Scalable Network Processing for Social Networks

Srikanta Bedathur

based on work jointly done with Stephan Seufert, Avishek Anand and Gerhard Weikum

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

My Research

- Navigating and Exploration of Large (Social) Graphs
  - Scaling reachability to million node graphs and beyond
  - Fast and accurate shortest paths
  - Modeling dynamic/temporal graphs
  - Visualization of dynamics in large graphs
- Combining social content with graph analytics
- Graph Reading
  - Or how to turn graphs into text
- Building time-machines
  - Temporal text retrieval and analytics

Social (and not so social) Graphs

- Ubiquity of graphs
  - Social Networks
  - Biological networks
  - Citation networks
  - World Wide Web
  - …

Properties and Implications

- Structural
  - Power-law distribution of degrees
  - Low-diameter
  - Clearly non-planar

  ⇒ Algorithms designed for near-planar graphs (e.g., road networks) don't perform well
- Content
  - Heavily annotated and/or labeled

  ⇒ Combination of content-search + structural exploration is needed

Properties and Implications

- Size
  - Very large – millions of nodes, billion edges

  ⇒In-memory algorithms do not scale
- Frequently used
  - Navigation
  - Exploration

  ⇒Interactive speeds are necessary

  ⇒*Visual exploration of results*

  ⇒*Small errors may be tolerable*

Ferrari for fast reachability

Reachability Queries

- Given a graph G = (V, E)
- Consider two vertices, v and u in V
- Answer the question:
  - Is there a path starting from v, ending at u?
- Classical problem
  - Recursive reasoning (Prolog, Datalog)
  - SQL recursion operator
  - Graph processing
    - Biological graphs
    - Call Graphs
    - Social Networks
    - …

How do we answer this?

- Online
  - Explore the graph, starting from u – find a path to v
  - O (V+E) time, no extra space
  - Slow and boring
- Offline –
  - Essentially a transitive closure computation
  - $O(n^2)$ space, $O(1)$ time
  - Consider a million node graph
    - Worst-case we will have 8×10^{12} storage = 8 Terabytes
  - Facebook has (estimated) ~~750 million~~ 1 billion nodes!!
    - An Exabyte (10^{18})!

Interval Labeling

Agrawal et al. 1989

- Condense the graph by collapsing SCCs
- Build a spanning tree over the resulting DAG
  - Preorder traversal, and label the nodes
  - The label range at a node gives its reachable set using tree edges
- For the remaining edges
  - Propagate the labels upwards
  - Merge the labels to reduce the number of ranges at each node

