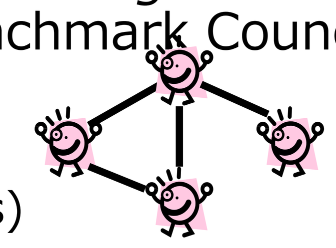
SIGMOD Programming Contest 2014
Efficient Implementation of Social Network Analysis System

team: H_minor_free
T. Akiba, T. Hayashi, S. Hirahara, T. Ikuta, Y. Iwata, Y. Kawata,
N. Ohsaka, K. Oka, Y. Yano (The University of Tokyo, Japan)
R. Okuta (Tohoku University, Japan)

Construct a social network analysis system

Synthetic social networks

Generated by
social network benchmark dataset generator
produced by Linked Data Benchmark Council



- Persons — Vertices of the graph
  - Posts (Comments and Replies)
  - Interests
- Friendship between persons — Edges of the graph

$\#persons \sim 1{,}000{,}000$

Four types of queries

- Query1: Shortest distance over frequent communication paths
- Query2: Interests with Large Communities
- Query3: Socialization Suggestion
- Query4: Most Central People

Contest goal
Execute all the queries as quick as possible

Evaluation Environment

- Processor 2.67 GHz Intel Xeon E5430
- Configuration 2 processors (8 cores total)
- L2 Cache Size 12MB
- Main Memory 15 GB

Overview of our system

- Implement one program for each query type.
- Run programs simultaneously.
- Parallel computing using OpenMP
  Query1 (1-2 threads), Query2(1 thread)
  Query3 (1 thread) Query4(6-8 threads)

Query1: Shortest distance over frequent communication paths

Query1(p_1, p_2, x)
- Answer the shortest path distance between persons p_1 and p_2.
- Only use edges whose communication frequency is more than x.

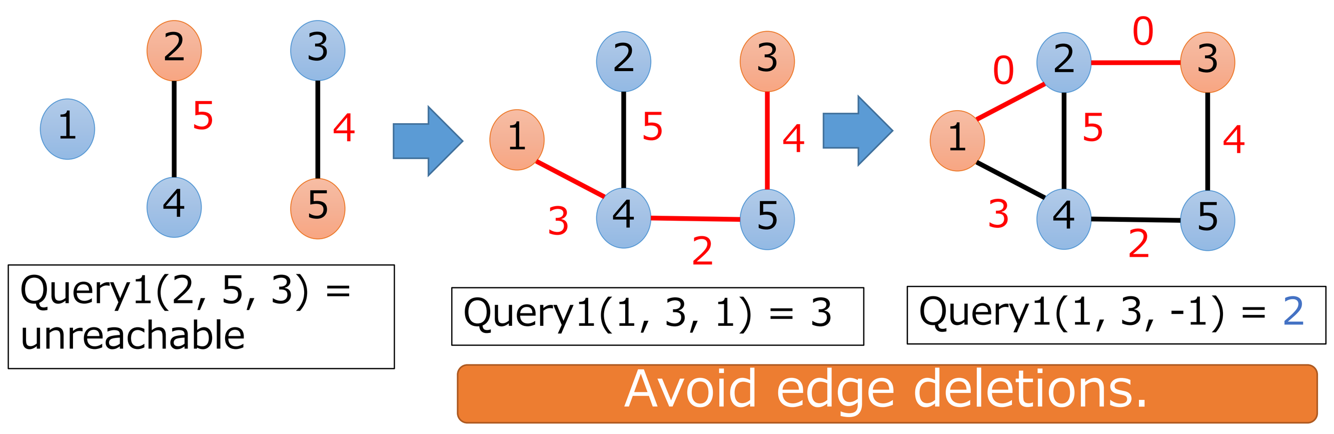
Process the queries in the degreasing order of x.

1. Query1(1, 3, 1)
2. Query1(2, 5, 3)
3. Query1(1, 3, -1)

→

1. Query1(2, 5, 3)
2. Query1(1, 3, 1)
3. Query1(1, 3, -1)

Before processing Query1(p_1, p_2, x), add passable edges.



Calculate communication frequency

The most time-consuming part is to compute the communication frequency $F(p_1, p_2)$ for all edges.

$$F(p_1,p_2) \coloneqq \min\,(\#(p_1\text{'s replies to } p_2\text{'s comments}),\ \#(p_2\text{'s replies to } p_1\text{'s comments}))$$

To compute $F(p_1, p_2)$, we must read two large files.
- comment_hasCreator_person.csv.
  Pairs of a comment and a person who makes this comment.
- comment_replyOf_comment.csv.
  Pairs of a reply and a comment that receives this replycomment_replyOf_comment.csv

Linear time graph construction without sorting.
- Use the technique similar to bucket sort.
- Use the hash to get the person who makes the comment.
- Use SIMD instructions to achieve fast file reading.

Query3: Socialization Suggestion

Query3(k, h, p):
- Answer the top-k similar pairs of persons based on the number of common interest tags.
- For each of k pairs,
  - The two persons must be located in p.
  - The shortest distance between two persons is not more than h.

