Exploring Shortest Paths on Large-scale Networks

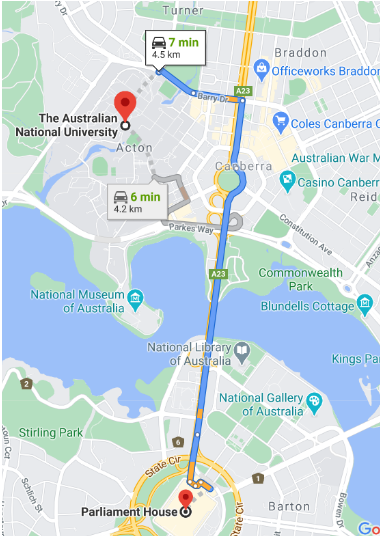
Ye Wang

School of Computing,
Australian National University

Introduction

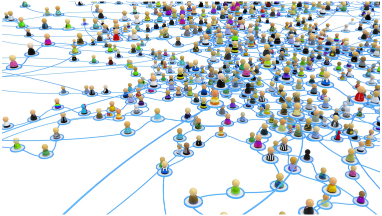
Shortest paths:

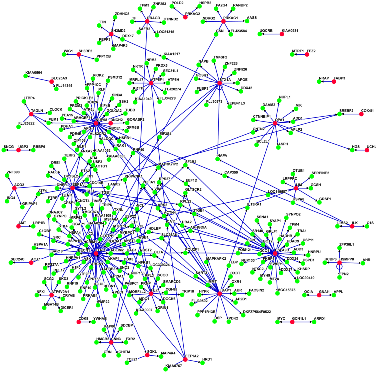
- a fundamental notion for graph analytics.



Shortest paths:

- a basis for tackling various related problems, e.g.,
  - social network analysis
  - find essential proteins in protein-protein interaction networks





Research Questions

1. How do two vertices in a graph connect with each other?
    - Point-to-point shortest path problem
    - ...
    - Shortest path graph problem [1]

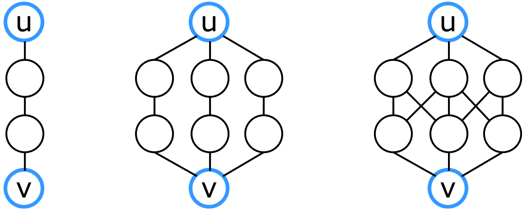
2. How does one vertex connect with other vertices?
    - Single source shortest path problem
    - ...
    - Top-k relative coverage problem

[1] Ye Wang, Qing Wang, Henning Koehler, and Yu Lin. 2021. Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks. In Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21).

Shortest Path Graph Problem

Given a graph G and two vertices u, v,

- Vertices with the same distance may be connected by different shortest paths.

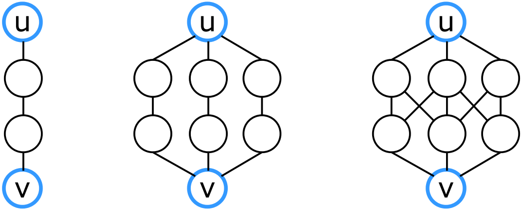


(a) (b) (c)

Shortest Path Graph Problem

Given a graph G and two vertices u, v,

- Vertices with the same distance may be connected by different shortest paths.
- Enumerating all shortest paths between vertices is combinatorially challenging.

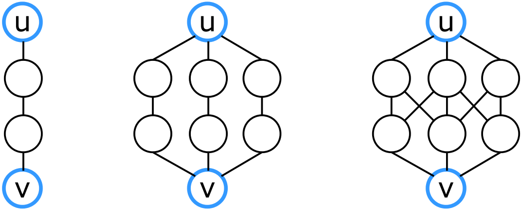


(a) (b) (c)

Shortest Path Graph Problem

Given a graph G and two vertices u, v,

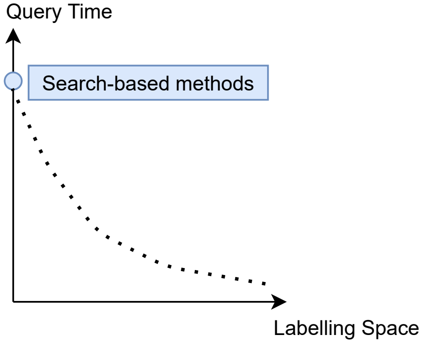
- Vertices with the same distance may be connected by different shortest paths.
- Enumerating all shortest paths between vertices is combinatorially challenging.
- Thus, we find shortest path graph G_{uv} - a subgraph containing *exactly all shortest paths* between vertices.