✘ Computing an optimal cover is as hard as transitive closure computation

  ✓**Don't focus on "optimal cover"**

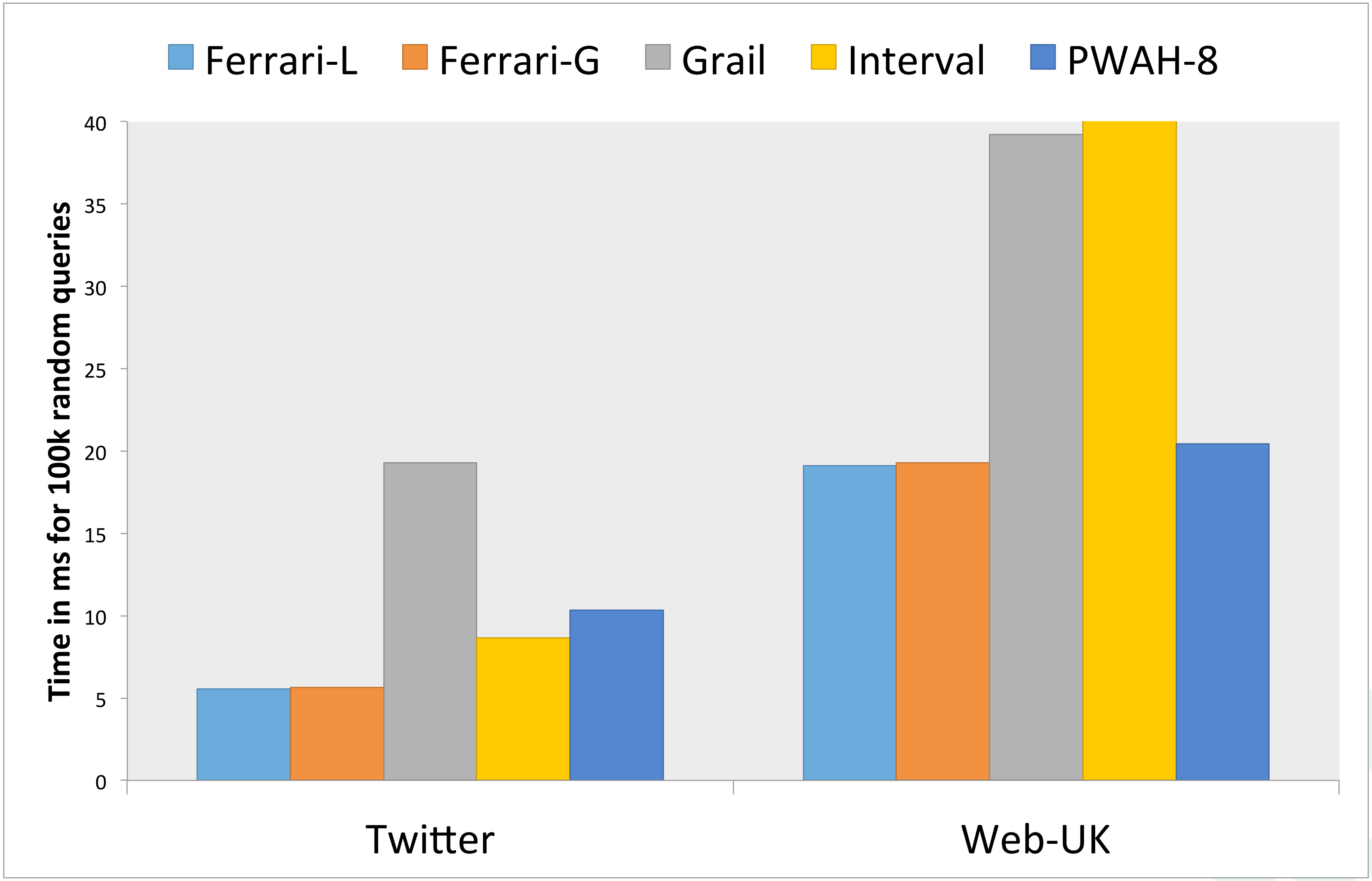
✘ The resulting index (i.e., interval set at each node) can be *fairly large*

  ✓**Explicit Space bound**

Ferrari : Flexible and Efficient Reachability Range Assignment for Indexing graphs

- Approximate the interval set at each node
  - Merge intervals that are "close by"
- You can answer negative reachability if target node id lies in the interval gap
- If they are within the approximated intervals
  - They may or may not be reachable (due to approximation)
  - So, recursively go down and **verify**
- Optimization goal:
  Minimize the number of such recursive queries

Experimental Results



Königsegg for Fast Navigation

Navigational Problems

Given a (un)directed graph G,

Reaching from node A to B

- Shortest path distance
- Shortest path
- Distances/paths that satisfy a constraint
- All paths in-between
- Ranked list of paths

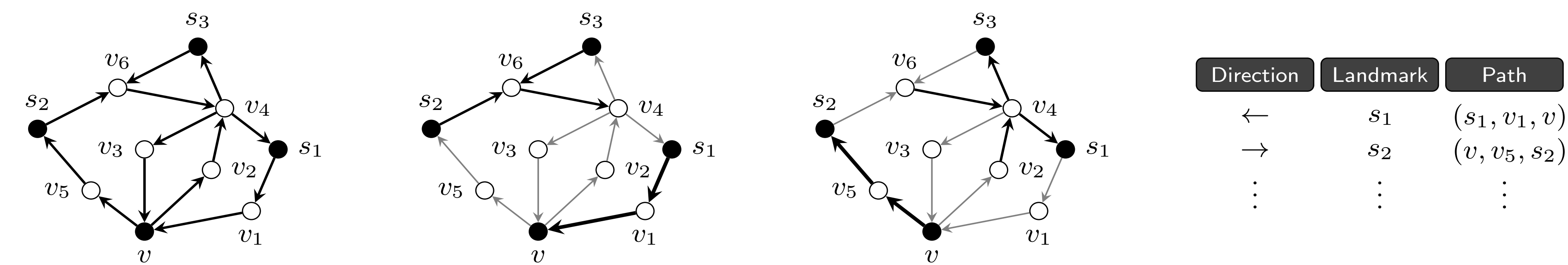
- Variants of Single-source Shortest-path (SSSP) problem

