THE AUSTRALIAN NATIONAL UNIVERSITY

Answering Shortest Path Distance Queries in Very Large Networks

Muhammad Farhan

School of Computing
Australian National University
muhammad.farhan@anu.edu.au

October 7, 2021

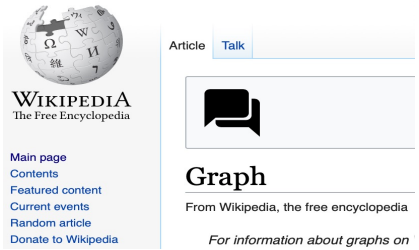
My Topic

- Shortest path distance queries in large-scale networks

My Topic

- Shortest path distance queries in large-scale networks

World Wide Web Networks



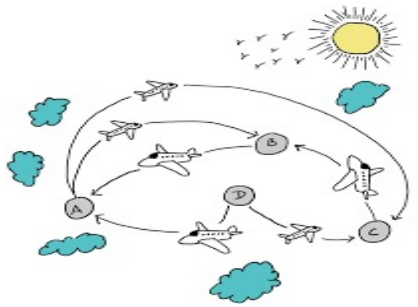
→ Context-Aware Web Search

Social Networks

→ Socially-Sensitive Search

→ Social Network Analysis
centrality, community search

Road Networks



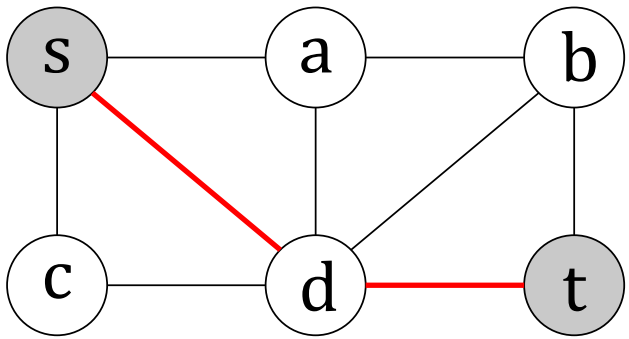
→ Route Navigation

- **Problem:**

  Given a graph $G = (V, E)$ and any two vertices $s, t \in V$, to answer the shortest path distance $d_G(s, t)$.

- **Problem:**

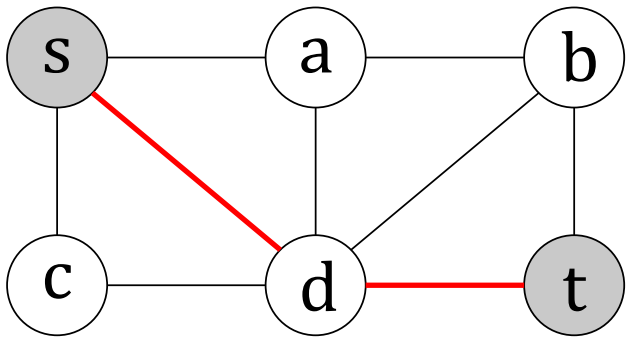
  Given a graph $G = (V, E)$ and any two vertices $s, t \in V$, to answer the shortest path distance $d_G(s, t)$.



$d_G(s, t) = 2$

- **Problem:**

  Given a graph $G = (V, E)$ and any two vertices $s, t \in V$, to answer the shortest path distance $d_G(s, t)$.



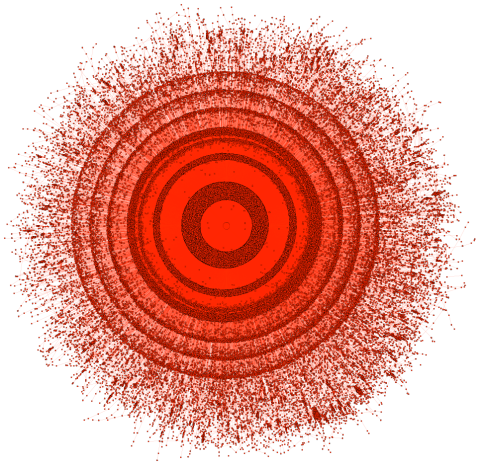
$d_G(s, t) = 2$

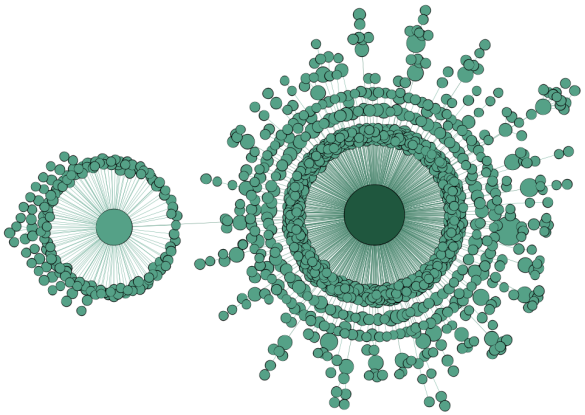
- **Challenge:**

  How to efficiently answer $d_G(s, t)$ in a very large network?

- **Challenge:**

  How to efficiently answer $d_G(s, t)$ in a very large network?



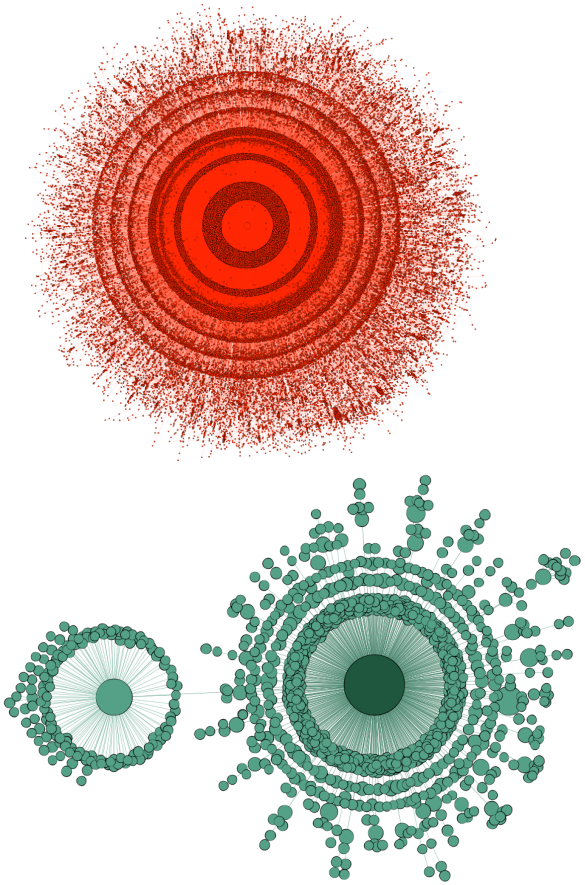


(two complex networks)

- *What kinds of networks?*
  - Complex networks

- **Challenge:**

  How to efficiently answer $d_G(s, t)$ in a very large network?



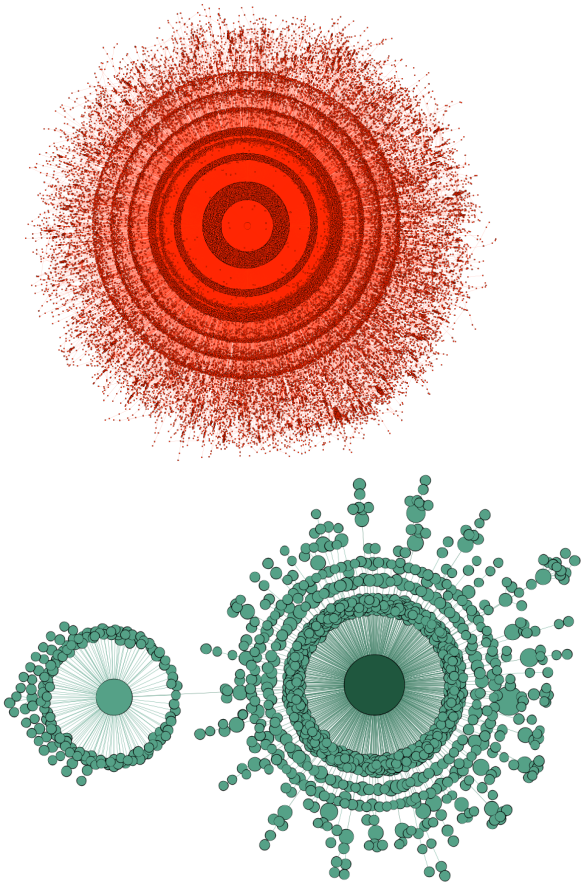
(two complex networks)

- *What kinds of networks?*
  - Complex networks
- *How large is a network?*
  - Billions of vertices and billions of edges

Shortest Path Distance Queries

- **Challenge:**

  How to efficiently answer $d_G(s, t)$ in a very large network?

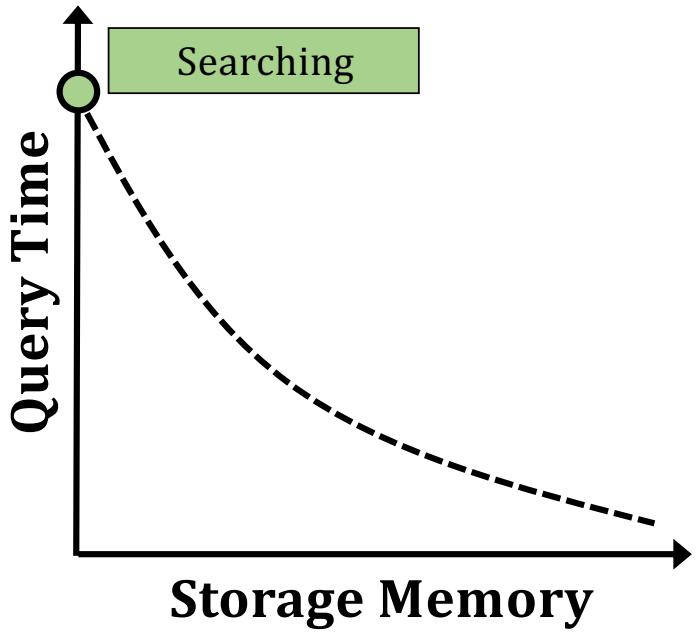


(two complex networks)

- *What kinds of networks?*
  - Complex networks
- *How large is a network?*
  - Billions of vertices and billions of edges
- *How efficiently is it answered?*
  - In the order of milliseconds

