Efficient Snapshot Retrieval over Historical Graph Data

Udayan Khurana, Amol Deshpande
University of Maryland, College Park
{udayan, amol}@cs.umd.edu

Agenda

- What
  - Historical Graph Data
  - Snapshot Retrieval
- Why
  - Motivation
- How
  - GraphPool
  - DeltaGraph
- Experiments

Historical Graph Data

- A collection of graph snapshots
  - A set of nodes and a set of edges
    - A list of attribute-value pairs
- Event
  - Creation or deletion of an edge or node
  - Change in an attribute value of a node or an edge

```
(a) {NE, N:23, N:4590, directed:no, 11/29/03 10:10}
(b) {UNA, N:23, 'job', old:'..', new:'..', 11/29/07 17:00}
```

$$G_k = G_{k-1} + E, \quad G_{k-1} = G_k - E$$

Snapshot Retrieval

- Singlepoint
  - **GetHistGraph(Time t, String attr options)**
- Multipoint
  - **GetHistGraphs(List<Time> t list, String attr options)**
  - TimeExpression
  - Time interval

Option	Explanation
-node:all (**default**)	None of the node attributes
+node:all	All node attributes
+node:attr1	Node attribute named "attr1"; overrides "-node:all" for that attribute
-node:attr1	Node attribute named "attr1"; overrides "+node:all" for that attribute

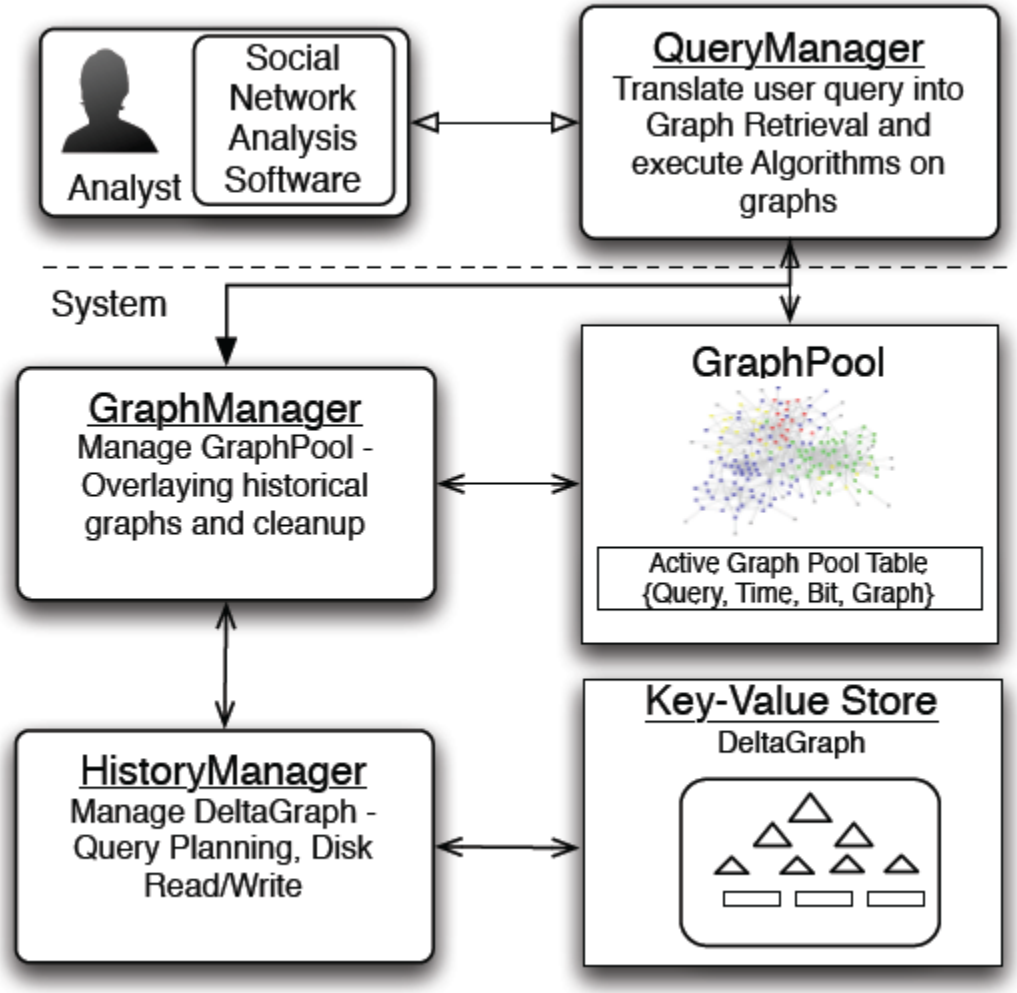
Snapshot Retrieval

```
/* Loading the index */
GraphManager gm = new GraphManager(. . . );
gm.loadDeltaGraphIndex(. . . );
. . .
/* Retrieve the historical graph structure along with node names as of
Jan 2, 1985 */
HistGraph h1 = gm.GetHistGraph("1/2/1985", "+node:name");
. . .
/* Traversing the graph*/
List<HistNode> nodes = h1.getNodes();
List<HistNode> neighborList = nodes.get(0).getNeighbors();
HistEdge ed = h1.getEdgeObj(nodes.get(0), neighborList.get(0));
. . .
/* Retrieve the historical graph structure alone on Jan 2, 1986 and Jan
2, 1987 */
listOfDates.add("1/2/1986");
listOfDates.add("1/2/1987");
List<HistGraph> h1 = gm.getHistGraphs(listOfDates, "");
. . .
```