Popular Solutions

- Online computation
  Dijkstra's algorithm, Bellman-Ford, Bidirectional,A*-,D*-Searches, …
  - Typically require the graph to be in memory
  - Consume *huge* amount of intermediate memory
  - Are extremely good on near-planar graphs

- Offline indexing (+ approximation)
  Distance oracles [Thorup 2001] , Transit-node routing [Bast et al. 2007], Sketches [Das Sarma et al. 2010] ,…
  - Generate approximate results
  - Only for estimating **shortest path distances**
  - **Constant factor multiplicative errors**

A Brief Sketch of Graph Sketches
Das Sarma et al., WSDM 2010



(a) Vertex v and seed set $S = \{s_1, s_2, s_3\}$ **(b)** BFS from S in forward direction reaches v **(c)** BFS from S in backward direction reaches v **(d)** Resulting entries in Sketch(v)

Query: *dist (u,v) = ?*

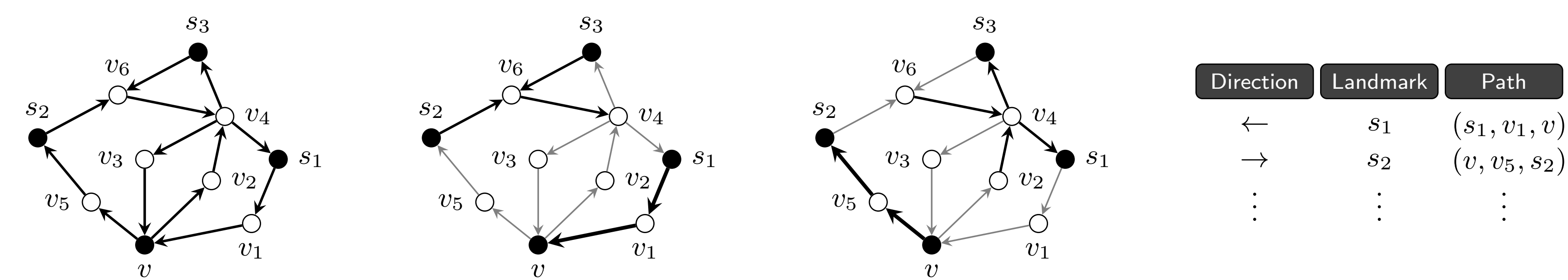
- Load *sketch(u)* and *sketch(v)*
- Look for common landmarks s.t.
  - *dist (u, s) ∈ sketch (u)*
  - *dist (s, v) ∈ sketch (v)*
- Answer the distance query by *dist(u, s) + dist(s, v) ≈ dist (u,v)*
- Has been shown to be (2c-1)-approximate, where