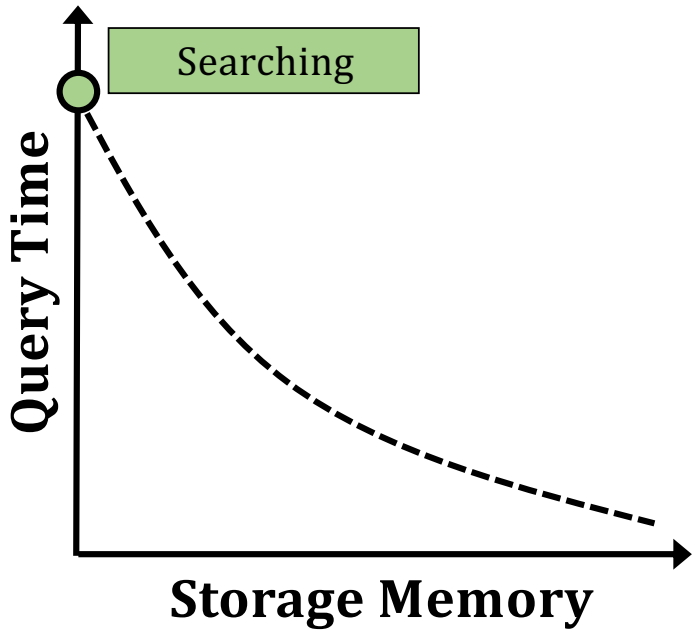
Shortest Path Distance Queries - Approaches

- Search over a graph (i.e. Dijkstra search)



Shortest Path Distance Queries - Approaches

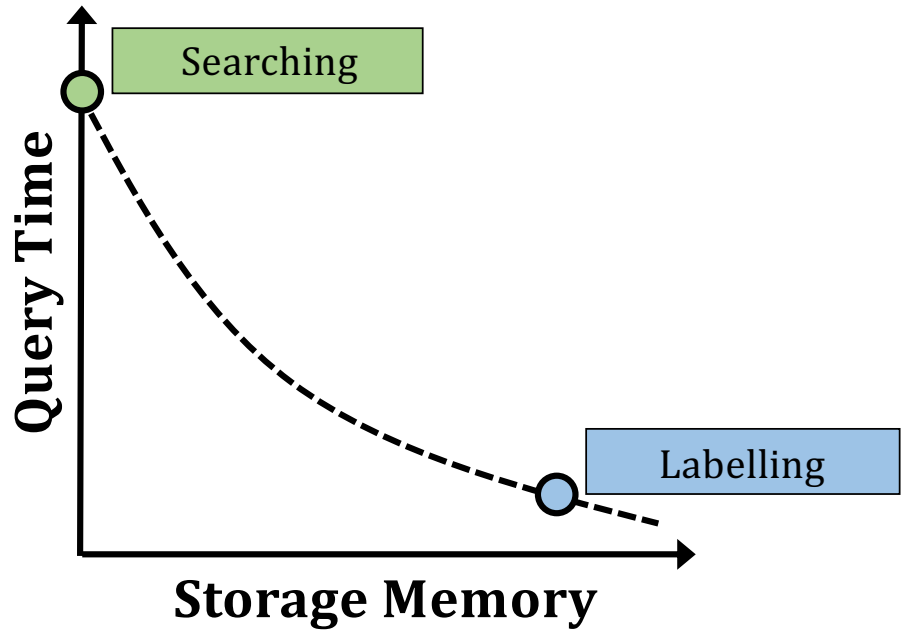
- Search over a graph (i.e. Dijkstra search)
  - slow querying
  - no memory



Shortest Path Distance Queries - Approaches

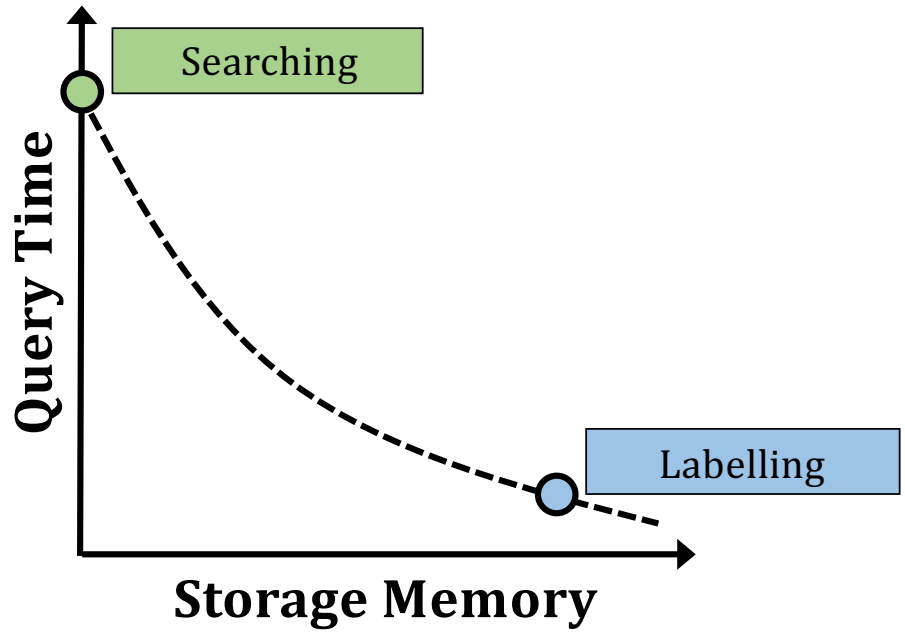
THE AUSTRALIAN NATIONAL UNIVERSITY

- Search over a graph (i.e. Dijkstra search)
  - slow querying
  - no memory
- precompute labelling (e.g. pair-wise distances)



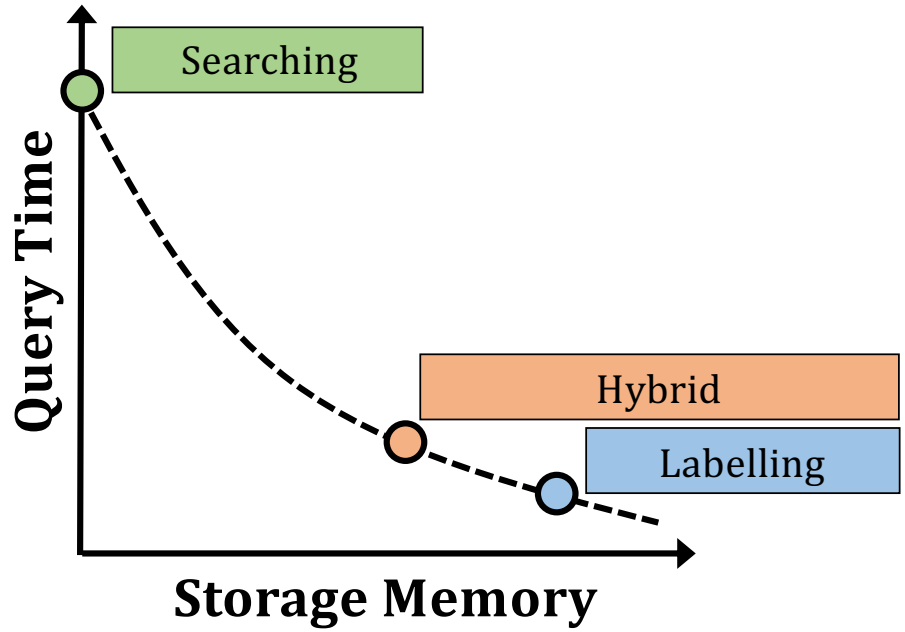
Shortest Path Distance Queries - Approaches

- Search over a graph (i.e. Dijkstra search)
  - slow querying
  - no memory
- precompute labelling (e.g. pair-wise distances)
  - fast querying
  - quadratic memory
  - limited scalability



Shortest Path Distance Queries - Approaches

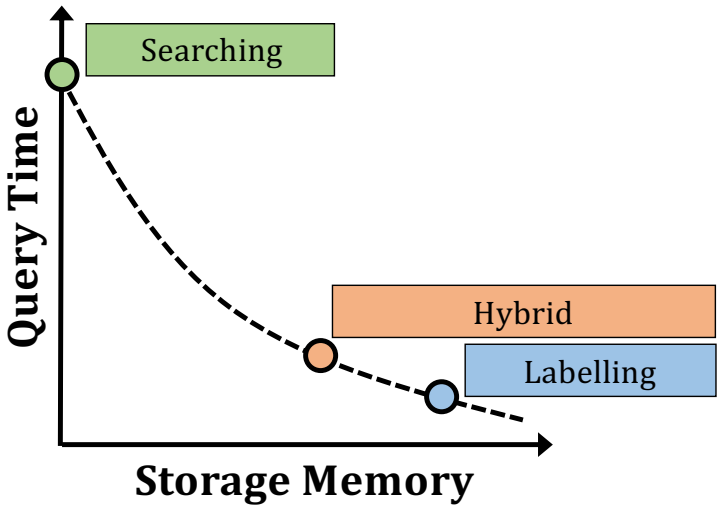
- Search over a graph (i.e. Dijkstra search)
  - slow querying
  - no memory
- precompute labelling (e.g. pair-wise distances)
  - fast querying
  - quadratic memory
  - limited scalability
- hybrid (i.e. combine labelling with search)
  - a better trade-off



Shortest Path Distance Queries - Research Goal

THE AUSTRALIAN NATIONAL UNIVERSITY

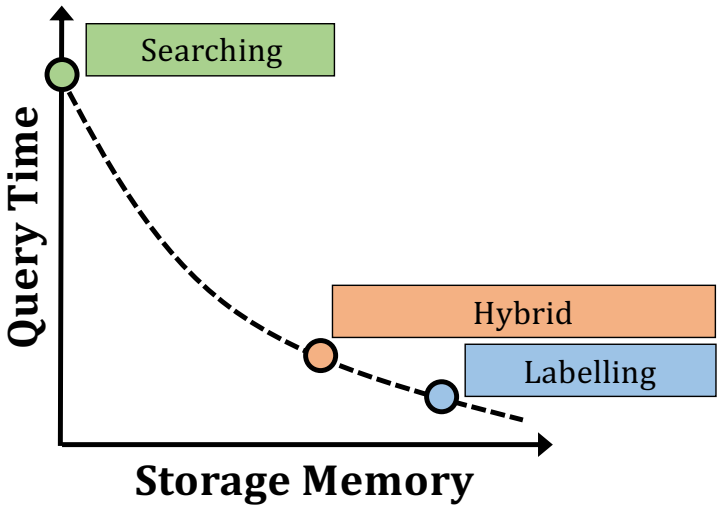
- **Scalable**
  - can handle networks with billions of vertices and edges
- **Fully dynamic**
  - can handle edge insertions/deletions
- **Efficient**
  - can answer exact distance queries in milliseconds
  - can reflect graph changes in milliseconds



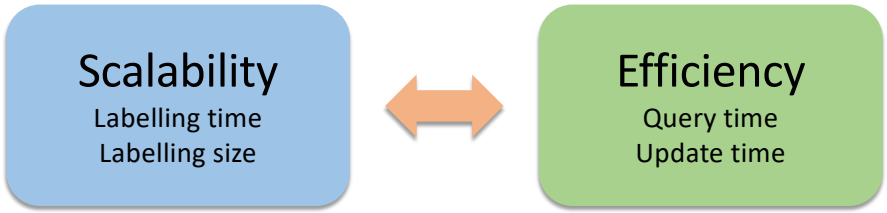
Shortest Path Distance Queries - Research Goal