Motivation

- To support a broad range of network analysis tasks
- The existing solutions for graph data management lack adequate techniques for temporal annotation

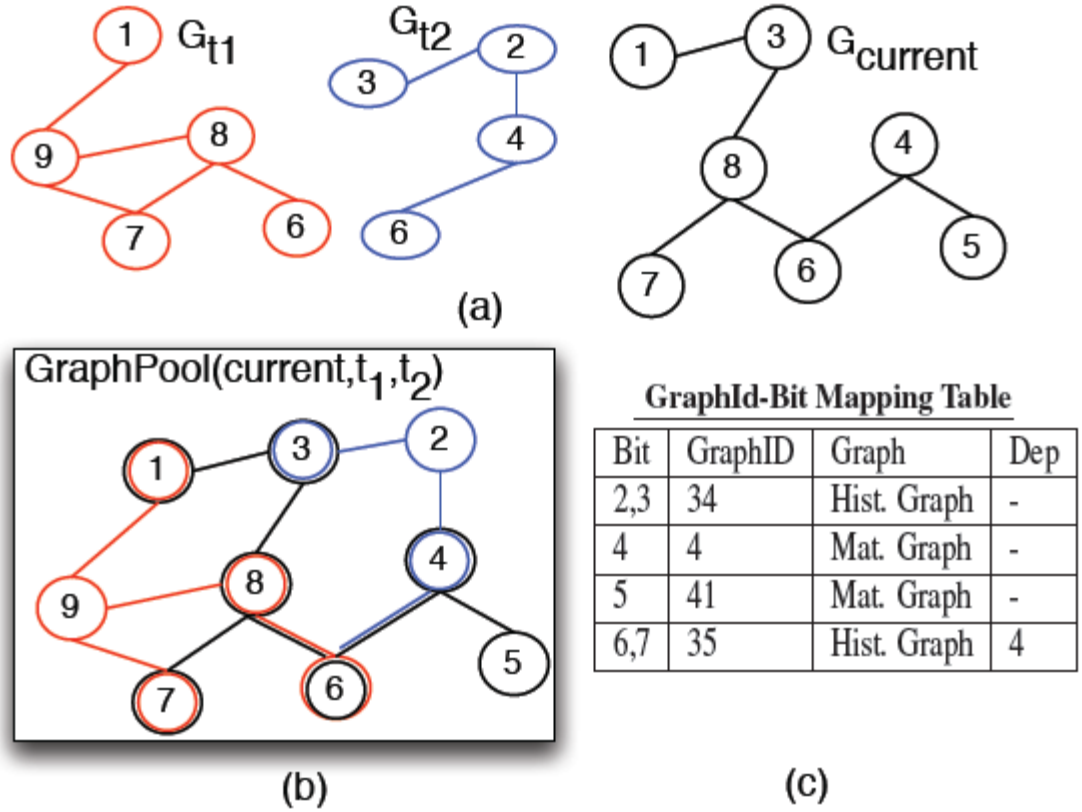
System Architecture



GraphPool

- An in-memory graph data structure in an overlapping manner
- A single graph that is the union of
  - current graph
  - historical snapshots
  - materialized graphs
- GraphID-Bit Mapping Table
- Update and Clean-up

