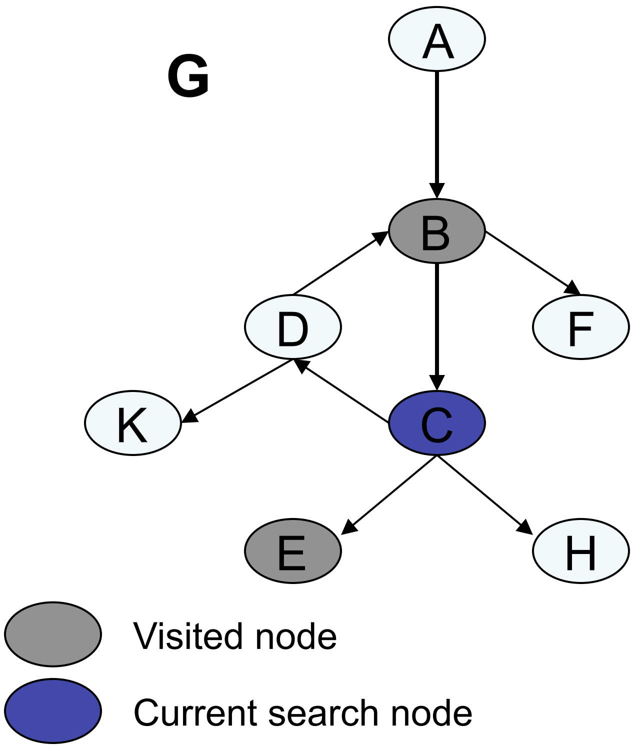
P(G)

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

pdfs – Example



- 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

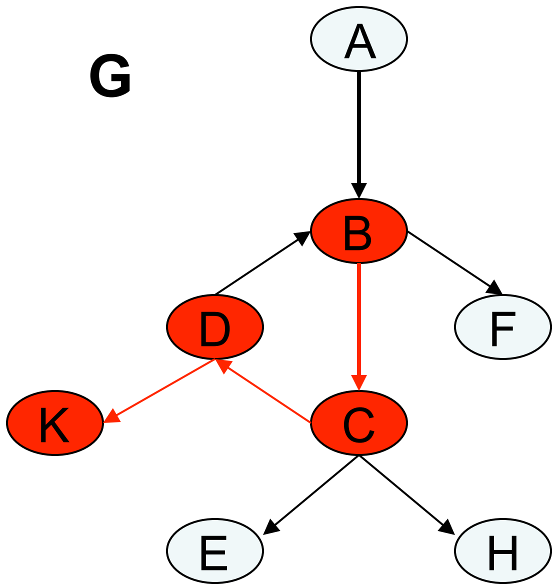
P(G)

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

pdfs – Example



FOUND path from B to K
(B,C,D,K)

P(G)

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

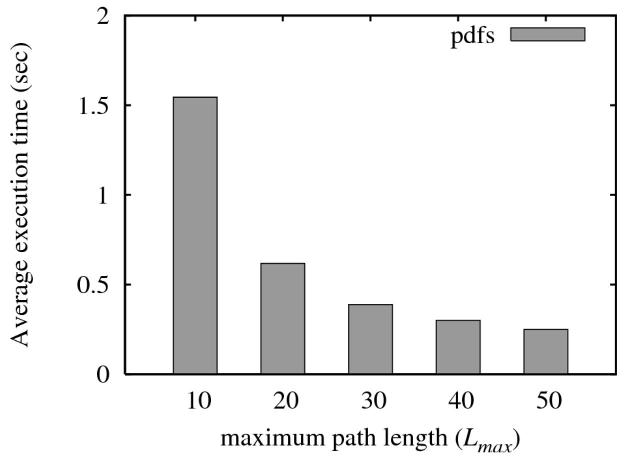
P-Index(G)