THE AUSTRALIAN NATIONAL UNIVERSITY

- **Scalable**
  - can handle networks with billions of vertices and edges
- **Fully dynamic**
  - can handle edge insertions/deletions
- **Efficient**
  - can answer exact distance queries in milliseconds
  - can reflect graph changes in milliseconds



a good trade-off



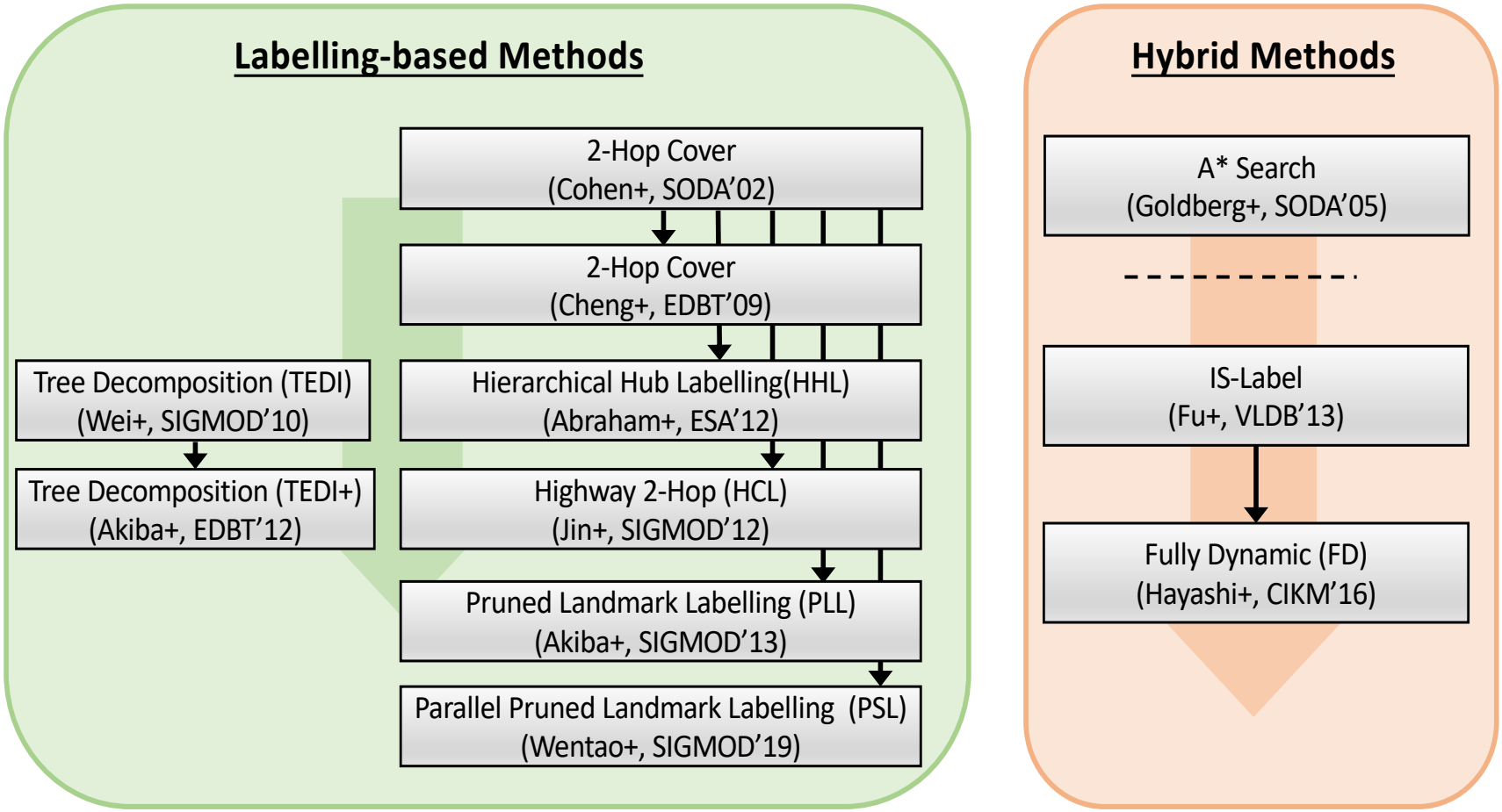
Main Contributions

1. A scalable method for answering exact distance queries on static networks
2. Efficient methods for answering exact distance queries on dynamic networks
3. An efficient batch-dynamic method to answer exact distance queries on dynamic networks undergoing batch of updates

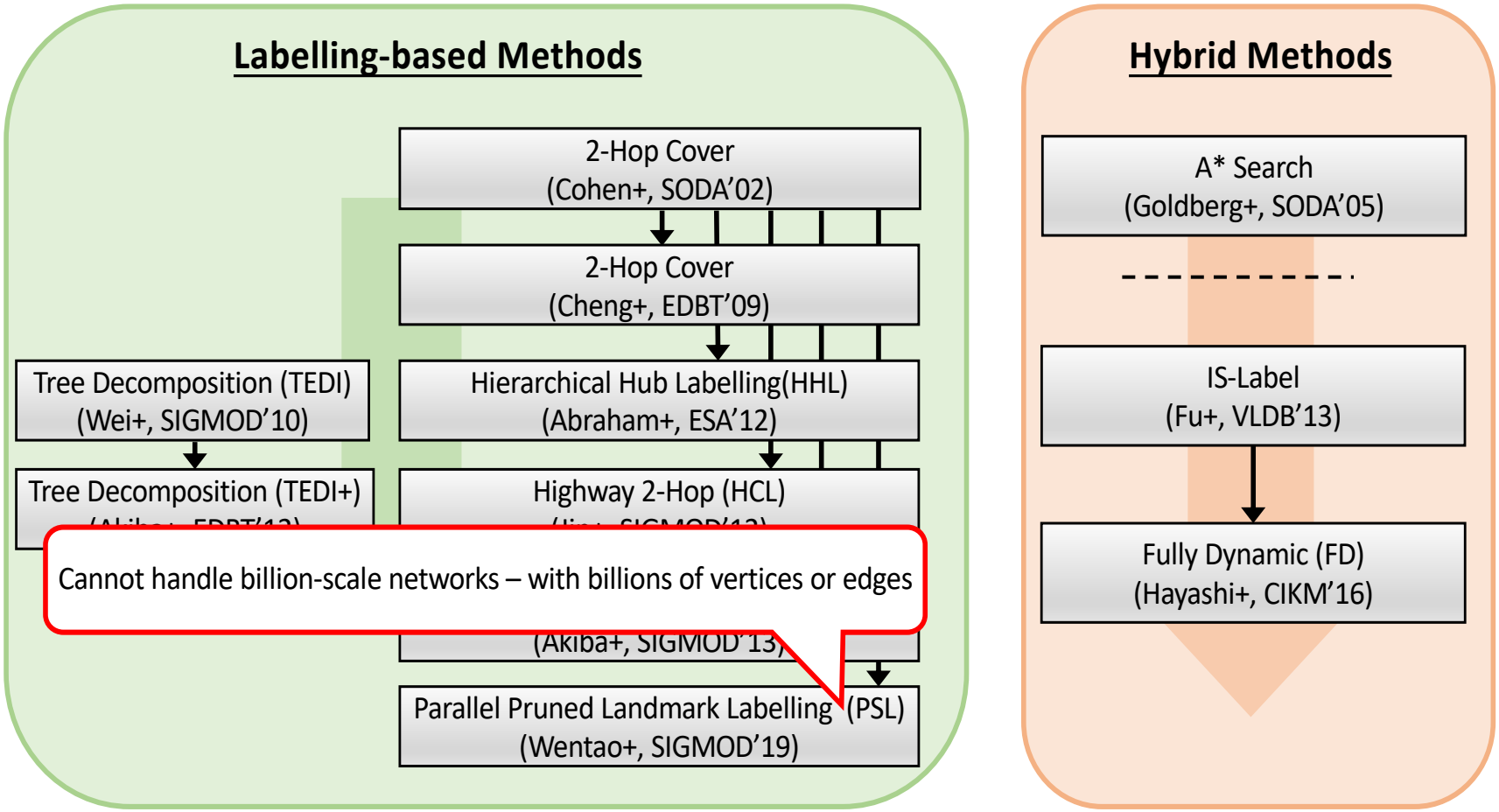
Answering Exact Shortest Path Distance Queries on Static Graphs [1]

[1] **Farhan, M.**, Wang, Q., Lin, Y., Mckay, B. "A Highly Scalable Labelling Approach for Exact Distance Queries in Complex Networks." 22nd International Conference on Extending Database Technology (EDBT), 2019.