(a) (b) (c)

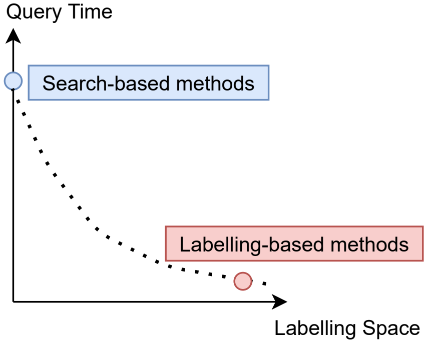
Shortest Path Graph - Related Work

- Search-based methods
  - Dijkstra: $O(|E|log|V|)$ query time
  - Breadth-first search: $O(|E|)$ query time



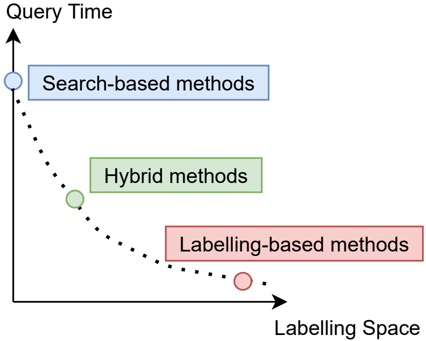
Shortest Path Graph - Related Work

- Search-based methods
  - Dijkstra: $O(|E|log|V|)$ query time
  - Breadth-first search: $O(|E|)$ query time
- Labelling-based methods
  - Pruned path labelling: $O(|V|^2)$ labelling space
  - Parent pruned path labelling: $O(|V||E|)$ labelling space



Shortest Path Graph - Related Work

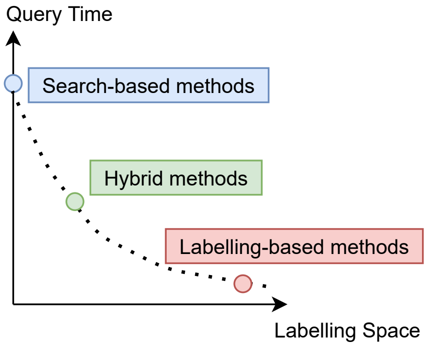
- Search-based methods
  - Dijkstra: $O(|E|log|V|)$ query time
  - Breadth-first search: $O(|E|)$ query time
- Labelling-based methods
  - Pruned path labelling: $O(|V|^2)$ labelling space
  - Parent pruned path labelling: $O(|V||E|)$ labelling space
- Hybrid methods?
  - A trade-off



Shortest Path Graph - Research Goal

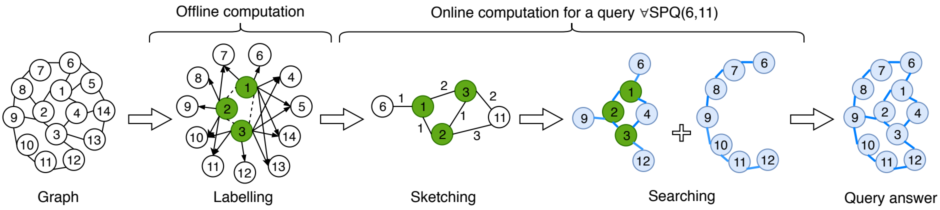
Design a hybrid method by combining search and labelling to achieve

- **Query time efficiency:** less than 0.5 seconds
- **Labelling space efficiency:** comparable size with the original graph
- **Scalability:** networks with billions of vertices and edges



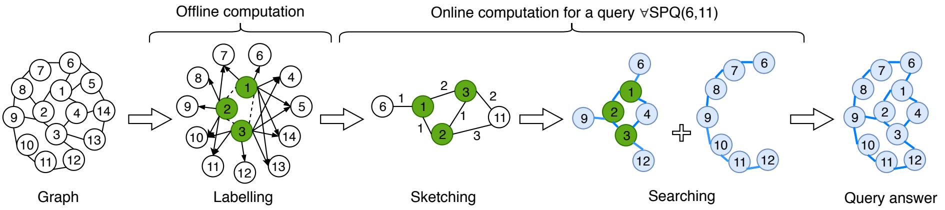
Shortest Path Graph - Proposed Method

- Propose a novel method, called *Query-by-Sketch*



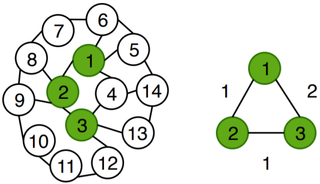
Shortest Path Graph - Proposed Method

- Propose a novel method, called *Query-by-Sketch*
- Three key ideas:
  (1) Labelling scheme;
  (2) Fast sketching;
  (3) Guided searching.