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

Experimental study

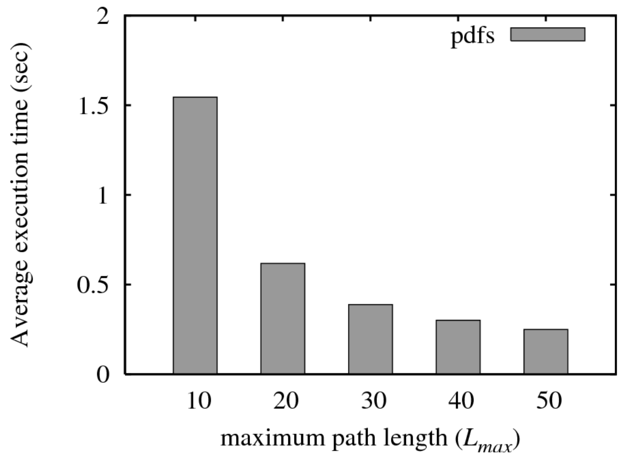
- Sets of random graphs
- Construct path representations
  - Traverse graph in depth-first manner starting from several nodes
  - Terminate when all graph edges included
  - Promote construction of long paths
    - Reusing graph edges
- Experimental parameters
  - Graph nodes $|V|$: 10^4, $5*10^4$, $\mathbf{10^5}$, $5*10^5$, 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



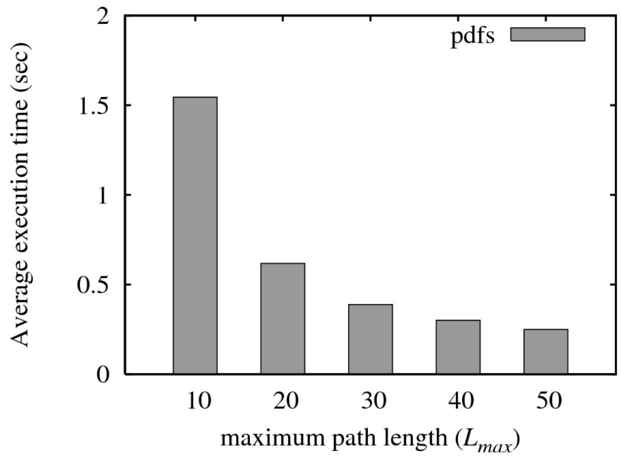
- Graph G: |V|=100,000 & d=4, 5 different path representations
- 1,000 “find a path” queries
- As L_{max} increases