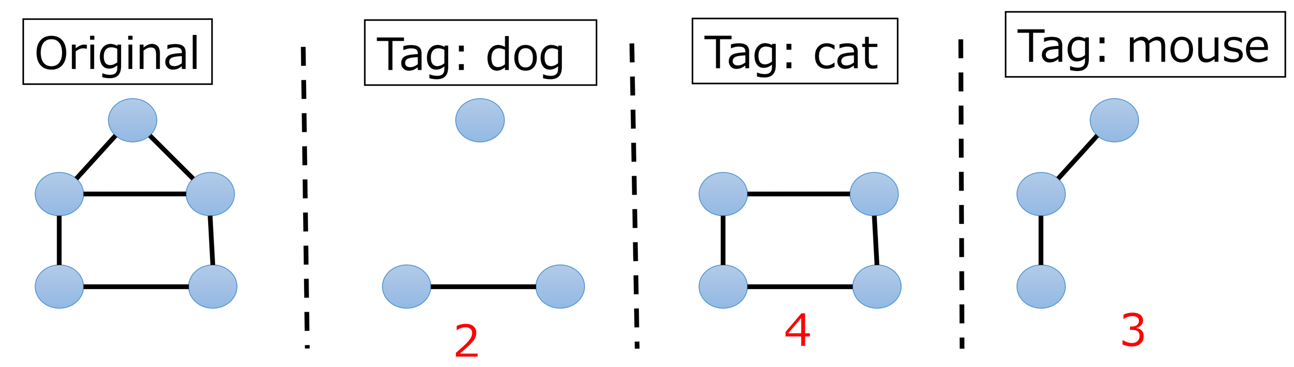
Our solution

1. Compute the pairs with high similarity values
   - Guarantee two persons are located in p
   - Do not check the distance between them.
2. Check the distance between the pairs in the decreasing order of similarity values
   - Output first k pairs that the distance between two persons is not larger than h

Query2: Interests with Large Communities

Query2(k, d):
Answer the k interests tags with the largest range.

The range of an interest tag t is the largest connected component in the graph induced by persons who
- Have the interest tag t
- Were born on d or later



Sort the queries in the decreasing order of d.

For each query $Query2(k, d)$,
- Add persons whose birthday are d or later.
- Maintain the connected components for each tag t by using an Union-Find tree.

Query4: Most Central People

Query4'(k, t):
Answer k persons with the smallest sum of shortest distances in the induced subgraph depending on t.

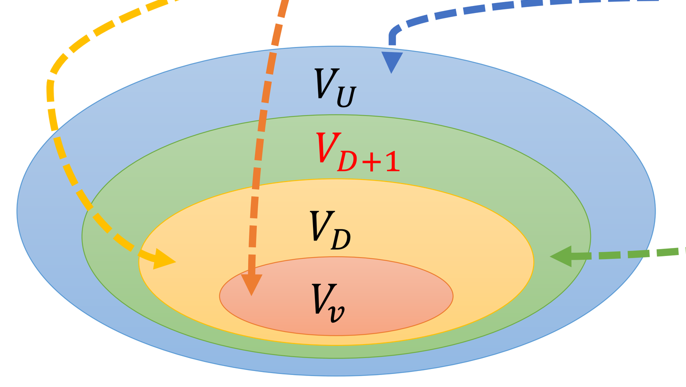
Prune the BFS by using current top-k values.
- Conduct BFSs in the decreasing order of degree in the graph.
  For a person p, $S(p)$ and $\deg(p)$ are strongly related.
- Estimate lowerbound of $S(p)$ during the BFSs
  If the lowerbound of $S(p)$ is large, the BFS is pruned.

How to estimate good lowerbound?

$$L(p) \coloneqq \sum_{q \in V_v} d(p,q) + D \times |V_D| + (D+1) \times |V_{D+1}| + (D+2) \times (|V_U| - |V_{D+1}|)$$

Estimate the number of persons whose distance from p is $D+1$.

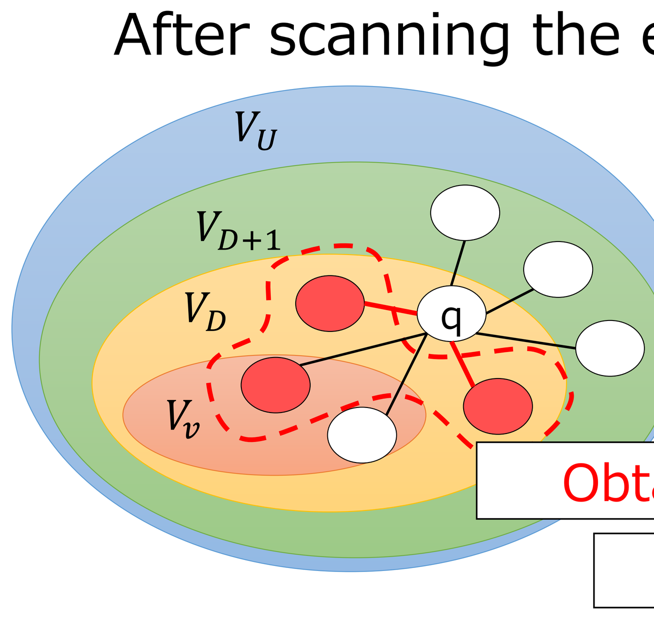
$$|V_{D+1}| \coloneqq \sum_{q \in V_D} (\deg(q) - 1)$$



- V_v ≔ already visited persons
- V_D ≔ persons in the queue
- V_U ≔ unreached persons
- D ≔ distance to vertices in V_D

Lowerbound update

After scanning the edges from $q \in V_Q$,



○ are newly reached vertices or the parent vertex in shortest path tree.

$|V_{D+1}|$ is overestimated by #●

$L(p) \leftarrow L(p) + \#$●

Obtain tighter lowerbound!

$= \deg(q) - 1 - \#(newly\ visited\ vertices)$