Shortest Path Graph - Proposed Method

- Let $G = (V, E)$ and R be a set of landmarks, and $|R| \ll |V|$. A labelling scheme $\mathcal{L} = (M, L)$ consists of
  - M: a meta-graph over R
  - L: a path labelling over G



Graph

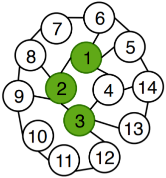
Meta-graph

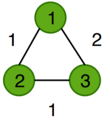
Shortest Path Graph - Proposed Method

- Let $G = (V, E)$ and R be a set of landmarks, and $|R| \ll |V|$. A labelling scheme $\mathcal{L} = (M, L)$ consists of
  - M: a meta-graph over R
  - L: a path labelling over G

(1,1): d(4,1)=1

(3,1): d(4,3)=1



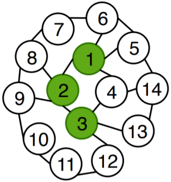


Label	Labelling Entries
$L(4)$	(1,1) (3,1)
$L(5)$	(1,1) (3,3)
$L(6)$	(1,1)
$L(7)$	(1,2) (2,2)
$L(8)$	(2,1)
$L(9)$	(2,1)
$L(10)$	(2,2) (3,3)
$L(11)$	(2,3) (3,2)
$L(12)$	(3,1)
$L(13)$	(1,3) (3,1)
$L(14)$	(1,2) (3,2)

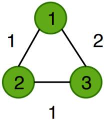
Graph Meta-Graph Path Labelling

Shortest Path Graph - Proposed Method

- Let $G = (V, E)$ and R be a set of landmarks, and $|R| \ll |V|$. A labelling scheme $\mathcal{L} = (M, L)$ consists of
  - M: a meta-graph over R
  - L: a path labelling over G



Graph

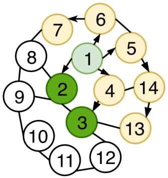


Meta-graph

Label	Labelling Entries
L(4)	(1,1) (3,1)
L(5)	(1,1) (3,3)
L(6)	(1,1)
L(7)	(1,2) (2,2)
L(8)	(2,1)
L(9)	(2,1)
L(10)	(2,2) (3,3)
L(11)	(2,3) (3,2)
L(12)	(3,1)
L(13)	(1,3) (3,1)
L(14)	(1,2) (3,2)

Path-Labelling

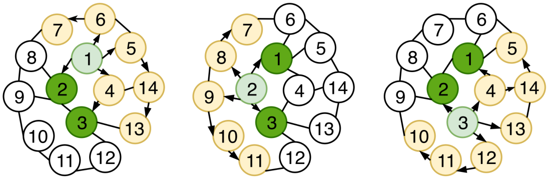
- Let $G = (V, E)$ and R be a set of landmarks, and $|R| \ll |V|$. A labelling scheme $\mathcal{L} = (M, L)$ consists of
  - M: a meta-graph over R
  - L: a path labelling over G



(a)

Shortest Path Graph - Proposed Method

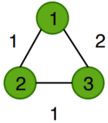
- Let $G = (V, E)$ and R be a set of landmarks, and $|R| \ll |V|$. A labelling scheme $\mathcal{L} = (M, L)$ consists of
  - M: a meta-graph over R
  - L: a path labelling over G



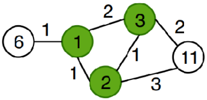
(a) (b) (c)

Shortest Path Graph - Proposed Method

- A sketch S_{uv} for u and v estimates how u and v are connected.

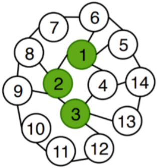


Label	Labelling Entries
L(4)	(1,1) (3,1)
L(5)	(1,1) (3,3)
L(6)	(1,1)
L(7)	(1,2) (2,2)
L(8)	(2,1)
L(9)	(2,1)
L(10)	(2,2) (3,3)
L(11)	(2,3) (3,2)
L(12)	(3,1)
L(13)	(1,3) (3,1)
L(14)	(1,2) (3,2)

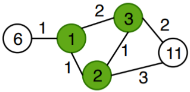


Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^- = [G \backslash R]$



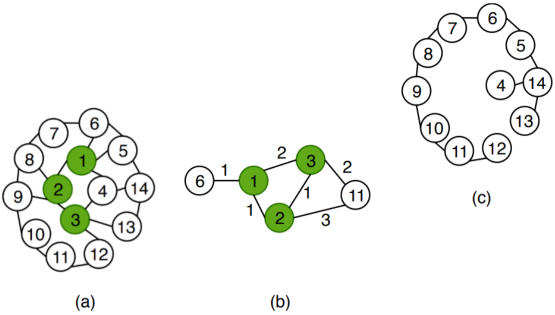
(a)



(b)