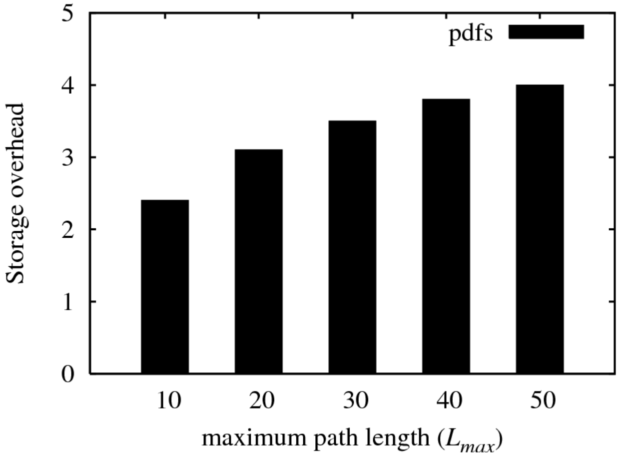
Varying max path length



- 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

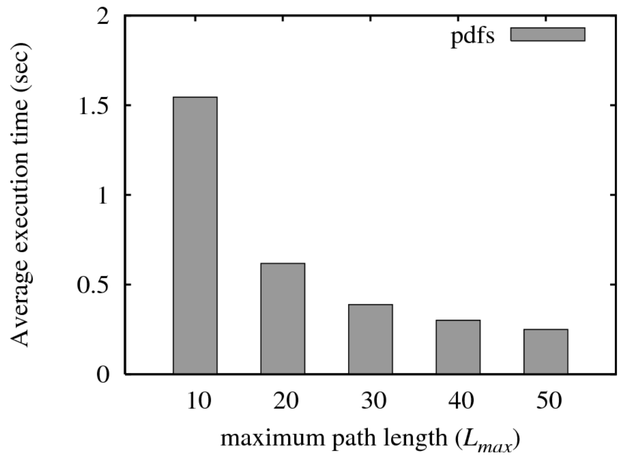
Varying max path length

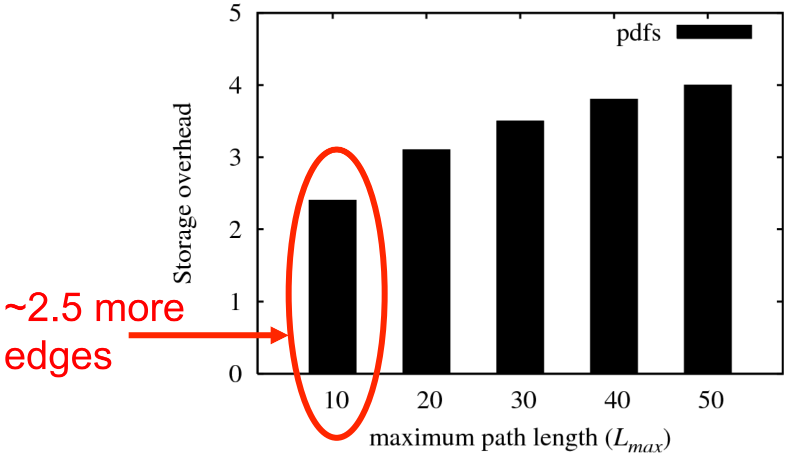




- Graph G: |V|=100,000 & d=4, 5 different path representations
- 1,000 “find a path” queries
- As L_{max} increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
  - Storage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

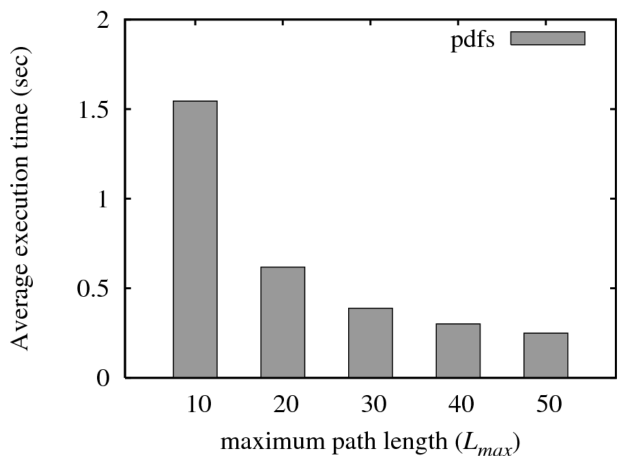
Varying max path length

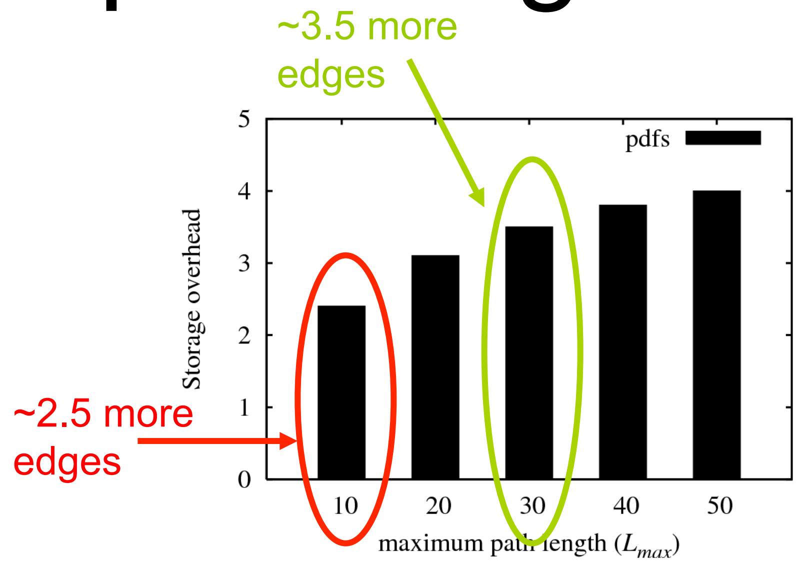




- Graph G: |V|=100,000 & d=4, 5 different path representations
- 1,000 "find a path" queries
- As L_{max} increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
  - Storage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

