Fast Reachability Computation on Big Attributed Graphs

BigGraph@IEEE BigData Conference 2016

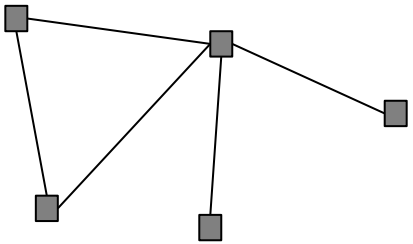
Duncan Yung and Shi-Kuo Chang

Department of Computer Science

Table of Contents

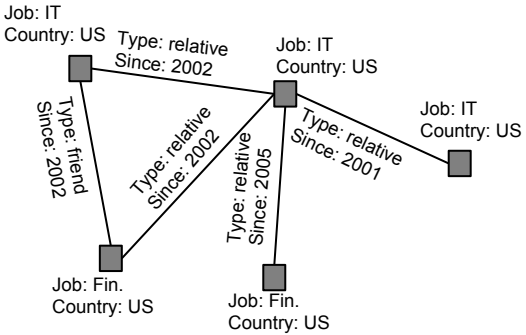
Research Problem

Graph



Topology

Attributed Graph



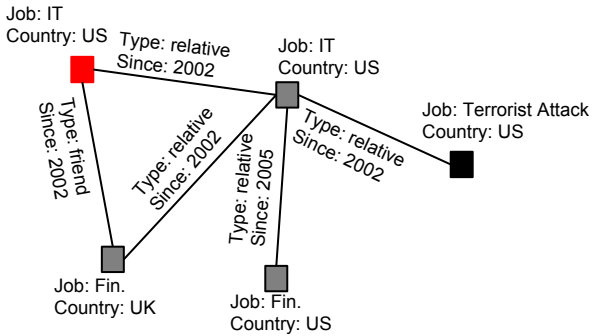
Topology in primary storage
Attributes in secondary storage
(proposed by MSR people in CIKM12)

An Attributed Graph Query Example

- **Attribute Constrained Reachability Query:**
  - whether there is a *path* from *Duncan* (red vertex) to a *terrorist* (black vertex) s.t. all *country=US* and *year=2002* on the path
  - Answer is **YES**

- **Baseline:**
  - BFS/DFS
  - Access secondary storage for attributes when visit

- **Optimization Goal:**
  - Efficiency!
  - i.e. execution time -> **reduce Sec. Storage Access**



Topology in primary storage
Attributes in secondary storage
(proposed by MSR people in CIKM12)

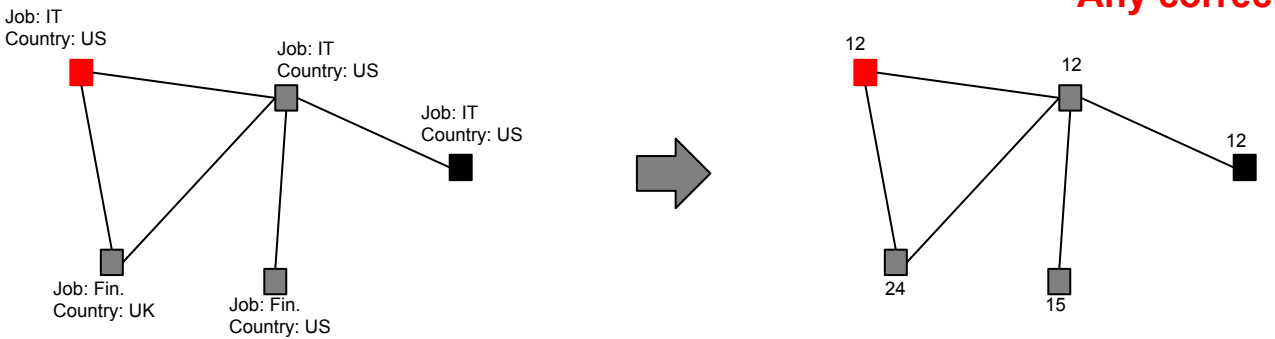
Contribution 1:

New Attribute Verification Approach

Use of "Perfect" Hashing

- **Goal:**
  - **Reduce Secondary Storage Access**
  - How?
    - Use hash values to represent attributes
    - Put hash values in memory
    - **Compare attribute constraint hash value with attribute hash value**
- **Example:**
  - *Point* Attr. Constraint={*Job=IT*,*Country=US*} -> hash{Job=IT,*Country=US*}=12

