ACM SIGMOD Programming Contest 2016

Engineering Distance Queries on Highly Evolving Networks

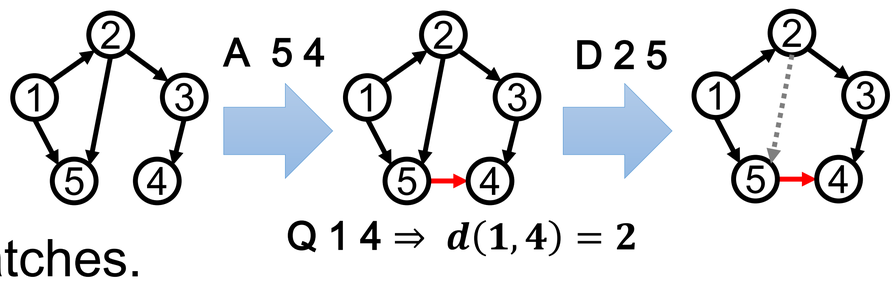
Team *H_minor_free*

Member: Takuto Ikuta, Takanori Hayashi, Yosuke Yano, Yoichi Iwata (The University of Tokyo)
Advisor: Takuya Akiba (National Institute of Informatics)

Task Overview

Given a “real” graph, process three types of queries.

- Edge addition (A)
- Edge deletion (D)
- Distance query (Q)



Queries are given as batches.

- 1,000 queries / batch. (# of batches ≈ 100)
- Process a current batch before processing a next batch.

Goal: Minimize the total time for processing all batches.

Environment

Intel E5-2620v2 (2.10Ghz (2.60Ghz turbo)),

- 20GB main memory
- 12 Cores / 24 Hyperthread

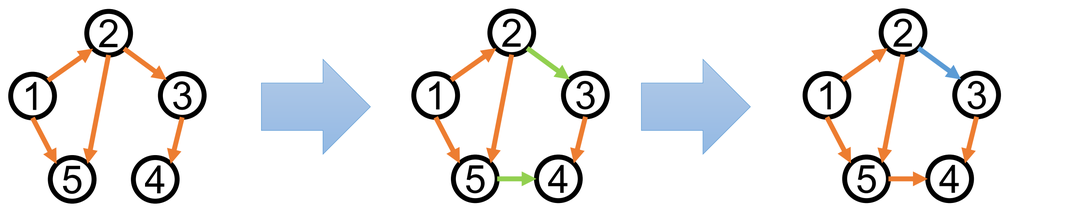
Difficulties

- Many threads
- Frequent updates **(A : D : Q = 1 : 1 : 3)**
  - Existing indexing methods cannot handle such frequent update efficiently.
- Different characteristics of real networks
  - Social? Web? Citation? Road?

Our Solution

Query
A 5 4
Q 1 4
Q 1 3
D 2 3

edge tag type
- ALIVE
- DEAD
- UNKNOWN



- In our algorithm, each edge has one of 3 type tag
  1. ALIVE/DEAD: the edge is passable/impassable
  2. UNKNOWN: edges modified in the batch, checked by binary search for queries in BFS
- For each batch, our algorithm takes 3-step process.
  1. modifying queries add UNKNOWN to the edge.
  2. bidirectional BFS processes distance query using one of 24 OpenMP threads.
  3. update edge tag to ALIVE or DEAD after bidirectional BFS.
- We select an appropriate algorithm depend on graph parameter (e.g. average distance).

Bidirectional Search Heuristics

Existing method [Goldberg, Harrelson. ’05] [Jin+.’13]

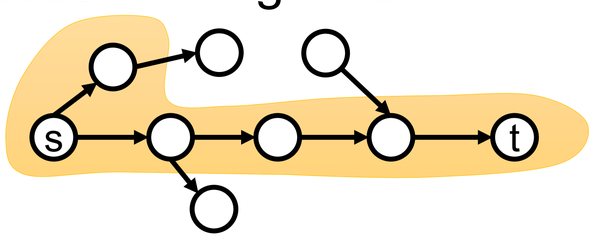
- For each iteration, alternately choose a vertex from frontiers of forward and backward searches and traverse edges from that vertices

Our method

- For each iteration, choose a search direction that visits less vertices and traverse edges from all vertices in the frontier of that direction.
- Finish the search right after we find a shortest path.

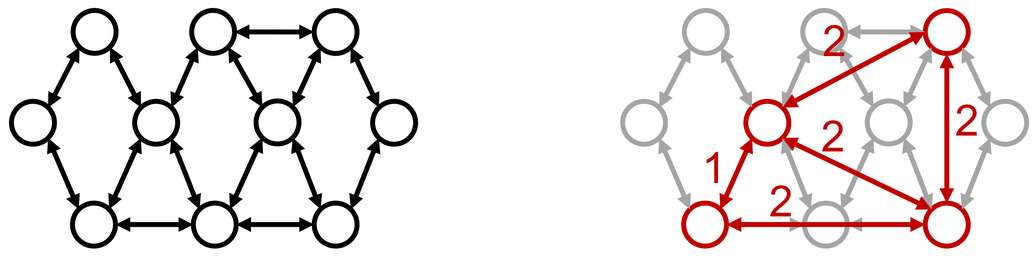
Pruning for Zero Out-degree Vertices

For citation network and social network, we introduce pruning for zero out-degree vertices.
In bidirectional BFS, there is no need to visit zero out-degree vertices with distance of 2 or more from start and target vertices.
If distance query is (s, t), our algorithm visits vertices in colored area only.
In X-Large test, about half of vertices (1.67 million / 3.77 million) have zero out degree. This pruning extensively reduces running time.



Overlay graph

Speedup searches between distant vertices



Original graph G — Overlay graph G'

Construct a overlay graph $G' = (V', E', w')$

- $V' \subseteq V, E' \subseteq V' \times V'$
- For each $(u, v) \in E'$, $w'(u, v) = d_G(u, v)$
- For each dynamic update, we update w' by carrying out a local search from that edge.
- We set **k-all-path cover** [Funke+. VLDB’14] to V'

Every simple path with length K share a common vertex with V'

- We obtain 4096-all-path cover by using heuristics
- Good trade-off between querying and updating time

Query processing: Bidirectional Dijkstra’s method.

- From $v \in V - V'$, traverse edges in G
- From $v \in V'$, traverse edges in G'

Speedup of query processing

RoadNet-CA
http://snap.stanford.edu/data/roadNet-CA.html

Original	**Overlay**
$V = 2.0\text{M}, E = 5.5M$	$V = 205K, E = 1{,}4M$
5.2 ms / per query	1.3 ms / per query