GraphPool



GraphId-Bit Mapping Table

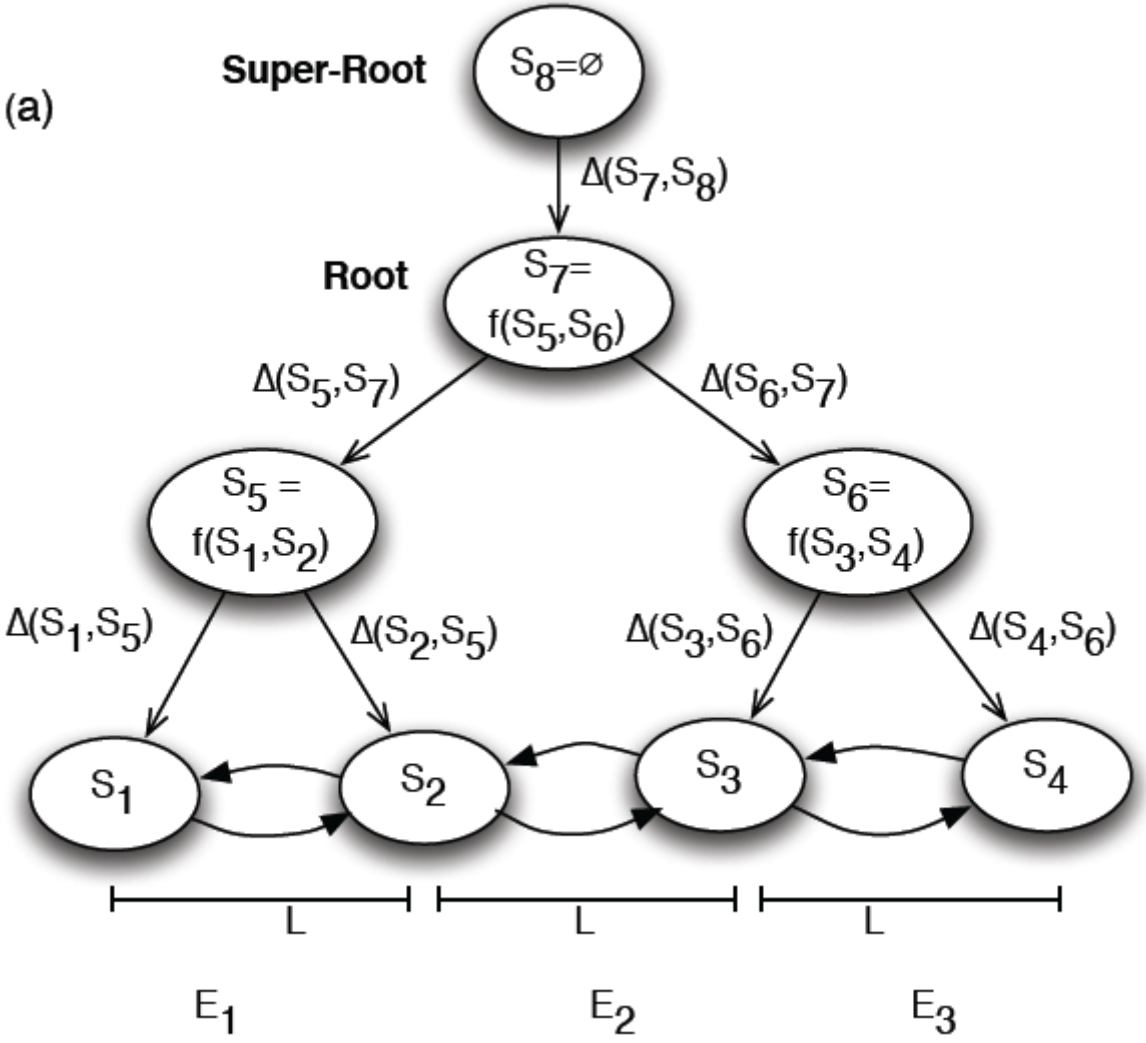
Bit	GraphID	Graph	Dep
2,3	34	Hist. Graph	-
4	4	Mat. Graph	-
5	41	Mat. Graph	-
6,7	35	Hist. Graph	4

(c)