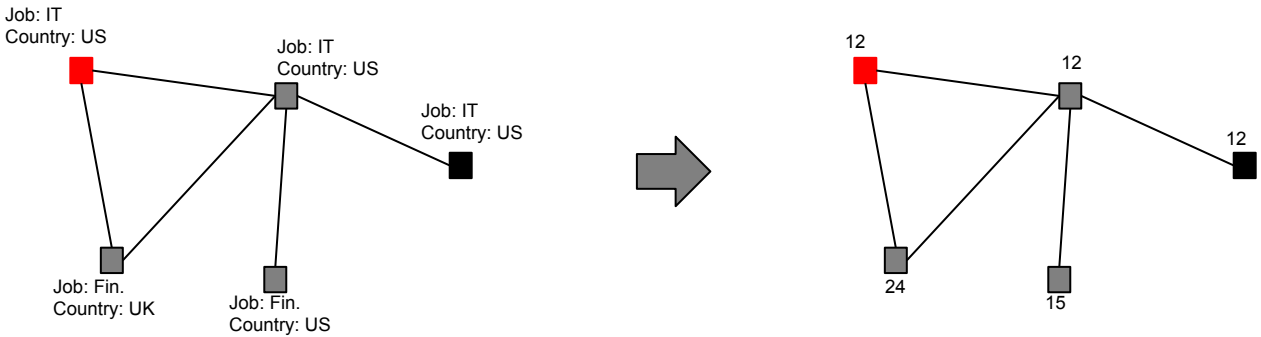
Any correctness problem?



Use of "Perfect" Hashing

- Hash value **collision** may happen!
  - i.e. 2 different attributes map to the same hash value

- When hash value comparison is valid?
  - **Theorem 1:** a hash value has to satisfy a few conditions!



Theoretical Result for Sec. Storage Access

- **Worst Case I/O:** O(|V|+|E|)
  - All attributes map to the same hash value.

- **Theorem 2: O(1) Expected I/O for Point Attr. Constraint Query**
  - **Optimal** for this setting!

- **Theorem 3: O(A_{diff}) Expected I/O for Set Attr. Constraint Query**
  - A_{diff}: number of different attr. visited

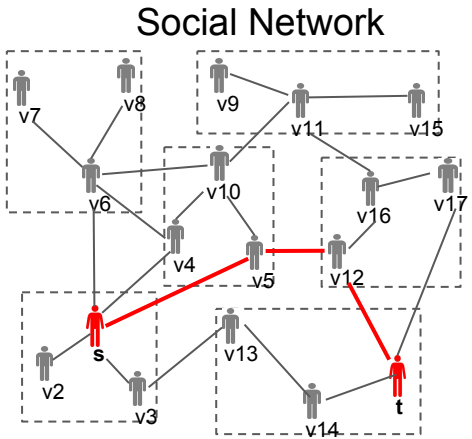
Contribution 2:

Heuristic Search Technique

Heuristic Search for Reachability Query

- **Motivation**
  - Expected I/O
    - Point Attr. Constraint Query: O(1)
    - Set Attr. Constraint Query: $O(A_{diff})$
  - Reduce A_{diff}

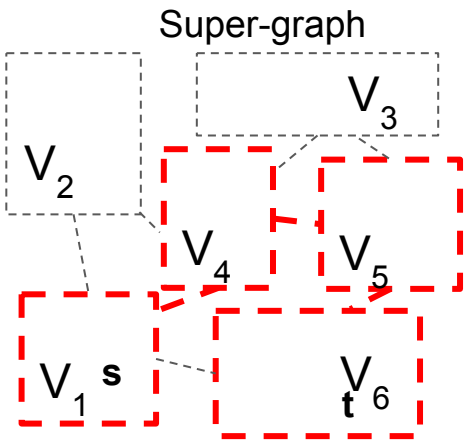
- **Intuition**
  - Find a short constraint satisfy path



Heuristic Search for Reachability Query

- **Idea:**
  - Traverse regions that are:
    - likely to pass through first and
    - near to destination

- **Implementation**
  - find cluster shortest path based on:
    - attr. constraint distribution
    - distance to destination
  - Constrained graph traversal in cluster shortest path



Why not existing reachability index?