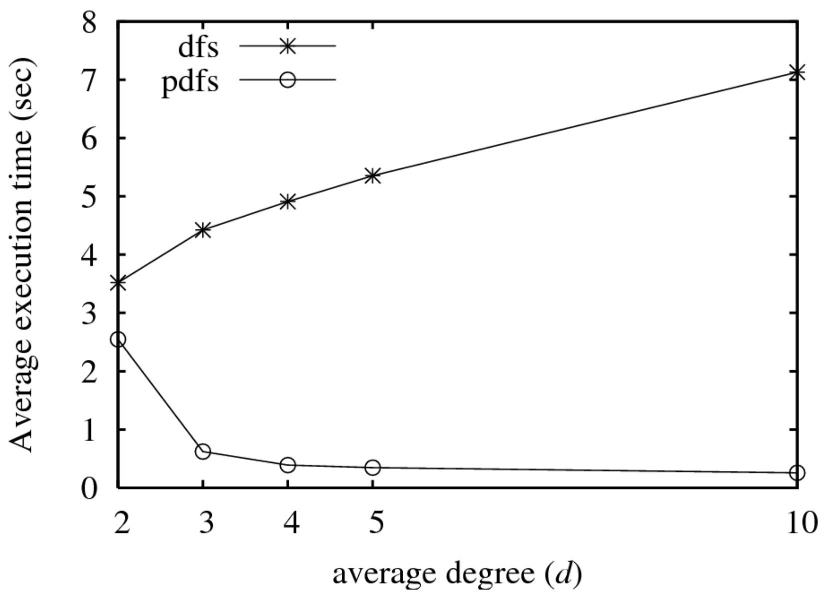
Varying max path length





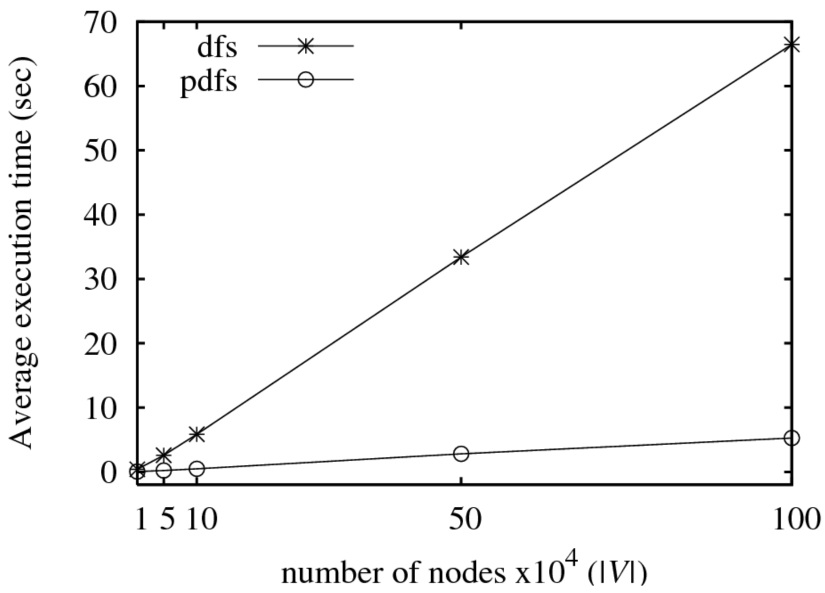
- Graph G: |V|=100,000 & d=4, 5 different path representations
- 1,000 “find a path” queries
- As L_{max} increases
  - Larger part of path information included
  - Fewer but longer paths
  - pdfs visits more nodes in each iteration
  - More possibly exists path where node u before target
  - Storage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

Varying avg degree



- Initial graph G: $|V|$ =100,000 & d=2 & L_{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 ?