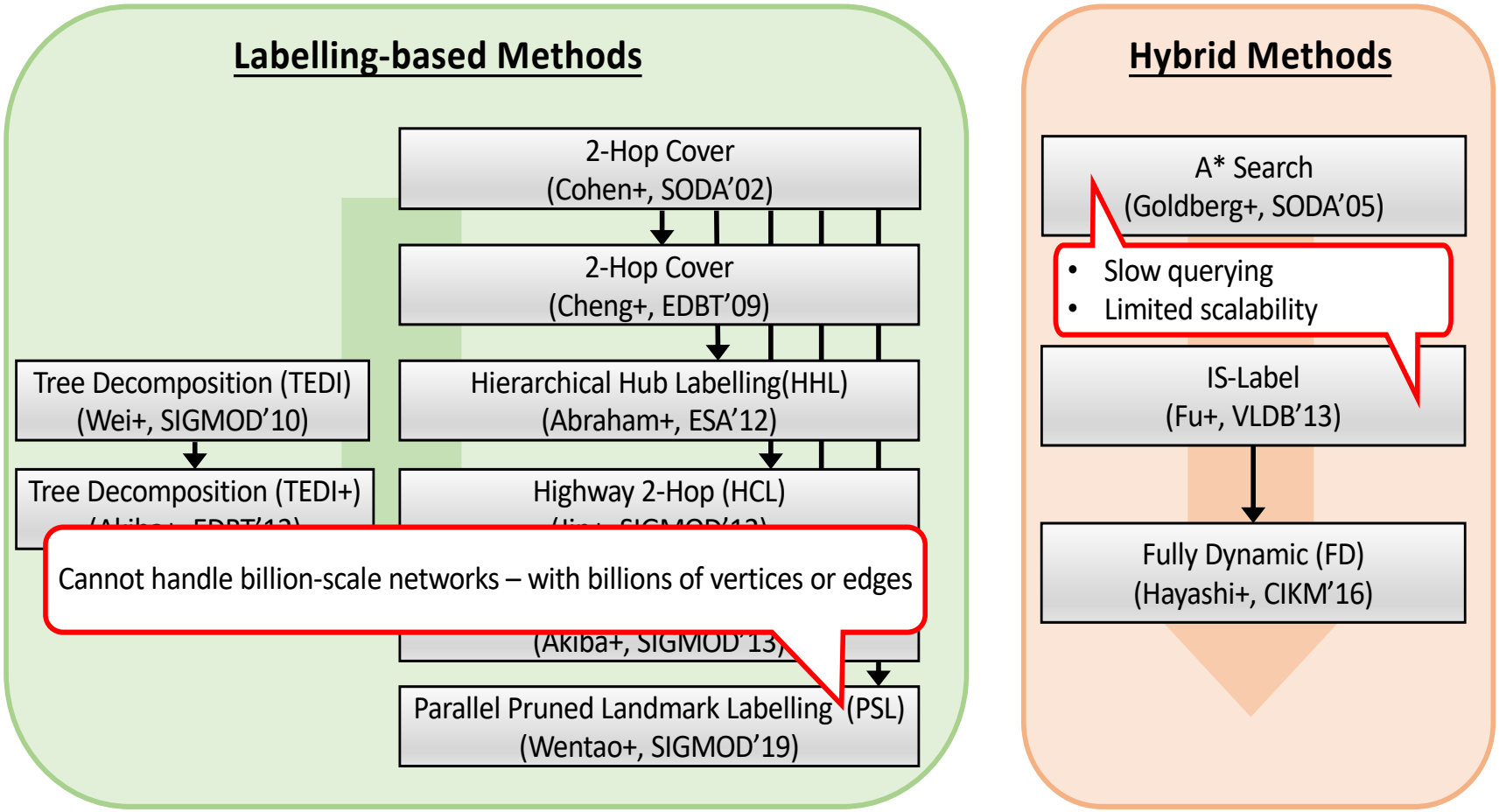
Existing Methods



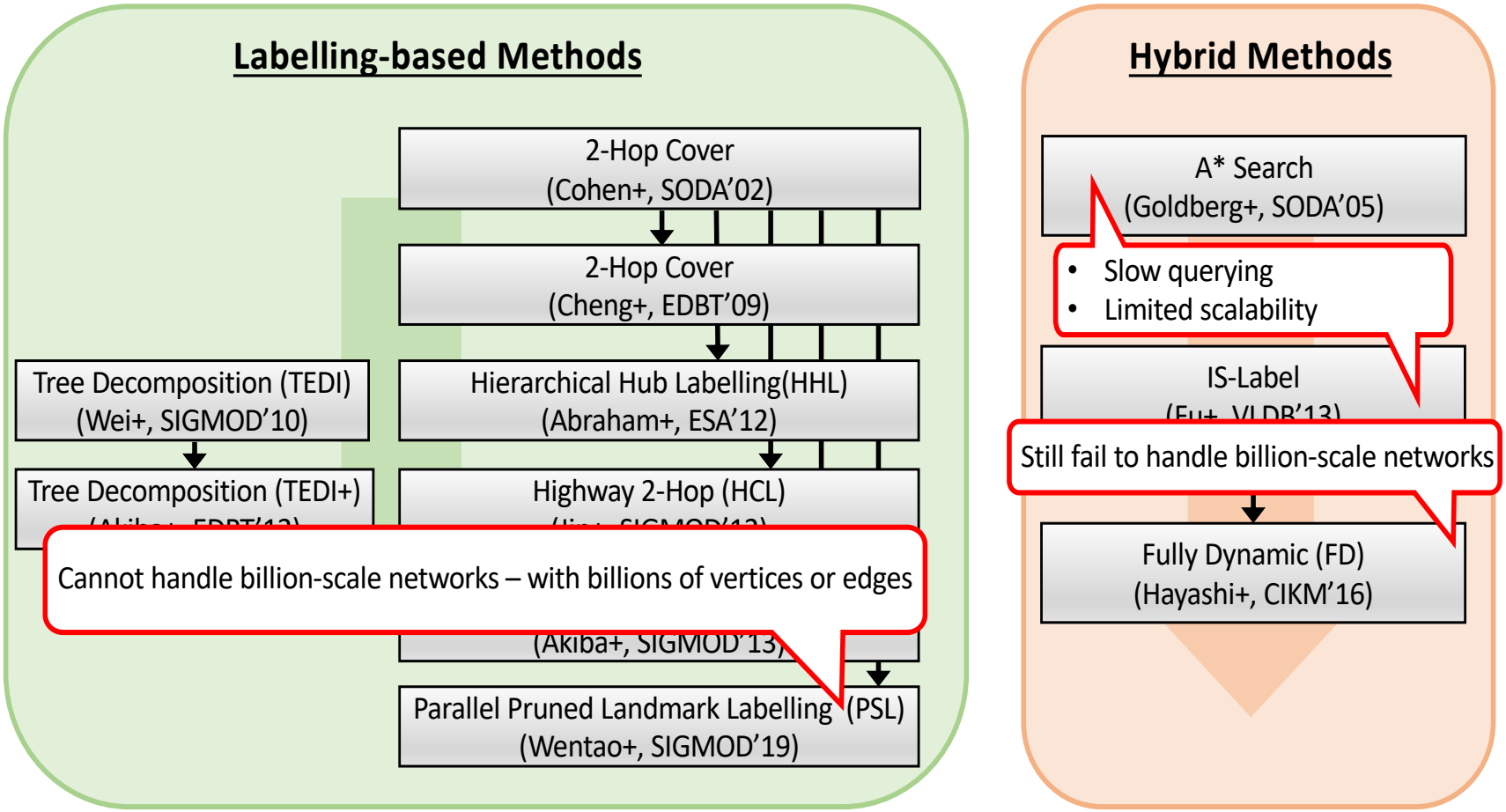
Existing Methods



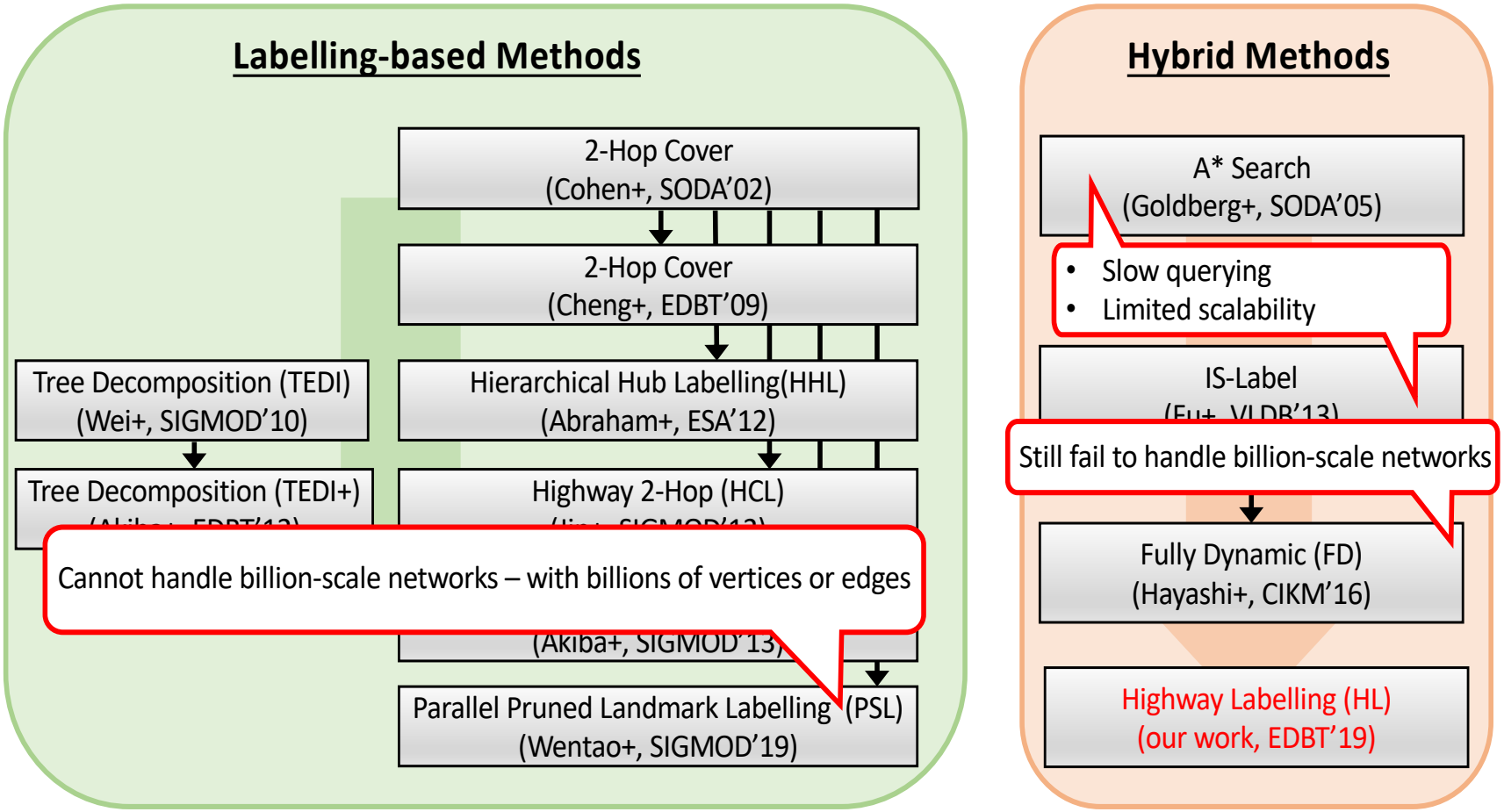
Existing Methods



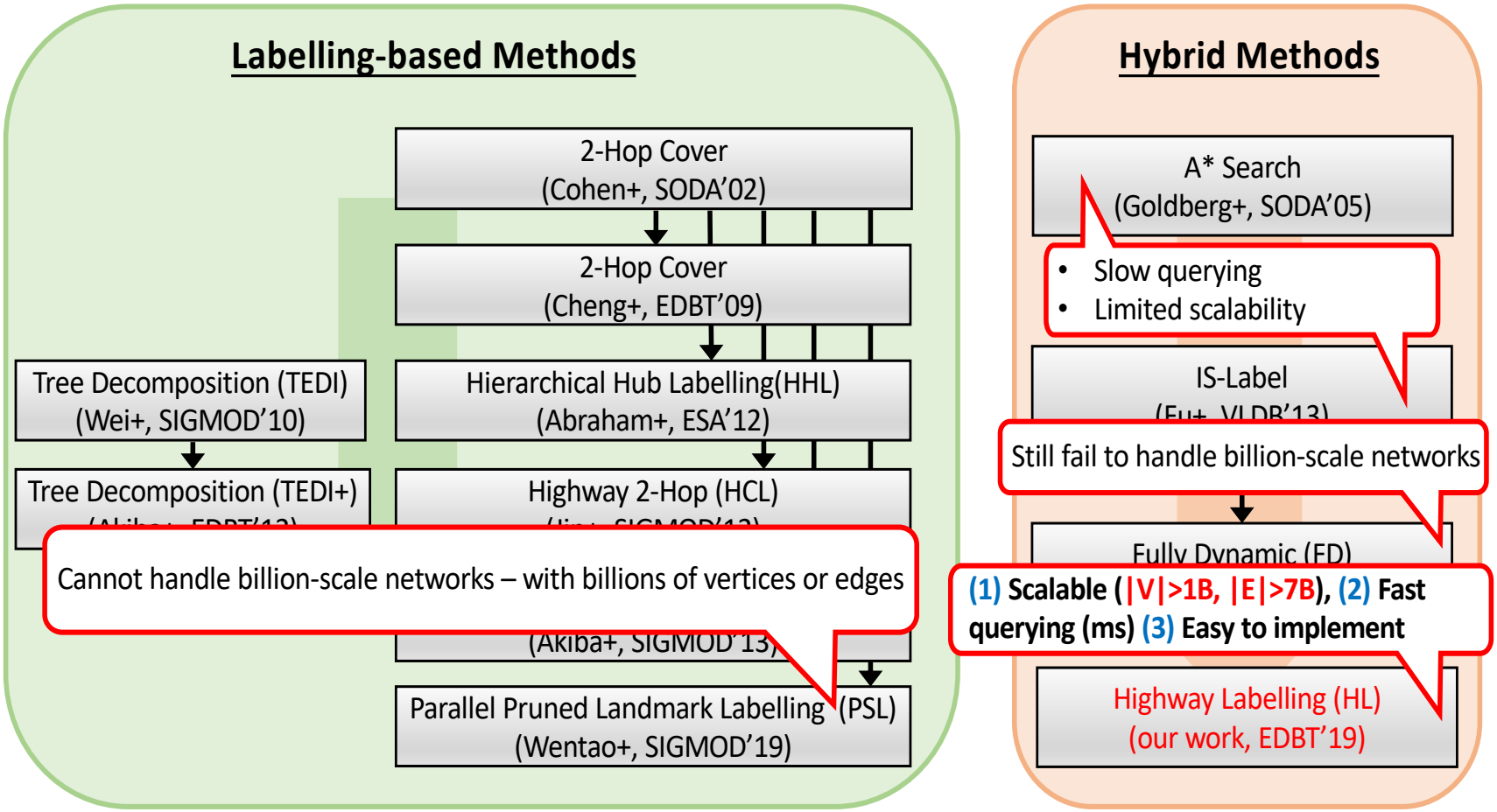
Existing Methods



Existing Methods



Existing Methods

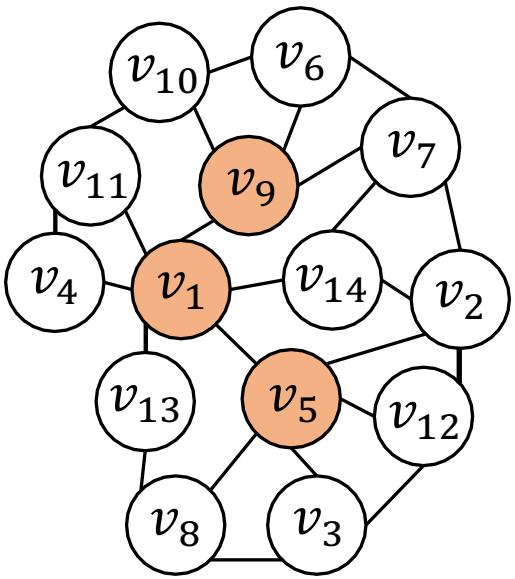


- **Distance Labelling:**

  Given a set of landmarks $R \subseteq V$, precompute a label $L(v)$ for every vertex $v \in V$ in G.

- **Distance Labelling:**

  Given a set of landmarks $R \subseteq V$, precompute a label $L(v)$ for every vertex $v \in V$ in G.



$R = \{v_1, v_5, v_9\}$

$L(v_1)$

Landmark	v_5	v_9
Distance	1	1

$L(v_2)$

Landmark	v_1	v_5	v_9
Distance	2	1	2

$d_G(v_5, v_2) = 1$

$L(v_3)$

Landmark	v_1	v_5	v_9
Distance	2	1	3

.....

- **2-Hop Cover Labelling:**

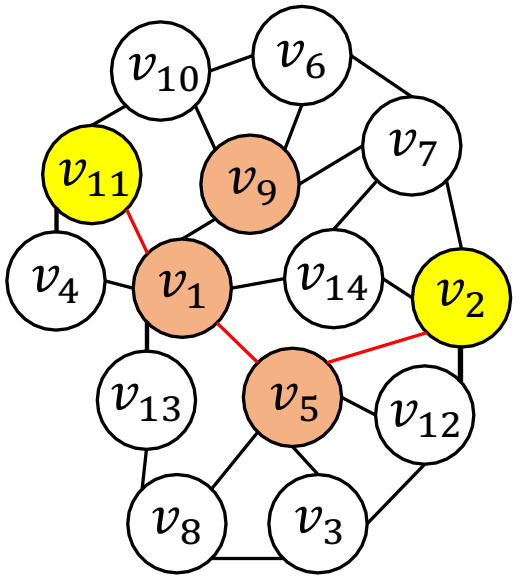
  Distance through a common landmark in $L(s)$ and $L(t)$.

$$d_G(s,t) = \min_{r \in L(s) \cap L(t)} \{\delta(r,s) + \delta(r,t)\}$$

- **2-Hop Cover Labelling:**

  Distance through a common landmark in $L(s)$ and $L(t)$.

$$d_G(s,t) = \min_{r \in L(s) \cap L(t)} \{\delta(r,s) + \delta(r,t)\}$$



$R = \{v_1, v_5, v_9\}$

$L(v_2)$

Landmark	v_1	v_5	v_9