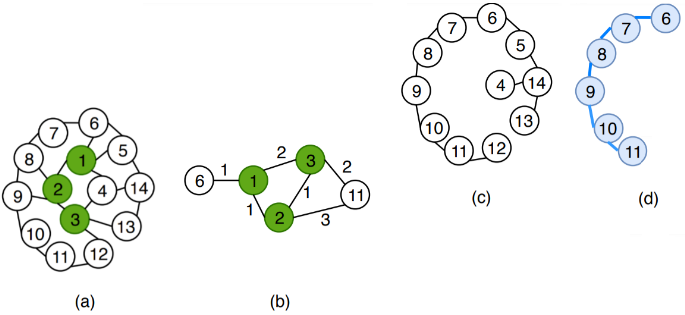
Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^{-} = [G \backslash R]$



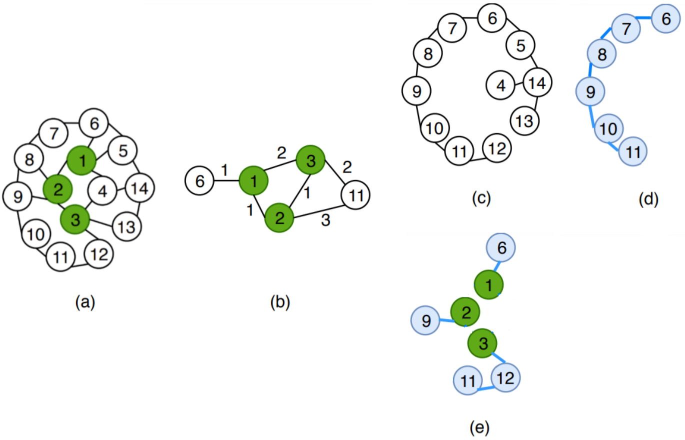
Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^- = [G \backslash R]$



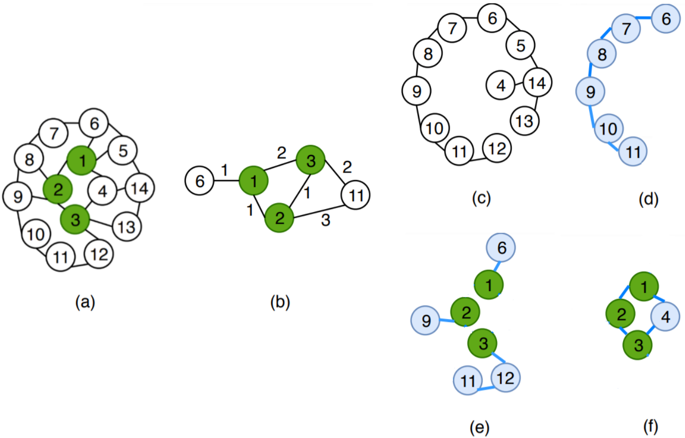
Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^{-} = [G \backslash R]$
- Searching shortest paths captured by the sketch



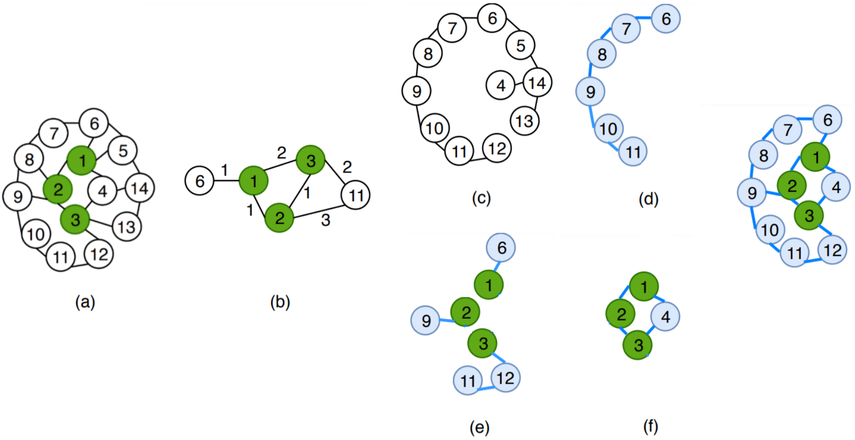
Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^{-} = [G \backslash R]$
- Searching shortest paths captured by the sketch



Shortest Path Graph - Proposed Method

- Searching shortest paths on the sparsified graph $G^{-} = [G \backslash R]$
- Searching shortest paths captured by the sketch



Shortest Path Graph - Experiments

We evaluate Query-by-sketch (QbS) against the baselines:

- Search-based methods:
  - Bi-directional BFS (Bi-BFS)
- Labelling-based methods:
  - Pruned path labelling (PPL)
  - Parent pruned path labelling (ParentPPL)