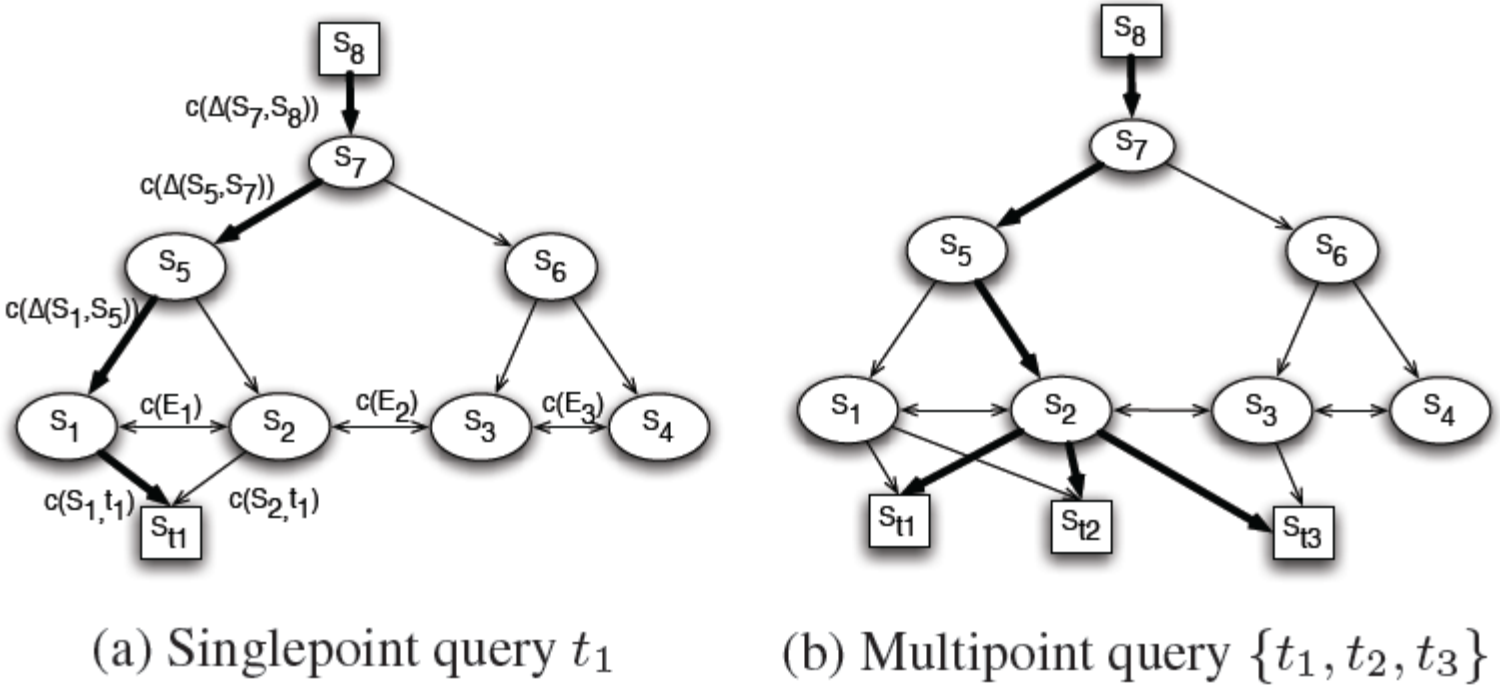
DeltaGraph

- A novel, extensible, highly tunable, and distributed hierarchical index structure
- Prior Techniques
  - External interval tree, Segment trees, Snapshot index, Copy+Log
  - Not efficiently support multipoint queries, Not highly tunable, Not support different persistent storage options

DeltaGraph


(a)
Super-Root
$S_8=\varnothing$
$\Delta(S_7,S_8)$
Root
$S_7=$
$f(S_5,S_6)$
$\Delta(S_5,S_7)$
$\Delta(S_6,S_7)$
$S_5=$
$f(S_1,S_2)$
$S_6=$
$f(S_3,S_4)$
$\Delta(S_1,S_5)$
$\Delta(S_2,S_5)$
$\Delta(S_3,S_6)$
$\Delta(S_4,S_6)$
S_1
S_2
S_3
S_4
L
L
L
E_1
E_2
E_3

