Distributed Set Reachability

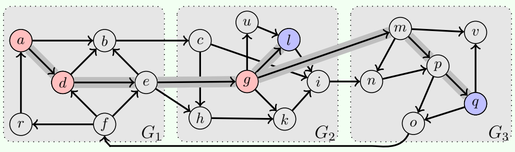
Sairam Gurajada[†] and Martin Theobald[‡]

[†]Max-Planck Institut für Informatik, [‡]Universität Ulm [†]gurajada@mpi-inf.mpg.de, [‡]martin.theobald@uni-ulm.de

Distributed Set Reachability

Definition. Given a directed graph $G(V, E)$, a k vertex-disjoint partitioning of G as $\mathcal{G} = \{G_1, G_2, \dots, G_k\}$, a source set $S \subseteq V$, and a target set $T \subseteq V$, a DSR query $S \rightsquigarrow T$ returns all reachable pairs, i.e.,

$$S \rightsquigarrow T = \{(s,t) | s \rightsquigarrow t \text{ where } s \in S \text{ and } t \in T\}$$



Partitioned Graph $\mathcal{G} = \{G_1, G_2, G_3\}$

Example $S = \{a, d, g\}$ and $T = \{l, q\}$,

$$S \rightsquigarrow T = \{(a,l),(a,q),(d,l),(d,q),(g,l),(g,q)\}$$

Related work. Distributed reachability (Fan et al. [1]), centralized multi-source multi-target reachability (Gao et al. [2], Then et al. [3]).

Applications

1. Property paths processing in SPARQL 1.1
2. Community connectedness in social networks

Solving DSR Queries

Vertex-centric approach.

- For each $s \in S$, perform BFS traversal
- Each $v \in V$ maintains a list of sources that reach v
  Challenges:
  1. iterative approach (no. of iterations $\leq$ diameter)
  2. no support for indexes to speedup processing
  3. no reuse of computations
  4. hard to interleave with other operators, e.g., joins in SPARQL

Our Objective.

- To reuse computations as much as possible at each local partition
- A non-iterative approach, to reduce number and size of messages exchanged
- Flexibility to use any existing centralized indexing techniques to speed up query processing

Our Approach: Indexing

Definitions.

Given a partitioning $\mathcal{G} = \{G_1, G_2, \dots, G_k\}$

- $G_i(V_i, E_i)$ is a partition of G and $C(V_C, E_C)$ denotes cut for a given $\mathcal{G}$
- $In-boundaries(I_i)$: set of all vertices in G_i which have incoming edges from vertices in other partitions
- $Out-boundaries(O_i)$: set of all vertices in G_i which have outgoing edges to vertices in other partitions

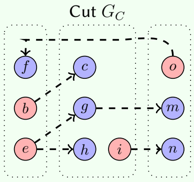
Example:

$$I_1 = \{f\}, I_2 = \{c, g, h\}, I_3 = \{m, n\}$$
$$O_1 = \{b, e\}, I_2 = \{g, i\}, I_3 = \{o\}$$

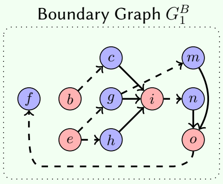
1. Boundary graph.

- Build a per-partition boundary graph $G_i^B(V_i^B, E_i^B)$
- $V_i^B = V_C$; $E_i^B = E_C \cup \{(u,v) | u, v \in V_j \text{ and } u \rightsquigarrow v \; \forall j \neq i\}$

Cut G_C



Boundary Graph G_1^B



2. Boundary graph compression via equivalence sets.

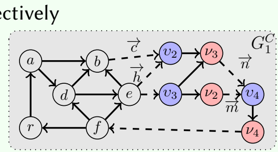
- Large boundary graph – $\mathcal{O}(\sum_{i=1}^{k}(|I_i| \cdot |O_i|) + |E_C|)$
- *Forward Equivalence (FE):* Two in-boundaries b_1, b_2 are forward-equivalent, i.e., $b_1 \equiv^f b_2$ iff for any vertex $v \in V_i - I_i$, and $b_1 \rightsquigarrow v$, it holds that $b_2 \rightsquigarrow v$
- *Backward Equivalence (BE):* Two out-boundaries b_1, b_2 are backward-equivalent, i.e., $b_1 \equiv^b b_2$ iff for any vertex $v \in V_i - O_i$, and $v \rightsquigarrow b_1$, it holds that $v \rightsquigarrow b_2$

Example:

- FE sets(*in-virtual nodes*): $\upsilon_1 = \{f\}, \upsilon_2 = \{c, h\}, \upsilon_3 = \{g\}, \upsilon_4 = \{m, n\}$
- BE sets(*out-virtual nodes*): $\nu_1 = \{b, e\}, \nu_2 = \{i\}, \nu_3 = \{g\}, \nu_4 = \{o\}$

3. Compound Graph.

- Build $G_i^C(V_i^C, E_i^C)$ by merging local graph G_i and the corresponding boundary graph G_i^B
- (Optional) Build index over G_i^C to speed up local query processing
- define forward list F_i and B_i, set of non-local in-virtual and out-virtual nodes respectively



Forward & backward lists:

$$F_1 = \{\upsilon_2, \upsilon_3, \upsilon_4\}$$
$$B_1 = \{\nu_2, \nu_3, \nu_4\}$$

Our Approach: Querying

Query: $S \rightsquigarrow T$

Step 0. Partition S, T into $\{S_1 \rightsquigarrow T_1, \dots, S_k \rightsquigarrow T_k\}$

For e.g.,

$$\{a,d,g\} \rightsquigarrow \{l,q\} : \quad \overset{G_1}{\{a,d\} \rightsquigarrow \emptyset} \quad \overset{G_2}{\{g\} \rightsquigarrow \{l\}} \quad \overset{G_3}{\emptyset \rightsquigarrow \{p\}}$$

Step 1. Compute local reachability $S_i \rightsquigarrow T_i$ and $S_i \rightsquigarrow F_j, \forall j \neq i$

Example:

	G_1	G_2	G_3
$S_i \rightsquigarrow T_i :$	$\{a,d\} \rightsquigarrow \emptyset$	$\{g\} \rightsquigarrow \{l\}$	$\emptyset \rightsquigarrow \{p\}$
$S_i \rightsquigarrow F_i :$	$\{a,d\} \rightsquigarrow \{c,g,h,m,n\}$	$\{g\} \rightsquigarrow \{f,m,n\}$	$\emptyset \rightsquigarrow \{f,c,g,h\}$

Step 2. Communicate reachability $S_i \rightsquigarrow F_i$ to other partitions

Example:

$1 \rightarrow 2 : \{(c,[a,d]),(g,[a,d]),(h,[a,d])\}$ $\quad 2 \rightarrow 1 : \{(f,[g])\}$ $\quad 3 \rightarrow 1 : \emptyset$

$1 \rightarrow 3 : \{(m,[a,d]),(n,[a,d])\}$ $\quad 2 \rightarrow 3 : \{(m,[g]),(n,[g])\}$ $\quad 3 \rightarrow 2 : \emptyset$

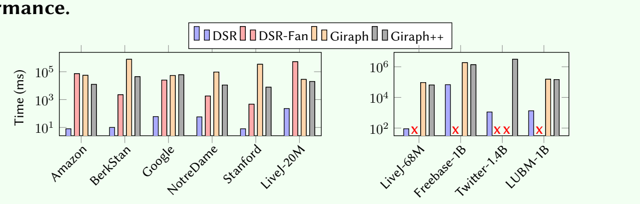
Step 3. Compute local reachability from boundaries to targets

Example:

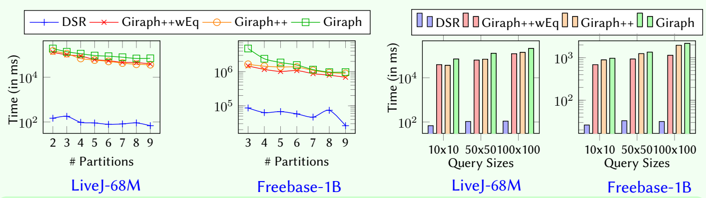
	G_1	G_2	G_3
	$\{f\} \rightsquigarrow \emptyset$	$\{c,g,h\} \rightsquigarrow \{l\}$	$\{m,n\} \rightsquigarrow \{p\}$
Result:	$\emptyset$	$\{\langle a,l\rangle, \langle d,l\rangle, \langle g,l\rangle\}$	$\{\langle a,p\rangle, \langle d,p\rangle, \langle g,p\rangle\}$

Evaluation

Performance.



Scalability.



References

[1] Fan, W. et al. Performance Guarantees for Distributed Reachability Queries. VLDB 2012.

[2] Gao, S. et al. PrefixSolve: efficiently solving multi-source multi-destination path queries on RDF graphs by sharing suffix computations. WWW 2013.

[3] Then, M. et al. The More the Merrier: Efficient Multi-Source Graph Traversal. VLDB 2014

ulm university universität uulm

max planck institut informatik