Distance	2	1	2

$L(v_{11})$

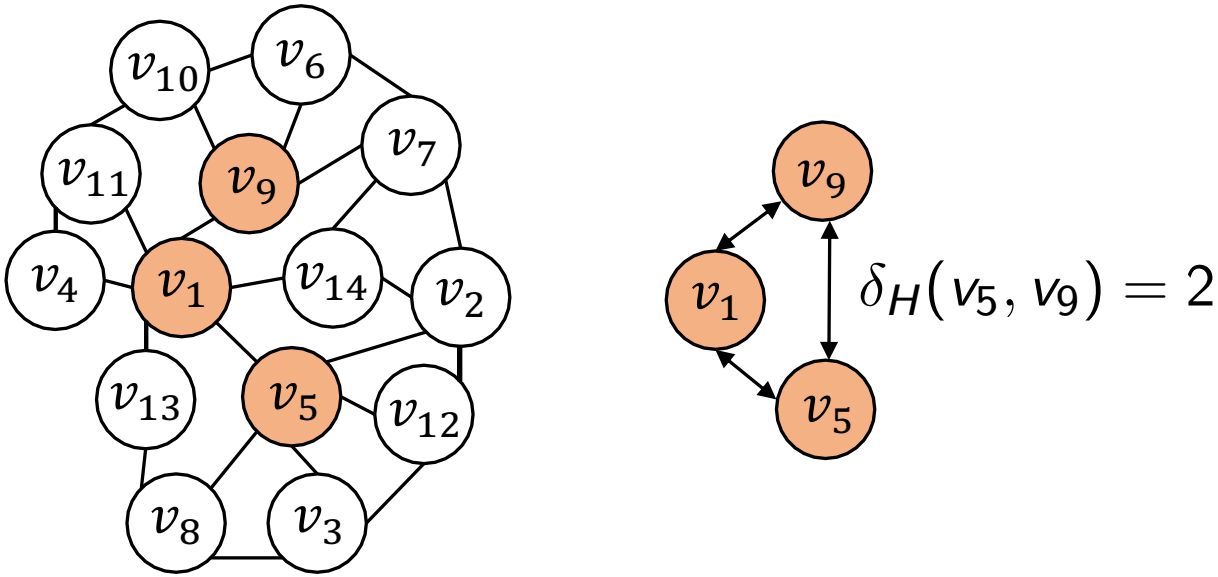
Landmark	v_1
Distance	1

$$d_G(v_2, v_{11}) = d_G(v_1, v_2) + d_G(v_1, v_{11}) = 3$$

THE AUSTRALIAN NATIONAL UNIVERSITY

Definition (Highway)

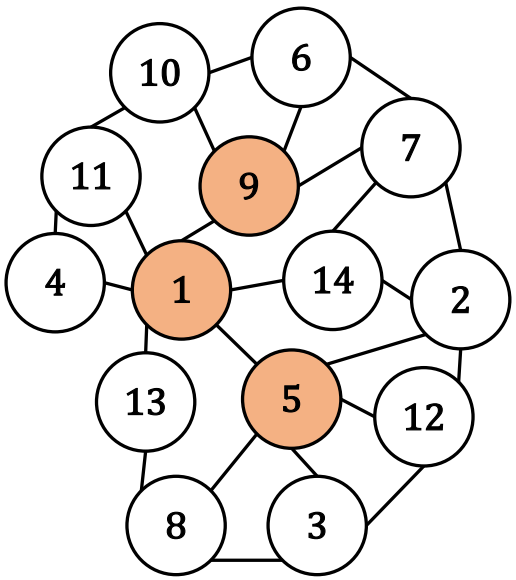
A highway H is a pair (R, δ_H), where R is a set of landmarks and δ_H is a *distance decoding function*, i.e. $\delta_H : R \times R \to \mathbb{N}^+$, such that for any $\{r_1, r_2\} \subseteq R$ we have $\delta_H(r_1, r_2) = d_G(r_1, r_2)$.



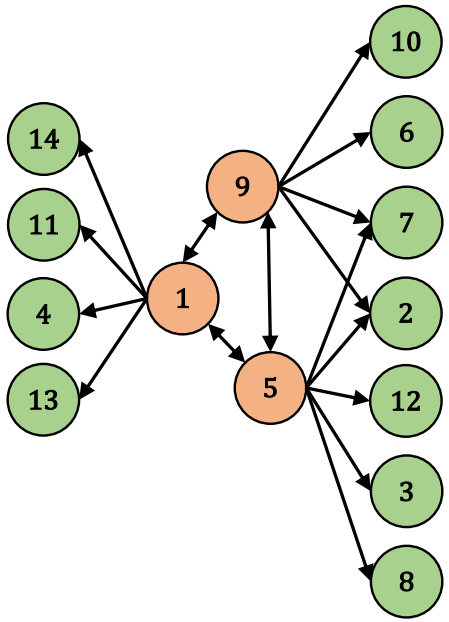
Highway Cover Labelling Framework

Definition (Highway Cover)

Let $G = (V, E)$ be a graph and $H = (R, \delta_H)$ a highway. For any vertex $u \in V \backslash R$ and any $r \in R$, there must exist $r' \in R$ in $L(u)$ such that r' is on a shortest path between u and r (r and r' may be the same).



Graph



Highway Cover

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

Labelling

Highway Cover Labelling Framework

THE AUSTRALIAN NATIONAL UNIVERSITY

Definition (Highway Cover Labelling Problem)

Given a graph G and a highway H over G, the *highway cover labelling problem* is to efficiently construct a highway cover labelling L.