$$1 \le r\ (= \log \lfloor |V| \rfloor) \le c$$

Enriching the Graph Sketches

Instead of storing just the distance, store the entire path in the sketch

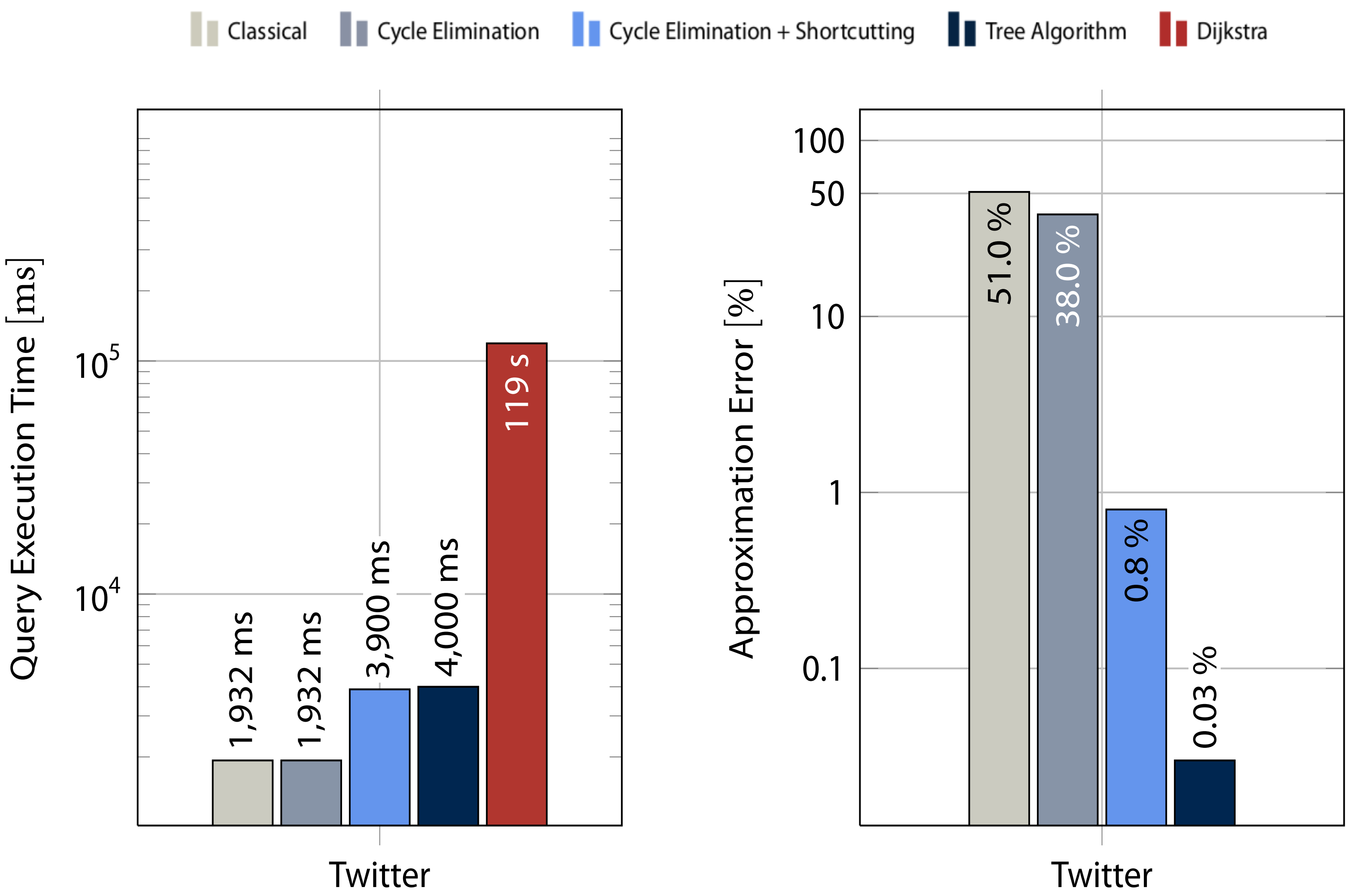
- No extra computation during construction
- Space overhead
    - A small constant-factor for networks with low diameter