DeltaGraph



(a) Singlepoint query t_1 (b) Multipoint query $\{t_1, t_2, t_3\}$

DeltaGraph

- Model of Graph Dynamics

$$|G_{|E|}| = |G_0| + |E| \times \delta_* - |E| \times \rho_*$$

- Differential Functions

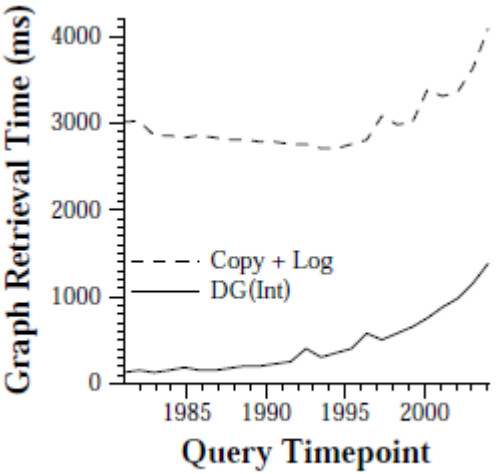
Table 2: Differential Functions

Name	Description
Intersection	$f(a, b, c \dots) = a \cap b \cap c \dots$
Union	$f(a, b, c \dots) = a \cup b \cup c \dots$
Skewed	$f(a, b) = a + r.(b - a), 0 \leq r \leq 1$
Right Skewed	$f(a, b) = a \cap b + r.(b - a \cap b), 0 \leq r \leq 1$
Left Skewed	$f(a, b) = a \cap b + r.(a - a \cap b), 0 \leq r \leq 1$
Mixed	$f(a, b, c \dots) = a + r_1.(\delta_{ab} + \delta_{bc} \dots) - r_2.(\rho_{ab} + \rho_{bc} \dots), 0 \leq r_2 \leq r_1 \leq 1$
Balanced	$f(a, b, c \dots) = a + \frac{1}{2}.(\delta_{ab} + \delta_{bc} \dots) - \frac{1}{2}.(\rho_{ab} + \rho_{bc} \dots)$
Empty	$f(a, b, c \dots) = \emptyset$

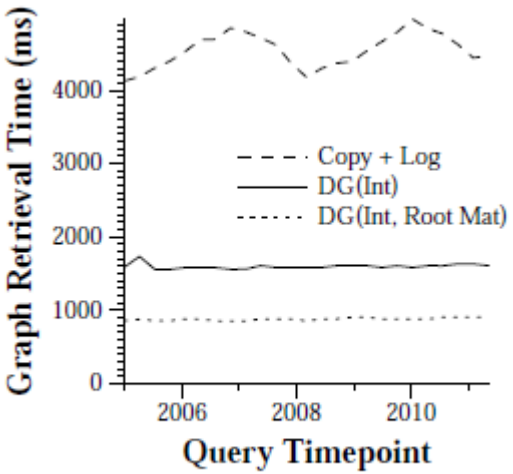
Experiments

- Implementation
  - Java
  - Kyoto Cabinet key-value store
- Datasets
  - DBLP
  - Randomly generated small
  - Randomly generated large