Existing Reachability Index

- **Reachability Index:**
  - Only answer Yes/No
  - No attribute information maintained
  - High index construction complexity and storage space
    - Not work for Big Graph

- **Reachability Query with Constraints**
  - Can only handle single label on edge
  - High index construction time and storage space
    - Not work for Big Graph

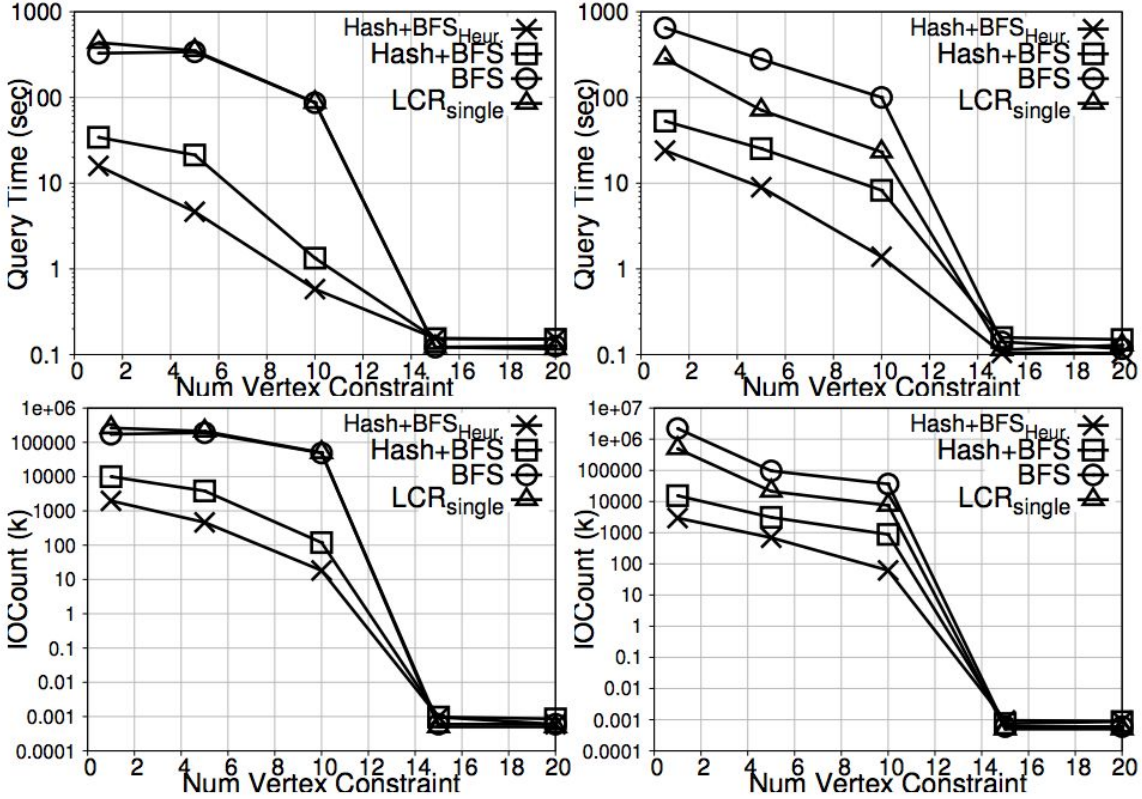
Sketch of Experimental Result

Experiment Setup

- **Datasets**
  - **Real Graph:**
    - [twitter-0.25] 52m vertices, 490m edges
    - [fb-bfs1] 1m vertices, 29m edges
  - **Synthetic Graph:** up to 200m vertices, 1b edges

- **Experiment Design:**
  - Vary:
    - number of vertex/edge attribute constraint
    - attribute domain size
    - number of attribute

- **Report:** Avg. and Max. Time and I/O Count



(a)Num V Attr.=10 (b)Num V Attr.=30

Fig. 6. **[twitter-0.25]**-Vary # of V Const. with Org. Dom.

Thank You

Questions?

Hashing Scheme for I/O Bound

- **Example of Theorem 1**
  - Suppose attr. Constraint C_V={*Job=IT Country=US*}
  - Hash value comparison is valid if:
    i. hash(C_V)=hash(vertex attr.)
    ii. Only 1 attribute map to this hash value
      - i.e. only hash(*Job=IT Country=US*)=12 in G
    iii. *Job=IT Country=US* is in G