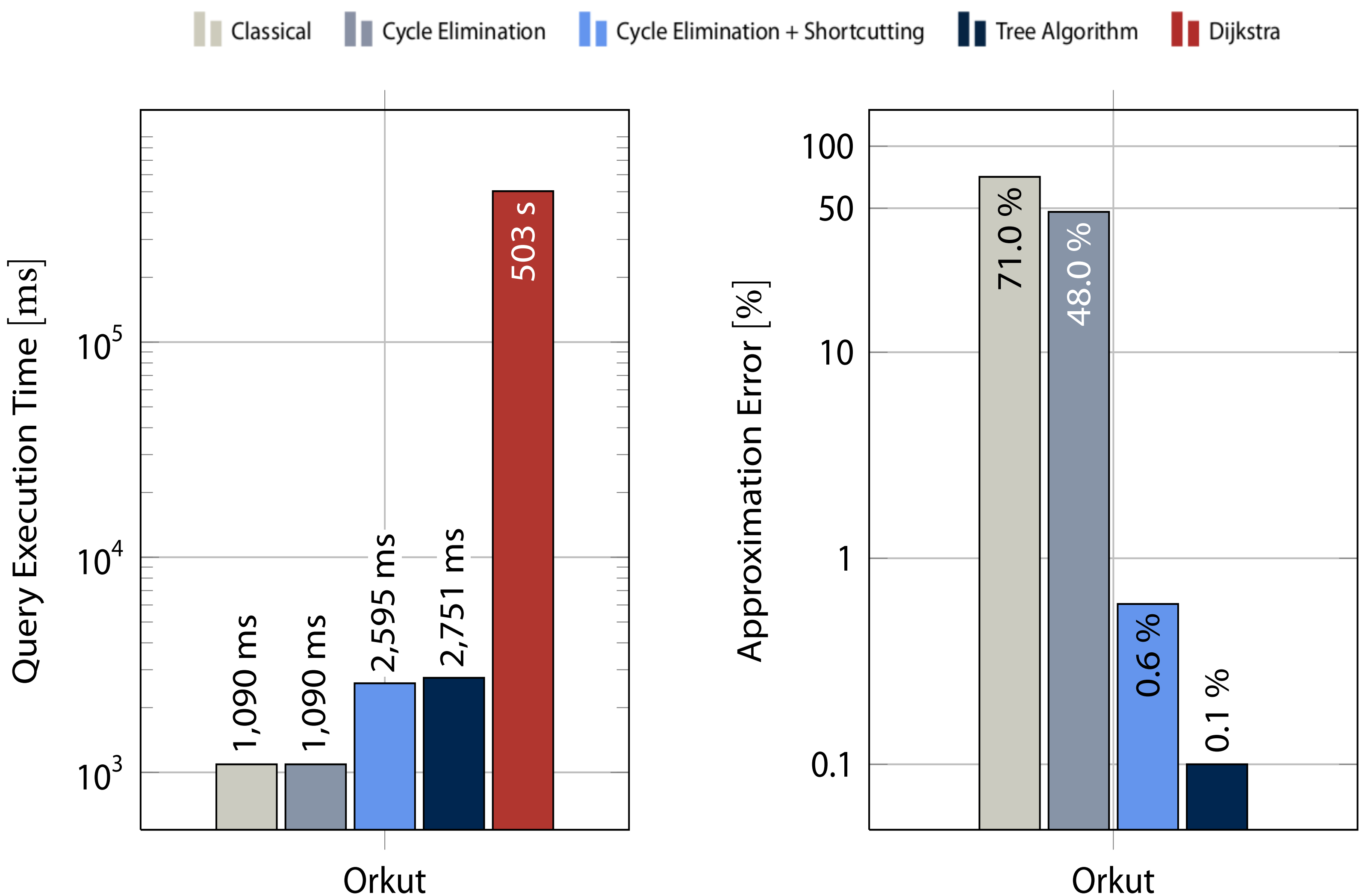
(a) Vertex v and seed set $S = \{s_1, s_2, s_3\}$ **(b)** BFS from S in forward direction reaches v **(c)** BFS from S in backward direction reaches v **(d)** Resulting entries in Sketch(v)

- Obvious advantage:
  Get the path corresponding to the estimated distance
- Anything else…?

Twitter


Classical
Cycle Elimination
Cycle Elimination + Shortcutting
Tree Algorithm
Dijkstra
Query Execution Time [ms]
10^5
10^4
1,932 ms
1,932 ms
3,900 ms
4,000 ms
119 s
Twitter
Approximation Error [%]
100
50
10
1
0.1
51.0 %
38.0 %
0.8 %
0.03 %
Twitter

Orkut



Classical
Cycle Elimination
Cycle Elimination + Shortcutting
Tree Algorithm
Dijkstra
Query Execution Time [ms]
10^5
10^4
10^3
1,090 ms
1,090 ms
2,595 ms
2,751 ms
503 s
Orkut
Approximation Error [%]
100
50
10
1
0.1
71.0 %
48.0 %
0.6 %
0.1 %
Orkut

Graphs with Social Leanings

Social Media for Trip Planning

Antourage

- Mining Distance-constrained Trips from Flickr
  - Shared photos on Flickr or Panoramio reflect interestingness of locations photographed
    - More than 90 million photos on Flickr are geotagged
- Question: Can we automatically provide tourist trails based on the social media?