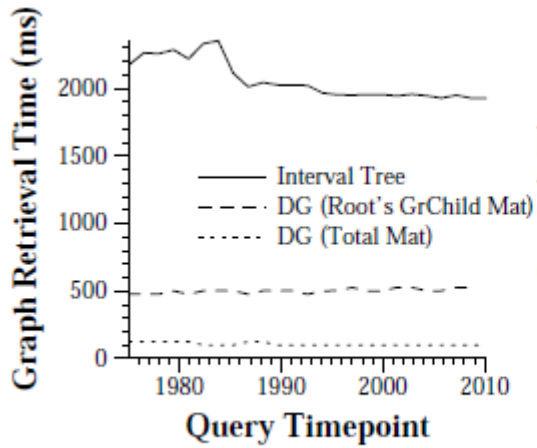
Experiments



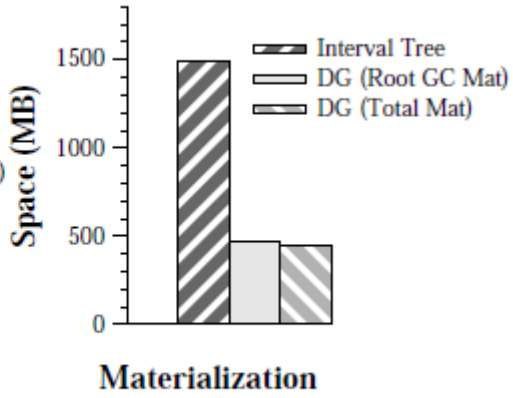
(a) Performance: Dataset 1



(b) Performance: Dataset 2

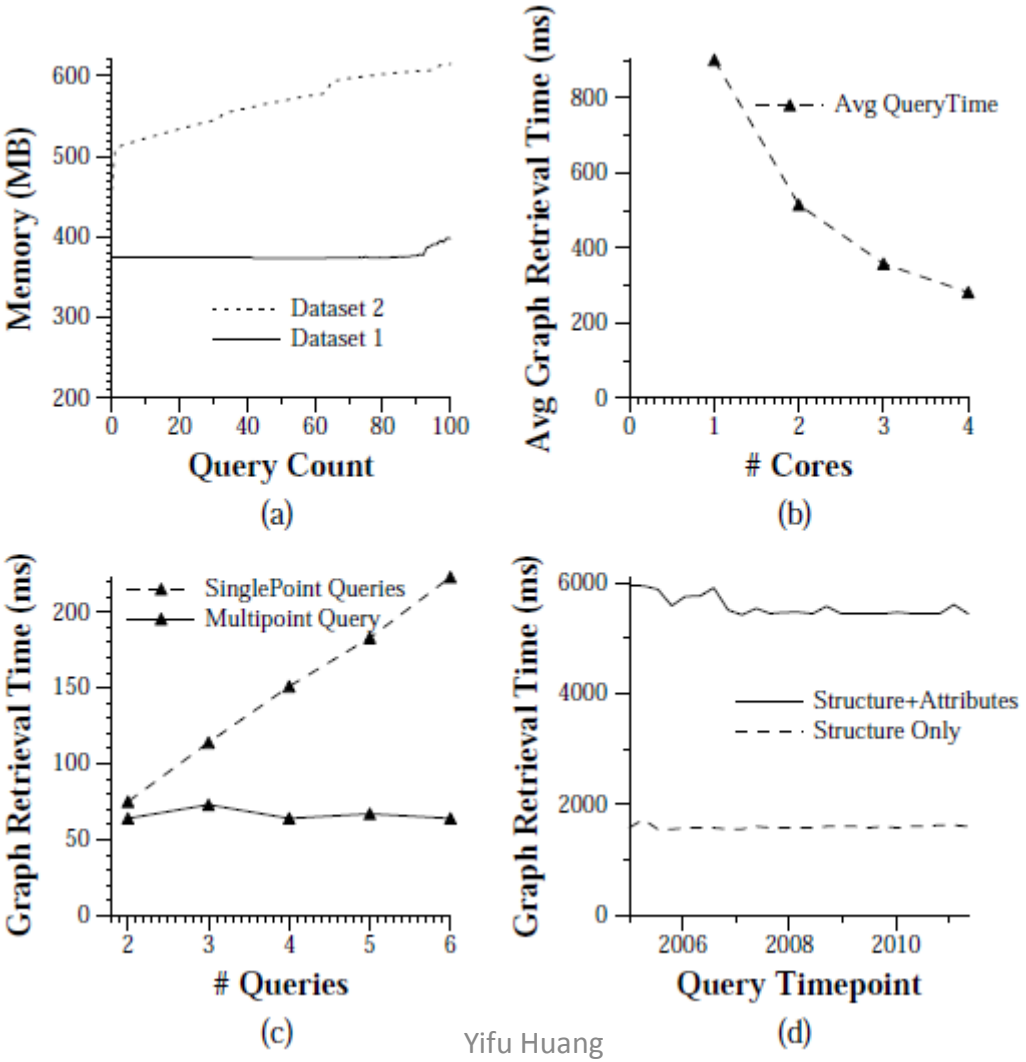


(a) Performance: Dataset 2

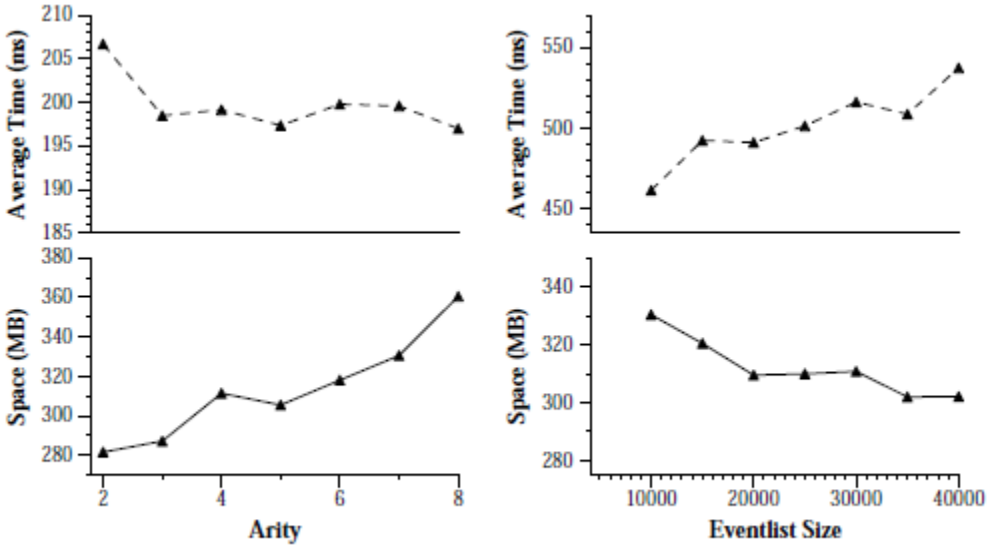


(b) Index Memory: Dataset 2