Definition (Highway Cover Labelling Problem)

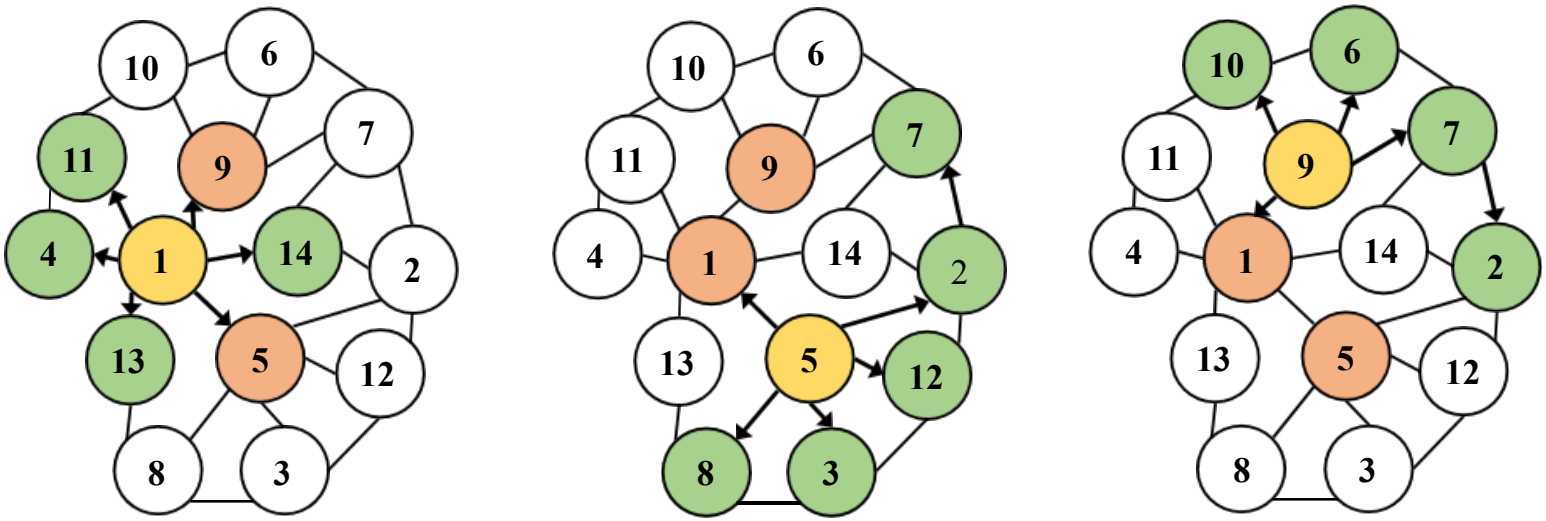
Given a graph G and a highway H over G, the *highway cover labelling problem* is to efficiently construct a highway cover labelling L.

- Some properties of labelling methods:

Methods	Ordering dependent?	Highway Cover minimal?	Parallel?
HL (ours)	**no**	**yes**	**landmarks**
FD [CIKM2016]	no	no	neighbours
IS-L [VLDB2013]	yes	no	no
PLL [SIGMOD2013]	yes	no	neighbours
PSL [SIGMOD2019]	yes	no	distance
HDB [VLDB2014]	yes	no	no
HHL [ESA2012]	yes	no	no

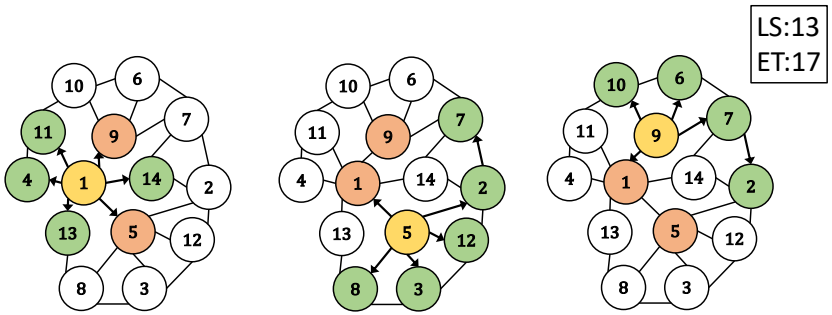
Highway Cover Labelling - Our Algorithm

- $L \leftarrow$ an empty index
- Conduct a BFS from each landmark $r \in R$
  - Add a distance entry $(r, \delta(r, v))$ into $L(v)$ the label of $v \in V \backslash R$ iff there does not exist any other landmark $r' \in R$ that appears in a shortest path between r and u.

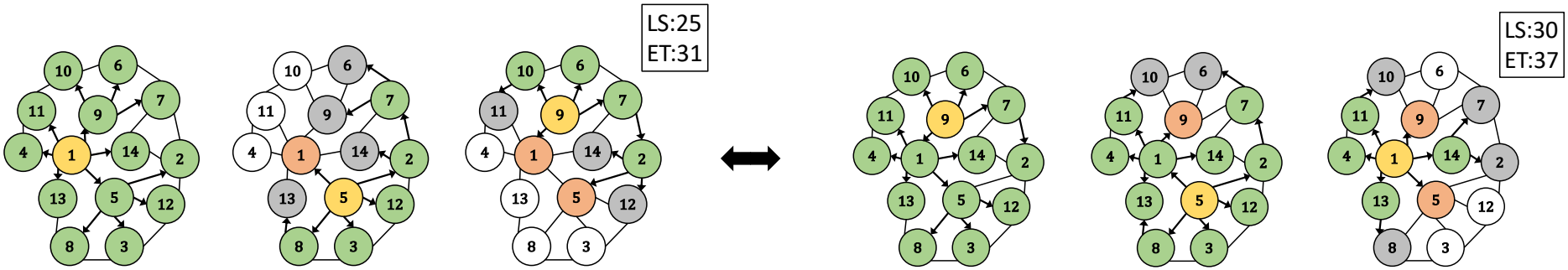


Order Independence - Properties

- Highway cover Labelling (HL) algorithm (ours)



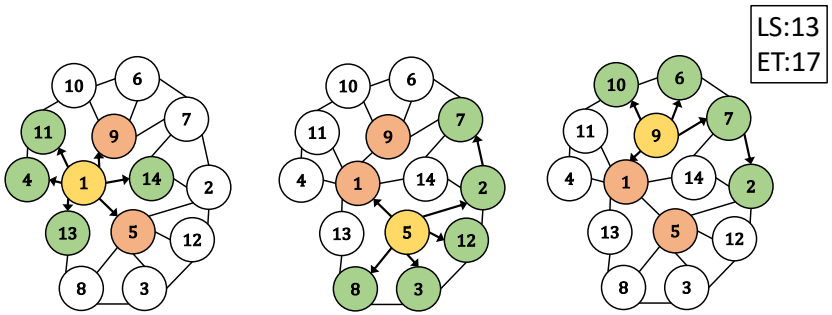
- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]



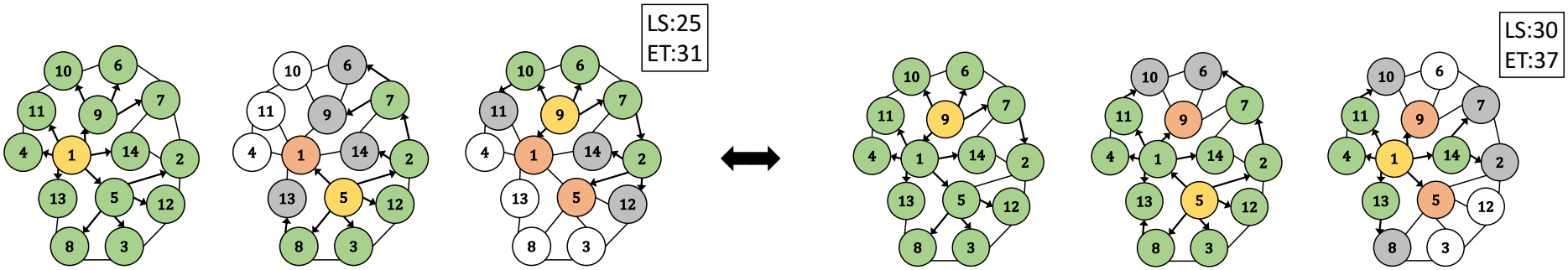
Labeled Vertices Pruned Landmarks Root Landmarks

Minimality - Properties

- Highway cover Labelling (HL) algorithm (ours)



- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]



Labeled Vertices Pruned Landmarks Root Landmarks

Parallelism - Properties

- Parallel Highway cover Labelling (HL-P) algorithm (ours)
  - **Landmark-based Parallelism**: Run parallel BFSs from multiple landmarks
- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]
  - **Neighbor-based Parallelism**: Perform BFSs from a given landmark and up to 64 of its neighbors simultaneously.
- Parallel Shortest Distance Labelling (PSL) algorithm [SIGMOD2019]
  - **Distance-based Parallelism**: Perform BFSs from a given landmark and label vertices that are at distance $[1, D]$ in parallel.

Exact Distance Querying Framework

Two steps for answering $d_G(s,t)$ in a graph G:

(1) Computing an upper bound $d_{st}^{\top}$ of $d_G(s,t)$ using the highway cover distance labelling;

(2) Computing $d_G(s,t)$ using a distance-bounded shortest-path search over a sparsified graph $G[V \setminus R]$.

Exact Distance Querying Framework

Two steps for answering $d_G(s,t)$ in a graph G:

(1) Computing an upper bound $d_{st}^{\top}$ of $d_G(s,t)$ using the highway cover distance labelling;

(2) Computing $d_G(s,t)$ using a distance-bounded shortest-path search over a sparsified graph $G[V \backslash R]$.

Definition

The *bounded distance querying problem* is to efficiently compute the shortest path distance between s and t over $G' = G[V \backslash R]$ under the upper bound $d_{st}^{\top}$ such that,

$$d_G(s,t) = \begin{cases} d_{G'}(s,t), & \text{if } d_{G'}(s,t) \leq d_{st}^{\top} \\ d_{st}^{\top}, & \text{otherwise} \end{cases}$$

Computing Upper Bounds

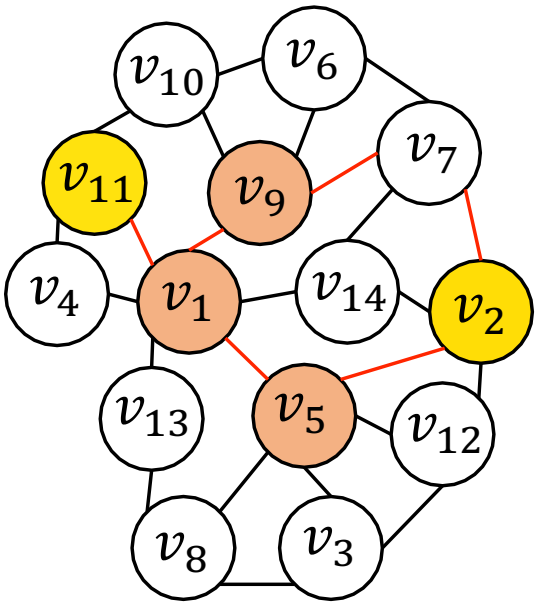
- Find a path of the minimal length through a highway H:

$$d_{st}^{\top} = \min_{\substack{(r_i,\delta(r_i,s))\in L(s) \\ (r_j,\delta(r_j,t))\in L(t)}} \{\delta(r_i, s) + \delta_H(r_i, r_j) + \delta(r_j, t)\}$$

Computing Upper Bounds

- Find a path of the minimal length through a highway H:

$$d_{st}^{\top} = \min_{\substack{(r_i,\delta(r_i,s))\in L(s)\\(r_j,\delta(r_j,t))\in L(t)}} \{\delta(r_i,s) + \delta_H(r_i,r_j) + \delta(r_j,t)\}$$



What is $d_{st}^{\top}$ for $s = v_2$ and $t = v_{11}$?