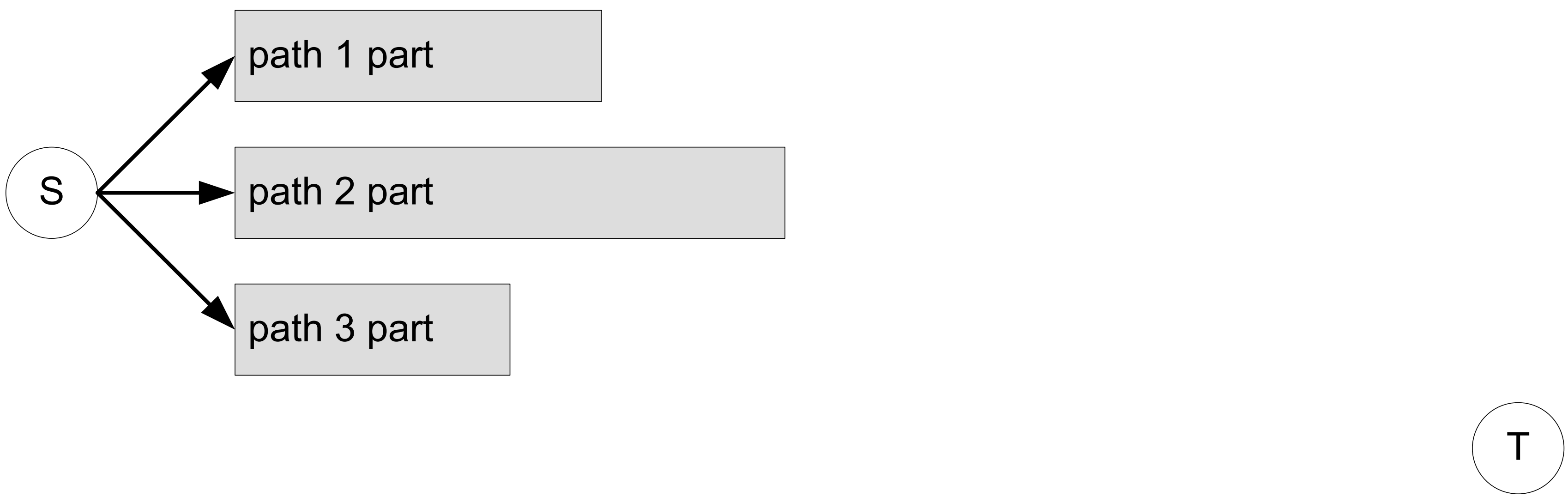
Evaluating “find a path” reachability queries