London 10 km tour

Rome 10 km tour

- ***Alternate sources***
- ***Indian context***
  - ***Multiple languages***
  - ***Mixed languages***

Outlook

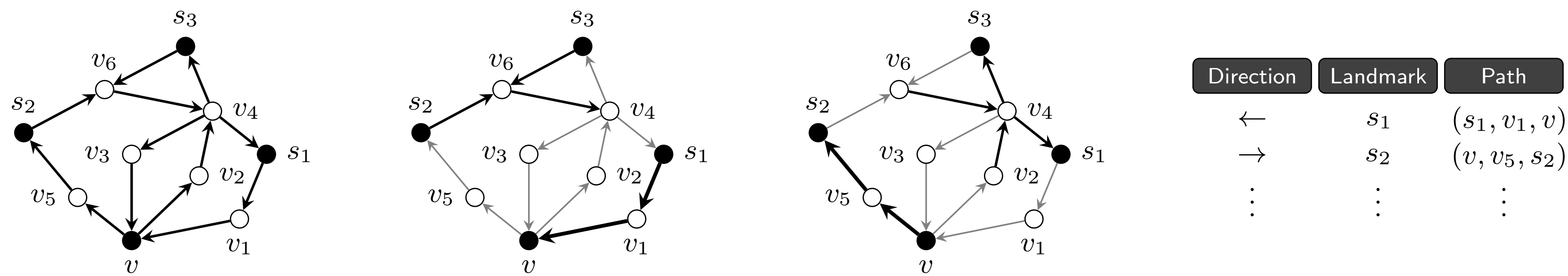
Current work

- *Scalable Graph Processing*
  - *Basic graph algorithmic blocks*
  - *Developing an interactive graph toolkit for large graphs*
- *Thematic tourist guidance systems*
  - *Driven entirely by social media*
  - *"Stuck" at extraction from noisy text*
- *Bringing graphs to masses*
  - *Query on graphs, turn results to text summaries*

Questions?

Sketching the Graph Sketches

Das Sarma et al., WSDM 2010



(a) Vertex v and seed set $S = \{s_1, s_2, s_3\}$

(b) BFS from S in forward direction reaches v

(c) BFS from S in backward direction reaches v

(d) Resulting entries in Sketch(v)

Let G=(V,E) be a directed graph

1. Set r = log ⌊|V|⌋
2. Sample r+1 sets of nodes (uniformly at random) of sizes: *1, 2, 2^2, 2^3, … , 2^r*
3. For every u ∈V and for every set S
    1. Find the closest nodes to u in S (landmarks)
       landmark h1 ∈ S: dist (u, h1) = dist (u, S)
       landmark h2 ∈ S: dist (h2, u) = dist (S, u)
    2. Store these in sketches:
       Sketch(u) = <u> <distance> <h1>; <h2> <distance> <u>
4. Repeat steps 2-3 k-times

Answering Queries using Sketches

Query: *dist (u,v) = ?*

- Load *sketch(u)* and *sketch(v)*
- Look for common landmarks s.t.
  - $dist\ (u, s) \in sketch\ (u)$
  - $dist\ (s, v) \in sketch\ (v)$
- Answer the distance query by $dist(u, s) + dist(s, v) \approx dist\ (u,v)$
- Has been shown to be (2c-1)-approximate, where

$$1 \leq r \leq c$$

Limitations

- Each round of sampling needs to traverse the whole graph repeatedly
  - k=10 is suggested!
- Is only an estimator of the distance
  - Obtaining the actual path requires interleaved access to the graph and its sketch
- Empirically observed errors are high for social networks
  - Factor 2 – 3 error is not tolerable

Navigation

- Dealing with dynamics
  - How to efficiently update sketches when graph is updated?
- Dealing with constraints over edges/nodes
  - State-of-the-art resorts to online computation
  - Current index solutions do not scale [Jin et al. 2010]
- Guarantees on the result quality
- Integration with distributed graph databases (like Pregel, Trinity, Neo4J, etc.)

Exploration

- Usability study of graph mining algorithms
  - How well they support visual exploration?
  - Develop a real-world benchmark
- Inclusion of additional "interestingness" measures
  - Edge-weights
  - Density [Sozio & Gionis, 2010]
- Explore meta-heuristics for size-constrained problems [Jain, Seufert and Bedathur, 2010]

- Current work:
  - Adding constraints
  - Make it more disk-friendly
  - Explore its use in distributed graph reasoning