$L(v_2)$

Landmark	v_5	v_9
Distance	1	2

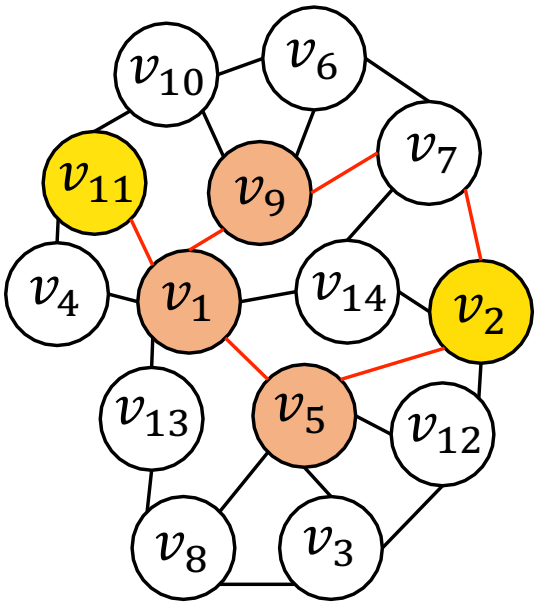
$L(v_{11})$

Landmark	v_1
Distance	1

Computing Upper Bounds

- Find a path of the minimal length through a highway H:

$$d_{st}^{\top} = \min_{\substack{(r_i,\delta(r_i,s))\in L(s)\\(r_j,\delta(r_j,t))\in L(t)}} \{\delta(r_i,s) + \delta_H(r_i,r_j) + \delta(r_j,t)\}$$



What is $d_{st}^{\top}$ for $s = v_2$ and $t = v_{11}$?

$L(v_2)$

Landmark	v_5	v_9
Distance	1	2

$v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_{11}$: length 3

$L(v_{11})$

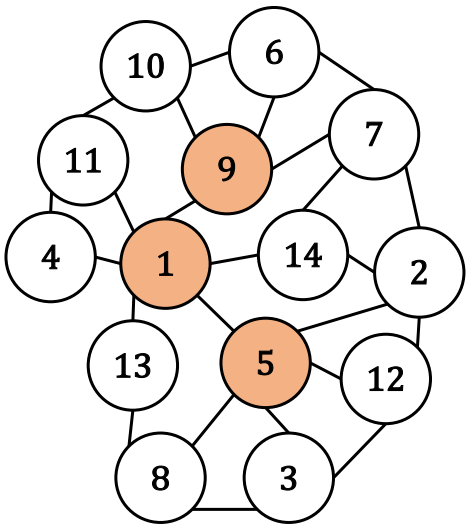
Landmark	v_1
Distance	1

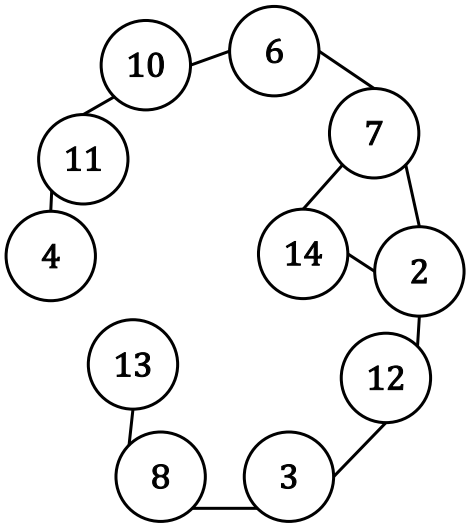
$v_2 \rightarrow v_9 \rightarrow v_1 \rightarrow v_{11}$: length 4

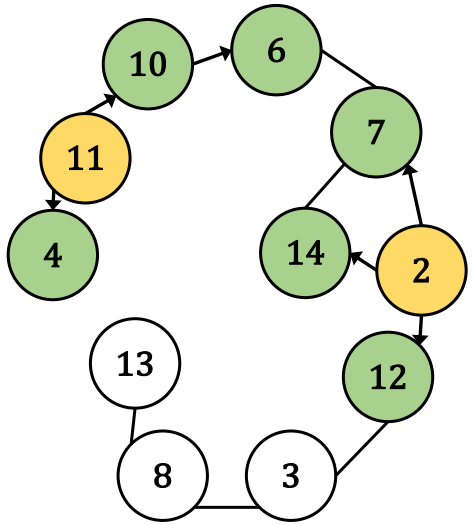
$d_{st}^{\top} = 3$

Distance Bounded Shortest Path Search

- Sparsify graph G by removing all landmarks in R, i.e. $G' = G[V \backslash R]$
- Conduct a bidirectional search on G' which is bounded by $d_{st}^{\top} - 1$

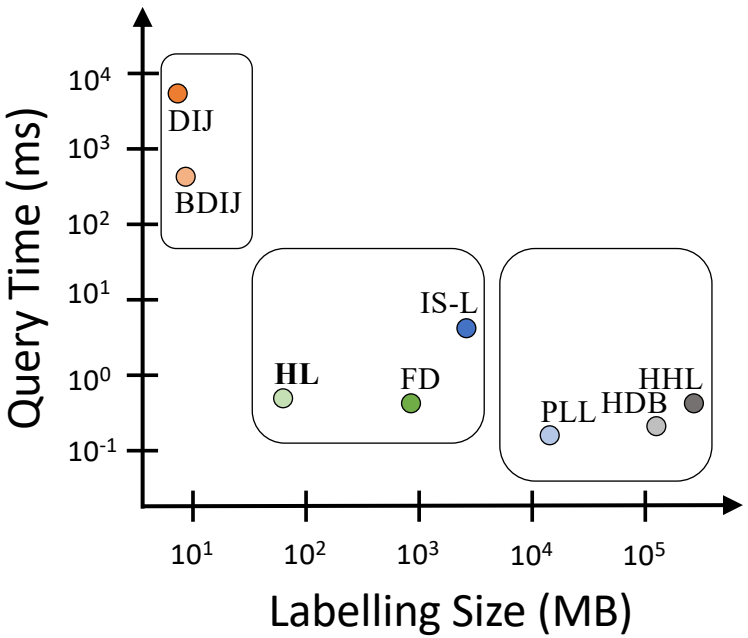


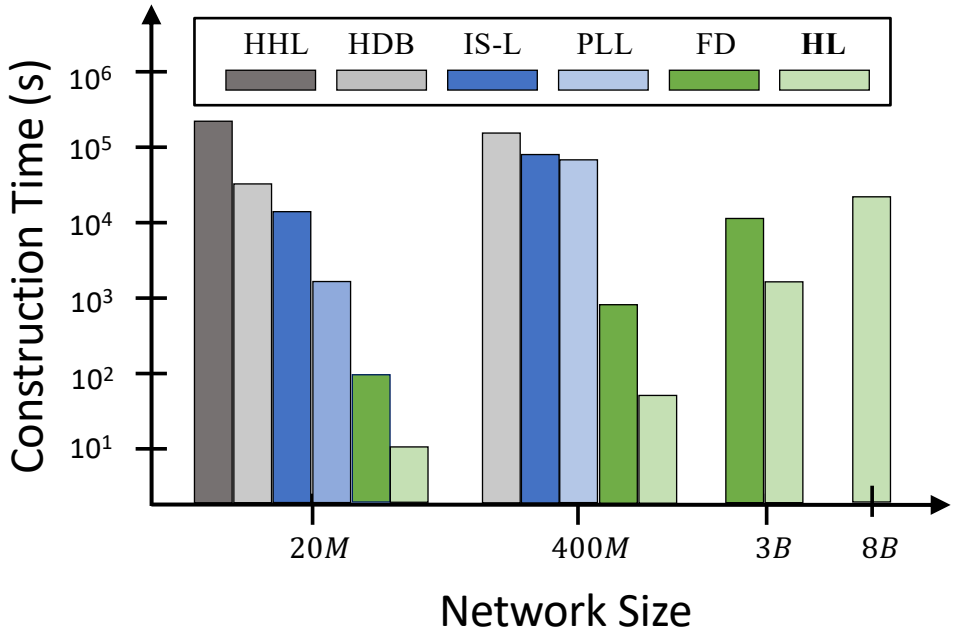




$d_G(2, 11) = 3$

Empirical Evaluation





(Note that, the left figure is based on networks of sizes up to 400M edges)

- Trade-offs among query time, labelling size and construction time.

Empirical Evaluation

THE AUSTRALIAN NATIONAL UNIVERSITY

Dataset			Construction Time [sec]					Query Time [ms]					Labelling Size			
	$\|V\|$	$\|E\|$	HL-P	HL	FD	PLL	IS-L	HL	FD	PLL	IS-L	Bi-BFS	HL	FD	PLL	IS-L
Skitter	1.7M	11M	**2**	**13**	30	638	1042	**0.067**	0.043	0.008	3.556	3.504	**42 MB**	202 MB	2.5 GB	507 MB
Flickr	1.7M	16M	**2**	**14**	41	1330	8359	**0.015**	0.028	0.01	33.760	4.155	**34 MB**	178 MB	3.7 GB	679 MB
Hollywood	1.1M	114M	**3**	**17**	107	31855	DNF	**0.047**	0.075	0.051	-	6.956	**28 MB**	293 MB	13 GB	-
Orkut	3.1M	117M	**10**	**62**	366	DNF	DNF	**0.224**	0.251	-	-	21.086	**70 MB**	756 MB	-	-
Enwiki	4.3M	101M	**9**	**77**	308	22080	DNF	**0.190**	0.131	0.027	-	19.423	**83 MB**	743 MB	12 GB	-
Livejournal	4.8M	69M	**9**	**77**	166	DNF	20583	**0.088**	0.111	-	56.847	17.264	**123 MB**	778 MB	-	3.8 GB
Indochina	7.4M	194M	**8**	**50**	144	9456	DNF	**1.905**	1.803	0.02	-	9.734	**81 MB**	999 MB	21 GB	-
IT	41M	1.2B	**66**	**304**	1623	DNF	DNF	**2.684**	2.118	-	-	92.187	**855 MB**	5.6 GB	-	-
Twitter	42M	1.5B	**133**	**1380**	1838	DNF	DNF	**1.424**	0.432	-	-	426.949	**1.2 GB**	4.8 GB	-	-
Friendster	66M	1.8B	**135**	**2229**	9661	DNF	DNF	**1.091**	1.435	-	-	534.576	**2.5 GB**	11.8 GB	-	-
UK	106M	3.7B	**110**	**1124**	6201	DNF	DNF	**11.84**	18.979	-	-	355.688	**1.8 GB**	14.1 GB	-	-
Clueweb09	1.7B	7.8B	**4236**	**28124**	DNF	DNF	DNF	**0.309**	-	-	-	-	**4.7 GB**	-	-	-