Additional slides

Answering queries – Basic idea

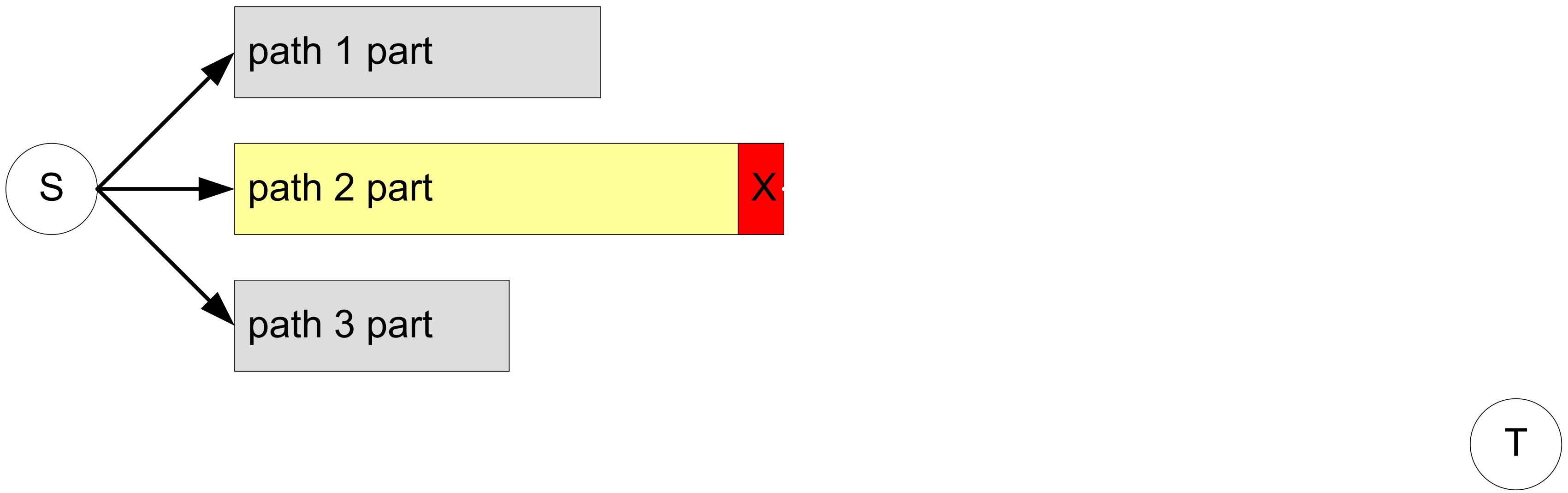
Find a path from S to T

Answering queries – Basic idea



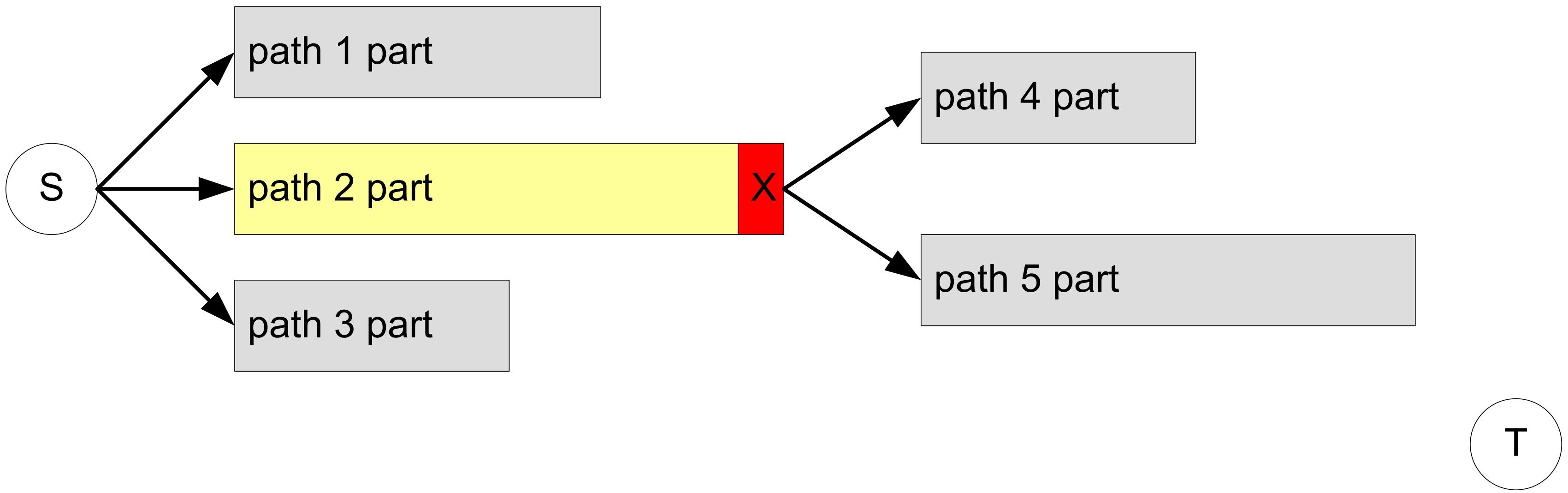
S contained in p_1,p_2,p_3