Shortest Path Graph - Experiments

- **Datasets:** 12 real-world complex networks

Dataset	$\lvert V \rvert$	$\lvert E^{un} \rvert$	max. deg	avg. deg	avg. dist
Douban (DO)	0.2M	0.3M	287	4.2	5.2
DBLP (DB)	0.3M	1.1M	343	6.6	6.8
Youtube (YT)	1.1M	3.0M	28,754	5.27	5.3
WikiTalk (WK)	2.4M	4.7M	100,029	3.89	3.9
Skitter (SK)	1.7M	11.1M	35,455	13.08	5.1
Baidu (BA)	2.1M	17.0M	97,848	15.89	4.1
LiveJournal (LJ)	4.8M	43.1M	20,334	17.79	5.5
Orkut (OR)	3.1M	117M	33,313	76.28	4.2
Twitter (TW)	41.7M	1.2B	2,997,487	57.74	3.6
Friendster (FR)	65.6M	1.8B	5,214	55.06	4.8
uk2007 (UK)	106M	3.3B	979,738	62.77	5.6
ClueWeb09 (CW)	1.7B	7.8B	6,444,720	9.27	7.5

Shortest Path Graph - Experiments

Q1: *How efficiently can QbS construct labelling?*

Dataset	Construction Time (sec.)		
	QbS	PPL	ParentPPL
Douban	0.3	154	2,736
DBLP	1.1	2,610	11,049
Youtube	4.4	22,601	DNF
WikiTalk	4.9	8,662	DNF
Skitter	12.7	86,326	DNF
Baidu	18.9	DNF	ROM
LiveJournal	52.2	DNF	ROM
Orkut	73.2	DNF	ROM
Twitter	1,345	DNF	ROM
Friendster	2,354	DNF	ROM
uk2007	1,485	ROM	ROM
ClueWeb09	17,060	ROM	ROM

Shortest Path Graph - Experiments

Q2: *How does QbS perform in terms of labelling size?*

Dataset	QbS	PPL	ParentPPL	$\vert G\vert$
Douban	2.98MB	0.4GB	0.8GB	2.5MB
DBLP	6.08MB	1.2GB	2.4GB	8.0MB
Youtube	22.2MB	1.7GB	–	23MB
WikiTalk	46.4MB	2.1GB	–	36MB
Skitter	52.7MB	9.2GB	–	85MB
Baidu	45.6MB	–	–	130MB
LiveJournal	93.6MB	–	–	329MB
Orkut	62.1MB	–	–	894MB
Twitter	1.54GB	–	–	9.0GB
Friendster	1.23GB	–	–	13.0GB
uk2007	2.06GB	–	–	24.8GB
ClueWeb09	31.9GB	–	–	58.2GB

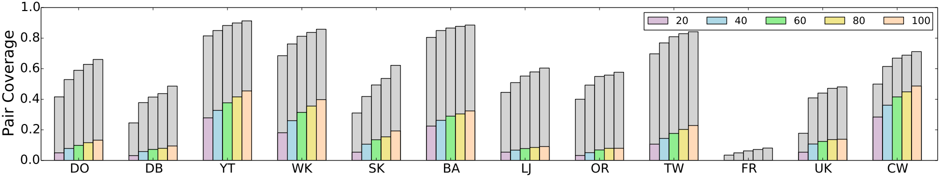
Shortest Path Graph - Experiments

Q3: *How does QbS perform in terms of query time?*

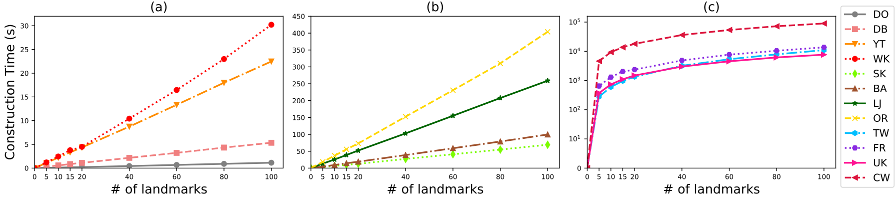
Dataset	Average Query Time			(ms.)
	QbS	PPL	ParentPPL	Bi-BFS
Douban	0.037	1.414	0.038	0.585
DBLP	0.097	1.782	0.052	2.995
Youtube	0.218	5.314	-	23.809
WikiTalk	0.693	3.536	-	6.984
Skitter	0.951	16.978	-	44.685
Baidu	0.845	-	-	174.412
LiveJournal	1.095	-	-	84.967
Orkut	4.237	-	-	207.541
Twitter	164.333	-	-	4,817.774
Friendster	11.972	-	-	3,600.362
uk2007	77.830	-	-	5,264.101
ClueWeb09	480.443	-	-	DNF