Experiments

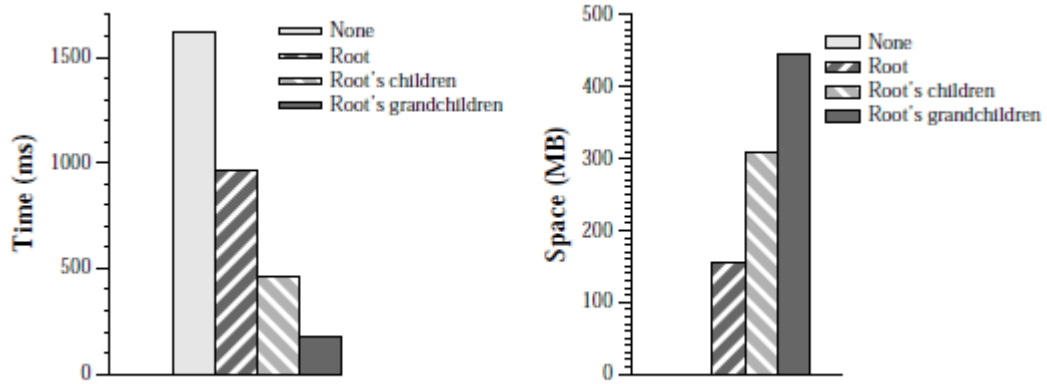


Experiments



(a) Varying Arity (Dataset 1)

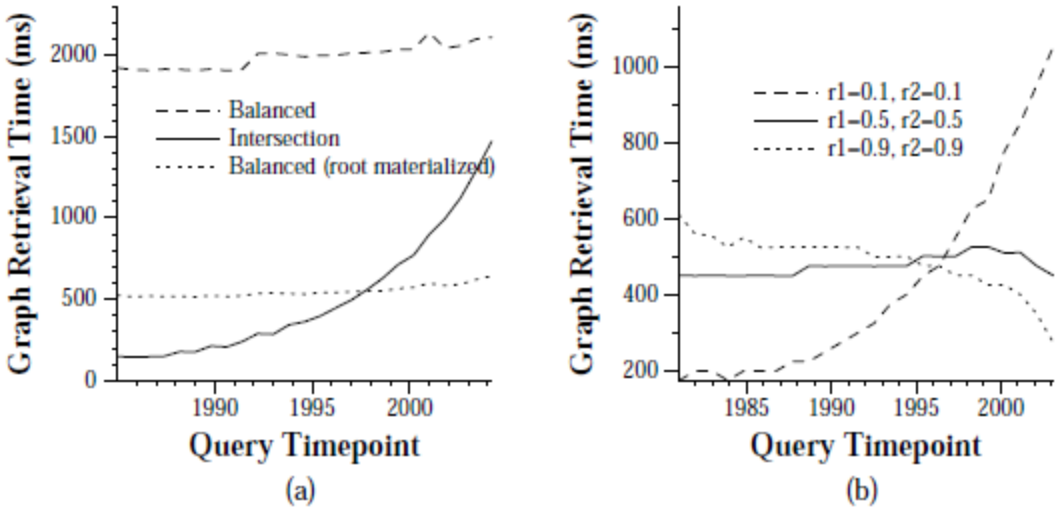
(b) Varying EventList Size (Dataset 1)



(a) Average Query Time

(b) Materialization Memory Requirement

Experiments



Thank You!