Answering queries – Basic idea



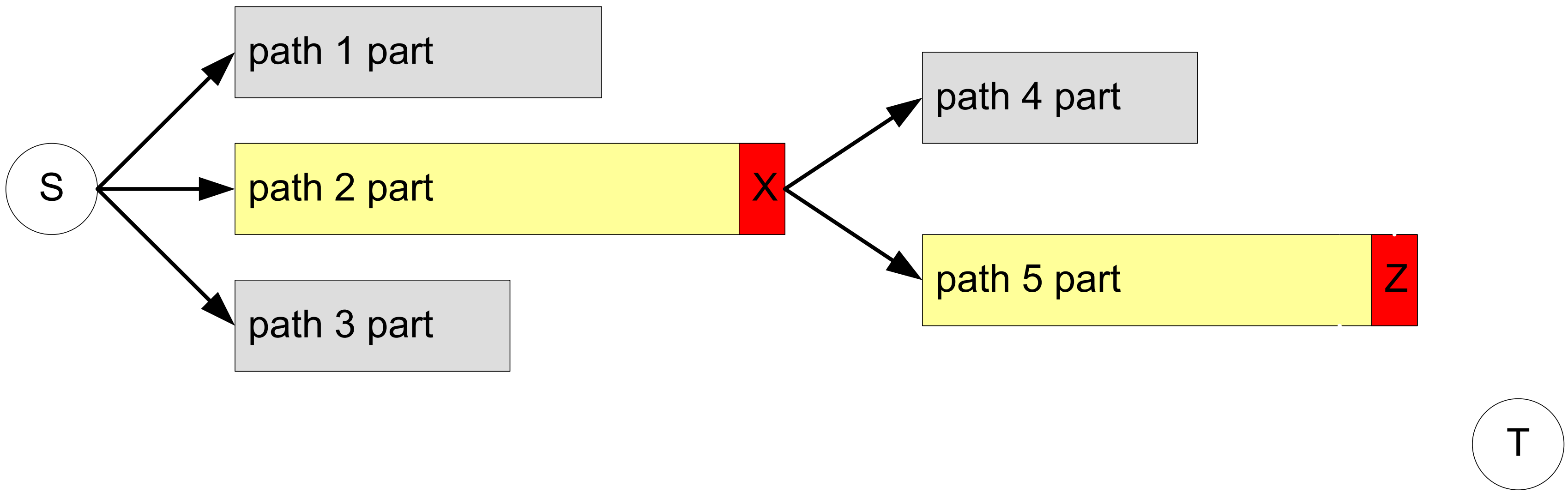
Consider p_2 part – X last node

Answering queries – Basic idea



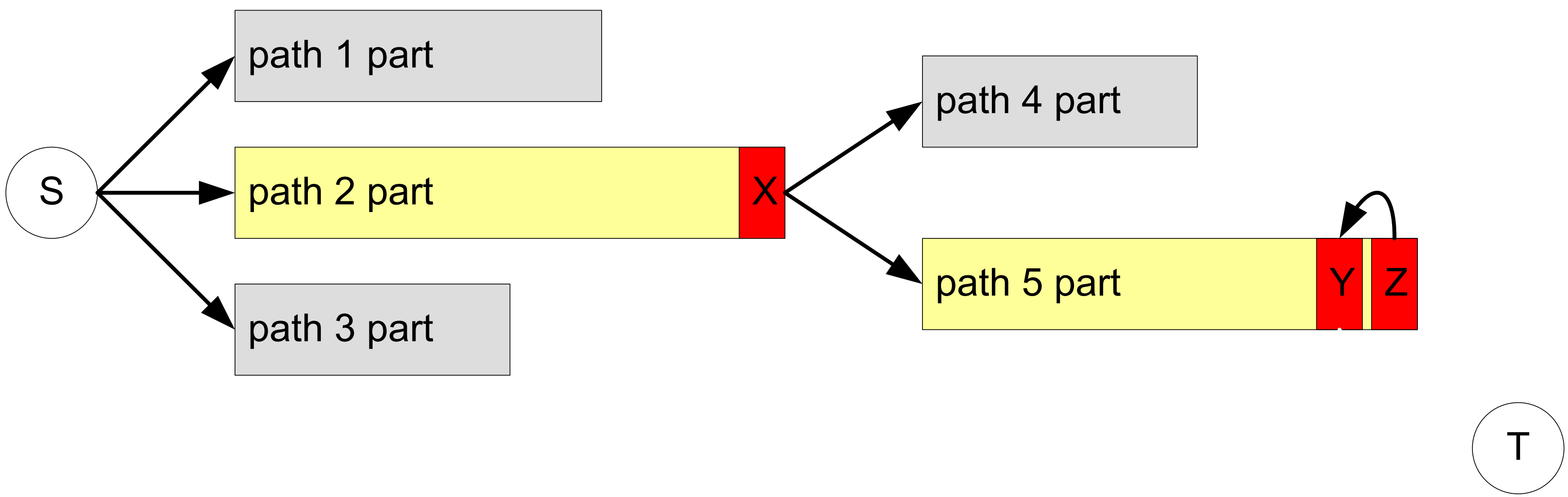
S contained in p_4,p_5

Answering queries – Basic idea