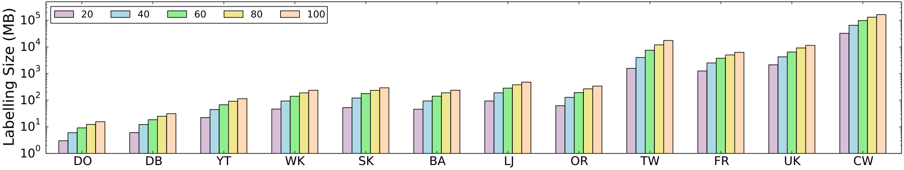
Shortest Path Graph - Experiments

Q4: *How well can "sketching" help improve the performance?*



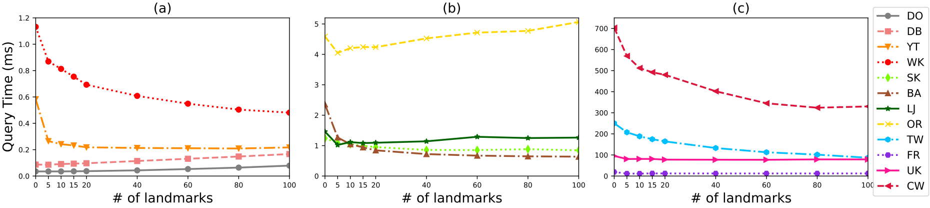
Q5: *How does the number of landmarks affect the performance?*





Shortest Path Graph - Experiments

Q5: *How does the number of landmarks affect the performance?*



Research Questions

1. How do two vertices in a graph connect with each other?
   - Point-to-point shortest path problem
   - ...
   - Shortest path graph problem

2. How does one vertex connect with other vertices?
   - Single source shortest path problem
   - ...
   - Top-k relative coverage problem

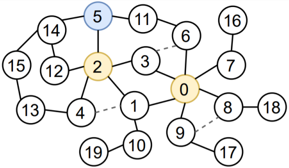
Top-k Relative Coverage Problem

Given a graph G and a vertex s,

- Centrality measures how important a vertex is in a network.

  Coverage centrality: $CC(u) = \{(s,t)|s,t \in V, u \in P_{st}\}$.
- However, vertices that are important in a network may not be important to a specific vertex.
- Thus, we find top-k vertices which are most influential in connecting a vertex s with other vertices, called *top-k relative coverage*.

  Relative coverage: $RC(u|s) = \{t|t \in V, u \in P_{st}\}$.



Top-k Relative Coverage - Applications

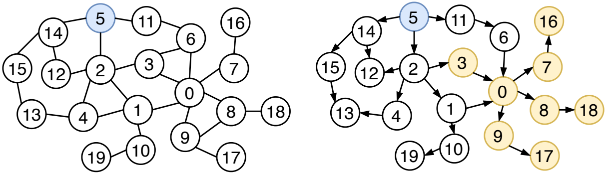
- Road network upgrade.
- Content spread enhancement.

Top-k Relative Coverage - Related Work

- Shortest-path-based centralities:
  - betweenness centrality [Freeman1977]
  - coverage centrality [Yoshida2014]
- Computing coverage centrality:
  - temporal coverage centrality [Takaguchi2016]
  - approximate coverage centrality [Chehreghan2018]
- Controlling coverage centrality:
  - coverage centrality maximization [Medya2018, D'Angelo2019]

Top-k Relative Coverage - Proposed Method

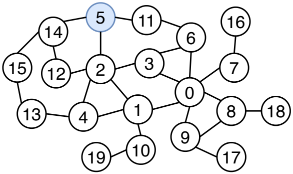
How to compute relative coverage of a vertex u w.r.t s?



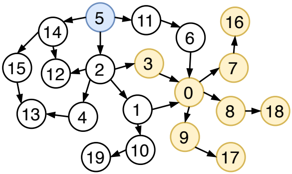
(a) (b)

How to compute relative coverage of a vertex u w.r.t s?

- Detect the cover set $T_{u|s} = \{t | t \in V, u \in P_{st}\}$ in a BFS.
  - A vertex v is in $T_{u|s}$, then $Succ_s(v)$ are in $T_{u|s}$.



(a)

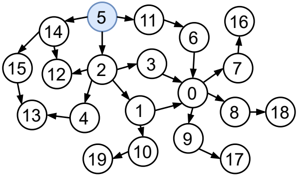


(b)

Top-k Relative Coverage - Proposed Method

Can we reduce the search space? Select candidates.

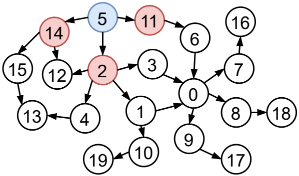
- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.



Top-k Relative Coverage - Proposed Method

Can we reduce the search space? Select candidates.

- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.
    - Which vertex can be v_1?

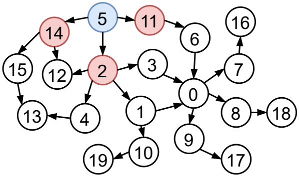


Sequence:
Candidates: {2, 11, 14}
RC(2|5) = 15;
RC(11|5) = 9;
RC(14|5) = 4;

Top-k Relative Coverage - Proposed Method

Can we reduce the search space? Select candidates.

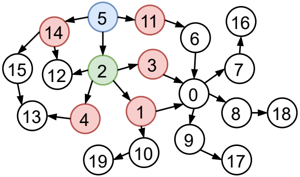
- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.
  - Which vertex can be v_1?
- In the top-k result $X_k = \{v_1, v_2, \ldots, v_k\}$, $Pred_s(v_k) \subseteq \{s, v_1, v_2, \ldots, v_{k-1}\}$.
  - Which vertex can be v_i?



Sequence:
Candidates: {2, 11, 14}
RC(2|5) = 15;
RC(11|5) = 9;
RC(14|5) = 4;

Can we reduce the search space? Select candidates.

- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.
  - Which vertex can be v_1?
- In the top-k result $X_k = \{v_1, v_2, \ldots, v_k\}$, $Pred_s(v_k) \subseteq \{s, v_1, v_2, \ldots, v_{k-1}\}$.
  - Which vertex can be v_i?

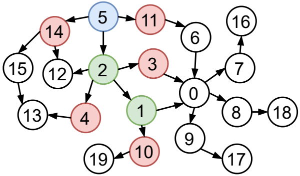


Sequence:{2}
Candidates: {1, 3, 4, 11, 14}
~~RC(2|5) = 15;~~
RC(1|5) = 10;
RC(11|5) = 9;
RC(3|5) = 8;
RC(14|5) = 4;
RC(4|5) = 2;

Top-k Relative Coverage - Proposed Method

Can we reduce the search space? Select candidates.

- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.
  - Which vertex can be v_1?
- In the top-k result $X_k = \{v_1, v_2, \ldots, v_k\}$, $Pred_s(v_k) \subseteq \{s, v_1, v_2, \ldots, v_{k-1}\}$.
  - Which vertex can be v_i?

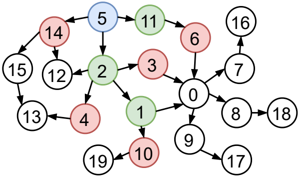


Sequence:{2, 1}
Candidates: {3, 4, 10, 11, 14}
~~RC(1|5) = 10;~~
RC(11|5) = 9;
RC(3|5) = 8;
RC(14|5) = 4;
RC(4|5) = 2;
RC(10|5) = 2;

Top-k Relative Coverage - Proposed Method

Can we reduce the search space? Select candidates.

- If $u' \in Succ_s(u)$, $T_{u'|s} \subseteq T_{u|s}$: $RC(u'|s) < RC(u|s)$.
  - Which vertex can be v_1?
- In the top-k result $X_k = \{v_1, v_2, \ldots, v_k\}$, $Pred_s(v_k) \subseteq \{s, v_1, v_2, \ldots, v_{k-1}\}$.
  - Which vertex can be v_i?

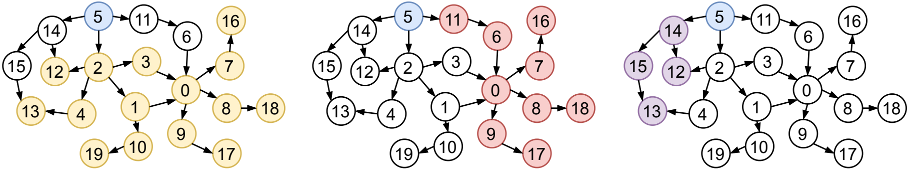


Sequence:{2, 1, 11}
Candidates: {3, 4, 6, 10, 14}
~~RC(11|5) = 9;~~
RC(3|5) = 8;
RC(6|5) = 8;
RC(14|5) = 4;
RC(4|5) = 2;
RC(10|5) = 2;

Top-k Relative Coverage - Proposed Method

How to compute RC of $U = \{u_1, u_2, ..., u_b\}$ w.r.t s efficiently?

Compute one-by-one: For each $u_i \in U$, compute $RC(u_i|s)$ by performing a BFS rooted in s.

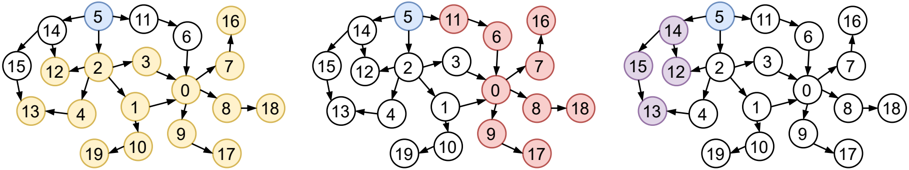


Top-k Relative Coverage - Proposed Method

How to compute RC of $U = \{u_1, u_2, ..., u_b\}$ w.r.t s efficiently?

Compute one-by-one: For each $u_i \in U$, compute $RC(u_i|s)$ by performing a BFS rooted in s.