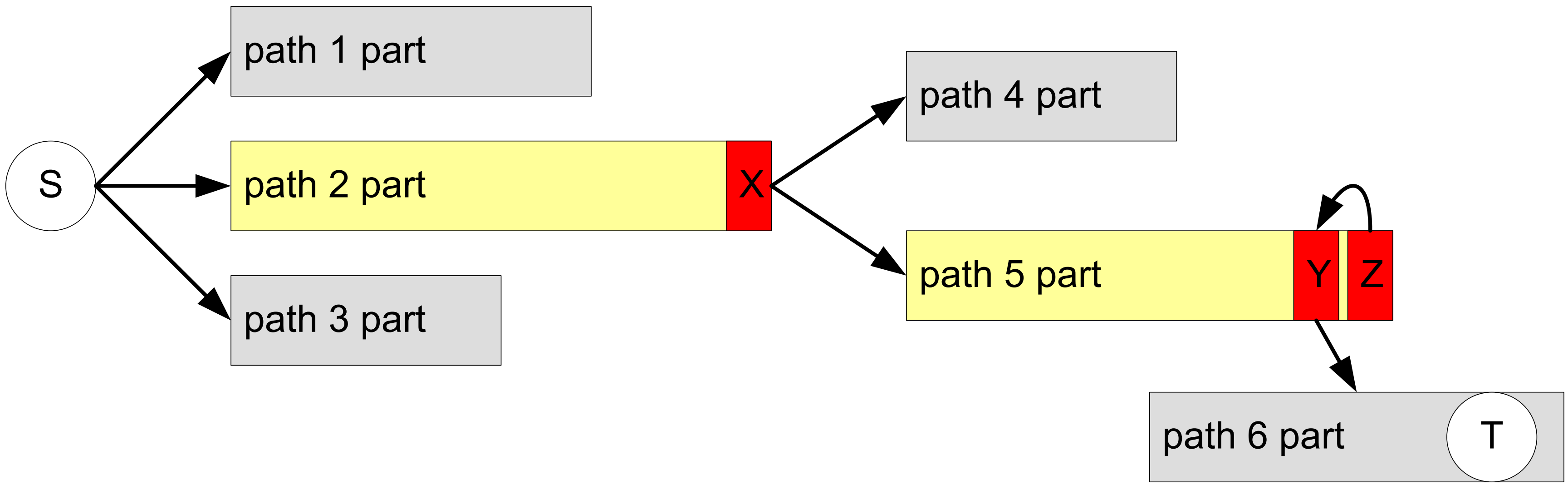
Consider p_5 part – Z last node

Answering queries – Basic idea



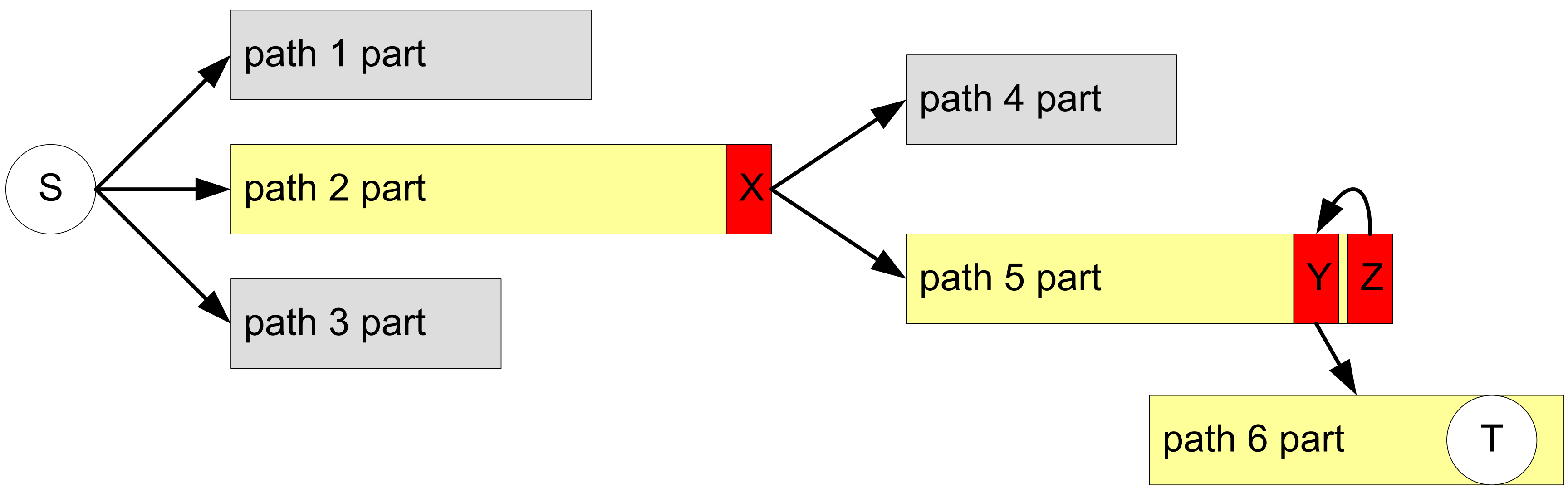
Z only contained in p_5 – backtrack to Y

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