- Sequential: $O(|U||E|)$ time
- Parallel: $O(|U||V|)$ space



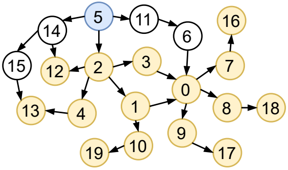
Top-k Relative Coverage - Proposed Method

To compute RC of $U = \{u_1, u_2, ..., u_b\}$ w.r.t s ...

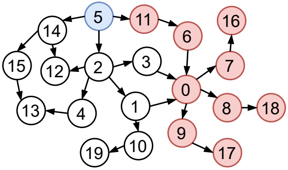
Bit-parallel!

For each $v \in V$, detect whose cover set v is in.

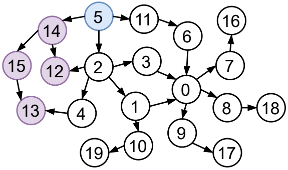
- $O(|E|)$ time and $O(|V|)$ space.



1 _ _ , 0 _ _



_ 1 _ , _ 0 _



_ _ 1, _ _ 0

Top-k Relative Coverage - Experiments

We evaluate OptCandRC against the baselines:

- AllRC: compute the relative coverage of all vertices
- CandRC: compute the relative coverage of candidates

Datasets: 6 real-world networks.

Dataset	$\lvert V \rvert$	$\lvert E \rvert$	avg.deg	max.deg	avg.dist	diameter
EuroRoad	1K	1K	2.5	10	18.7	62
Facebook	4K	88K	43.7	1,045	43.7	8
CondMat	21K	91K	8.5	280	5.3	15
USRoad	126K	162K	2.6	7	223.8	617
EmailEu	225K	340K	3.0	7,636	4.1	14
YouTube	1M	3M	5.3	28,754	5.3	24

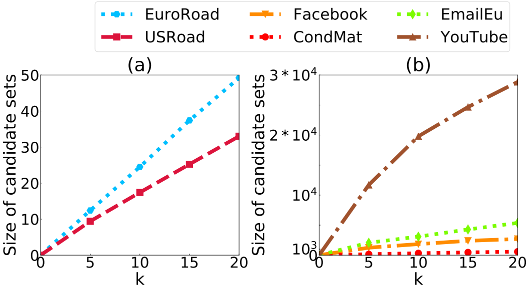
Top-k Relative Coverage - Experiments

Q1: *How efficient is our proposed method in terms of the size of candidates and query time?*

Dataset	$\frac{\lvert \cup_{i=1}^{k} C_i \rvert}{\lvert V \rvert}$	Average Query Time		
		AllRC	CandRC	optCandRC
EuroRoad	2.36%	42.74ms	1.30ms	0.75ms
Facebook	45.54%	5.924s	2.68s	0.038s
CondMat	1.50%	56.913s	0.939s	0.055s
USRoad	0.01%	DNF	0.248s	0.160s
EmailEu	1.37%	DNF	76.628s	4.259s
YouTube	1.74%	DNF	DNF	132.758s

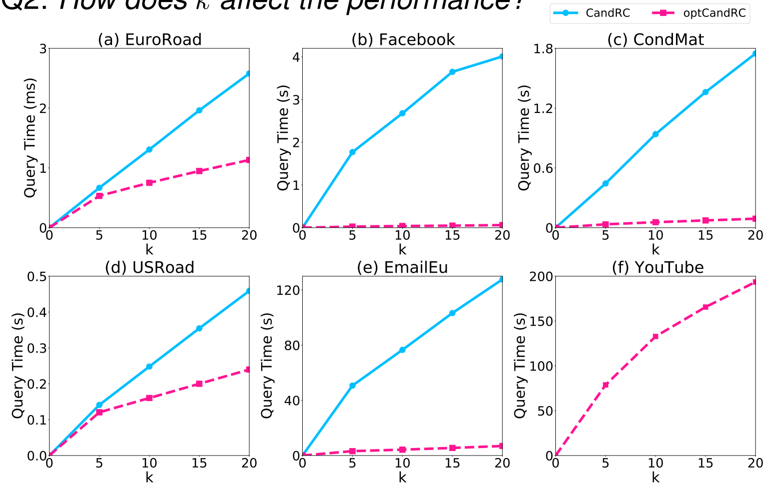
Top-k Relative Coverage - Experiments

Q2: *How does k affect the performance?*



Top-k Relative Coverage - Experiments

Q2: *How does k affect the performance?*



Conclusion and Future Work

Conclusion:

- study the shortest path graph problem for exploring the connections between two vertices
- study the top-k relative coverage problem for identifying influential vertices w.r.t one vertex

Future Work:

- study the shortest path graph problem on road networks
- identify important vertices w.r.t a set of vertices