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Construction Time [sec]					Query Time [ms]					Labelling Size			
			HL-P	HL	FD	PLL	IS-L	HL	FD	PLL	IS-L	Bi-BFS	HL	FD	PLL	IS-L
Skitter	1.7M	11M	**2**	**13**	30	638	1042	**0.067**	0.043	0.008	3.556	3.504	**42 MB**	202 MB	2.5 GB	507 MB
Flickr	1.7M	16M	**2**	**14**	41	1330	8359	**0.015**	0.028	0.01	33.760	4.155	**34 MB**	178 MB	3.7 GB	679 MB
Hollywood	1.1M	114M	**3**	**17**	107	31855	DNF	**0.047**	0.075	0.051	-	6.956	**28 MB**	293 MB	13 GB	-
Orkut	3.1M	117M	**10**	**62**	366	DNF	DNF	**0.224**	0.251	-	-	21.086	**70 MB**	756 MB	-	-
Enwiki	4.3M	101M	**9**	**77**	308	22080	DNF	**0.190**	0.131	0.027	-	19.423	**83 MB**	743 MB	12 GB	-
Livejournal	4.8M	69M	**9**	**77**	166	DNF	20583	**0.088**	0.111	-	56.847	17.264	**123 MB**	778 MB	-	3.8 GB
Indochina	7.4M	194M	**8**	**50**	144	9456	DNF	**1.905**	1.803	0.02	-	9.734	**81 MB**	999 MB	21 GB	-
IT	41M	1.2B	**66**	**304**	1623	DNF	DNF	**2.684**	2.118	-	-	92.187	**855 MB**	5.6 GB	-	-
Twitter	42M	1.5B	**133**	**1380**	1838	DNF	DNF	**1.424**	0.432	-	-	426.949	**1.2 GB**	4.8 GB	-	-
Friendster	66M	1.8B	**135**	**2229**	9661	DNF	DNF	**1.091**	1.435	-	-	534.576	**2.5 GB**	11.8 GB	-	-
UK	106M	3.7B	**110**	**1124**	6201	DNF	DNF	**11.84**	18.979	-	-	355.688	**1.8 GB**	14.1 GB	-	-
Clueweb09	1.7B	7.8B	**4236**	**28124**	DNF	DNF	DNF	**0.309**	-	-	-	-	**4.7 GB**	-	-	-

Construction Time

- Much faster (up to 70 times faster)
- Scalable (scale to graphs with billions of vertices and edges)

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Construction Time [sec]					Query Time [ms]					Labelling Size			
			HL-P	HL	FD	PLL	IS-L	HL	FD	PLL	IS-L	Bi-BFS	HL	FD	PLL	IS-L
Skitter	1.7M	11M	**2**	**13**	30	638	1042	**0.067**	0.043	0.008	3.556	3.504	**42 MB**	202 MB	2.5 GB	507 MB
Flickr	1.7M	16M	**2**	**14**	41	1330	8359	**0.015**	0.028	0.01	33.760	4.155	**34 MB**	178 MB	3.7 GB	679 MB
Hollywood	1.1M	114M	**3**	**17**	107	31855	DNF	**0.047**	0.075	0.051	-	6.956	**28 MB**	293 MB	13 GB	-
Orkut	3.1M	117M	**10**	**62**	366	DNF	DNF	**0.224**	0.251	-	-	21.086	**70 MB**	756 MB	-	-
Enwiki	4.3M	101M	**9**	**77**	308	22080	DNF	**0.190**	0.131	0.027	-	19.423	**83 MB**	743 MB	12 GB	-
Livejournal	4.8M	69M	**9**	**77**	166	DNF	20583	**0.088**	0.111	-	56.847	17.264	**123 MB**	778 MB	-	3.8 GB
Indochina	7.4M	194M	**8**	**50**	144	9456	DNF	**1.905**	1.803	0.02	-	9.734	**81 MB**	999 MB	21 GB	-
IT	41M	1.2B	**66**	**304**	1623	DNF	DNF	**2.684**	2.118	-	-	92.187	**855 MB**	5.6 GB	-	-
Twitter	42M	1.5B	**133**	**1380**	1838	DNF	DNF	**1.424**	0.432	-	-	426.949	**1.2 GB**	4.8 GB	-	-
Friendster	66M	1.8B	**135**	**2229**	9661	DNF	DNF	**1.091**	1.435	-	-	534.576	**2.5 GB**	11.8 GB	-	-
UK	106M	3.7B	**110**	**1124**	6201	DNF	DNF	**11.84**	18.979	-	-	355.688	**1.8 GB**	14.1 GB	-	-
Clueweb09	1.7B	7.8B	**4236**	**28124**	DNF	DNF	DNF	**0.309**	-	-	-	-	**4.7 GB**	-	-	-

Construction Time

- Much faster (up to 70 times faster)
- Scalable (scale to graphs with billions of vertices and edges)

Query Time

- Comparable (queries within 1ms on a graph with 8B edges)

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Construction Time [sec]					Query Time [ms]					Labelling Size			
			HL-P	HL	FD	PLL	IS-L	HL	FD	PLL	IS-L	Bi-BFS	HL	FD	PLL	IS-L
Skitter	1.7M	11M	**2**	**13**	30	638	1042	**0.067**	0.043	0.008	3.556	3.504	**42 MB**	202 MB	2.5 GB	507 MB
Flickr	1.7M	16M	**2**	**14**	41	1330	8359	**0.015**	0.028	0.01	33.760	4.155	**34 MB**	178 MB	3.7 GB	679 MB
Hollywood	1.1M	114M	**3**	**17**	107	31855	DNF	**0.047**	0.075	0.051	-	6.956	**28 MB**	293 MB	13 GB	-
Orkut	3.1M	117M	**10**	**62**	366	DNF	DNF	**0.224**	0.251	-	-	21.086	**70 MB**	756 MB	-	-
Enwiki	4.3M	101M	**9**	**77**	308	22080	DNF	**0.190**	0.131	0.027	-	19.423	**83 MB**	743 MB	12 GB	-
Livejournal	4.8M	69M	**9**	**77**	166	DNF	20583	**0.088**	0.111	-	56.847	17.264	**123 MB**	778 MB	-	3.8 GB
Indochina	7.4M	194M	**8**	**50**	144	9456	DNF	**1.905**	1.803	0.02	-	9.734	**81 MB**	999 MB	21 GB	-
IT	41M	1.2B	**66**	**304**	1623	DNF	DNF	**2.684**	2.118	-	-	92.187	**855 MB**	5.6 GB	-	-
Twitter	42M	1.5B	**133**	**1380**	1838	DNF	DNF	**1.424**	0.432	-	-	426.949	**1.2 GB**	4.8 GB	-	-
Friendster	66M	1.8B	**135**	**2229**	9661	DNF	DNF	**1.091**	1.435	-	-	534.576	**2.5 GB**	11.8 GB	-	-
UK	106M	3.7B	**110**	**1124**	6201	DNF	DNF	**11.84**	18.979	-	-	355.688	**1.8 GB**	14.1 GB	-	-
Clueweb09	1.7B	7.8B	**4236**	**28124**	DNF	DNF	DNF	**0.309**	-	-	-	-	**4.7 GB**	-	-	-

Construction Time

- Much faster (up to 70 times faster)
- Scalable (scale to graphs with billions of vertices and edges)

Query Time

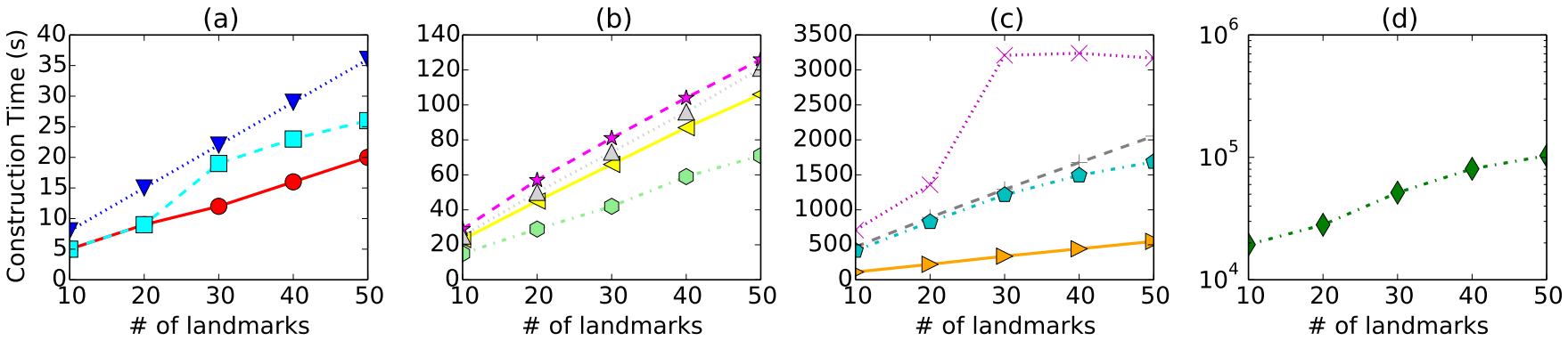
- Comparable (queries within 1ms on a graph with 8B edges)

Labelling Size

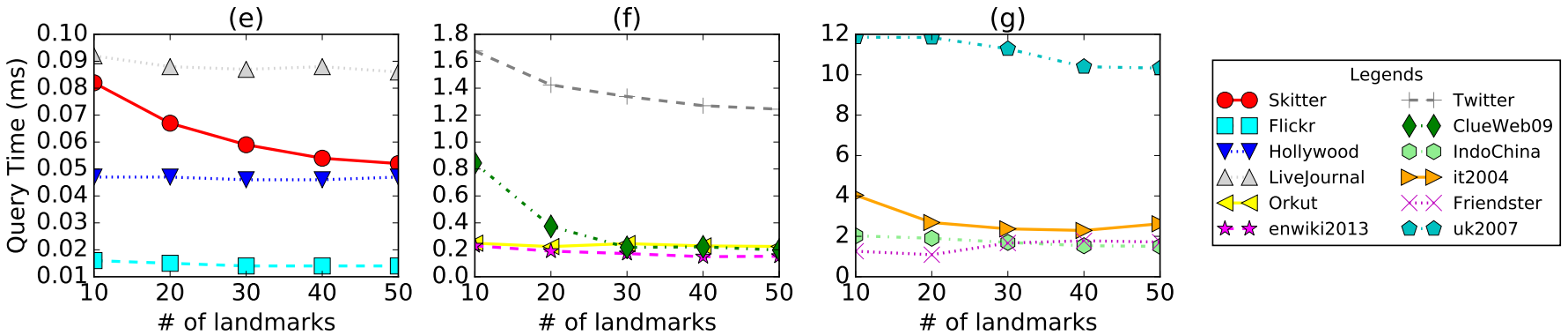
- Much smaller (up to 90% smaller)

Empirical Evaluation

- Construction time of our method HL under varying landmarks



- Query time of our method HL under varying landmarks

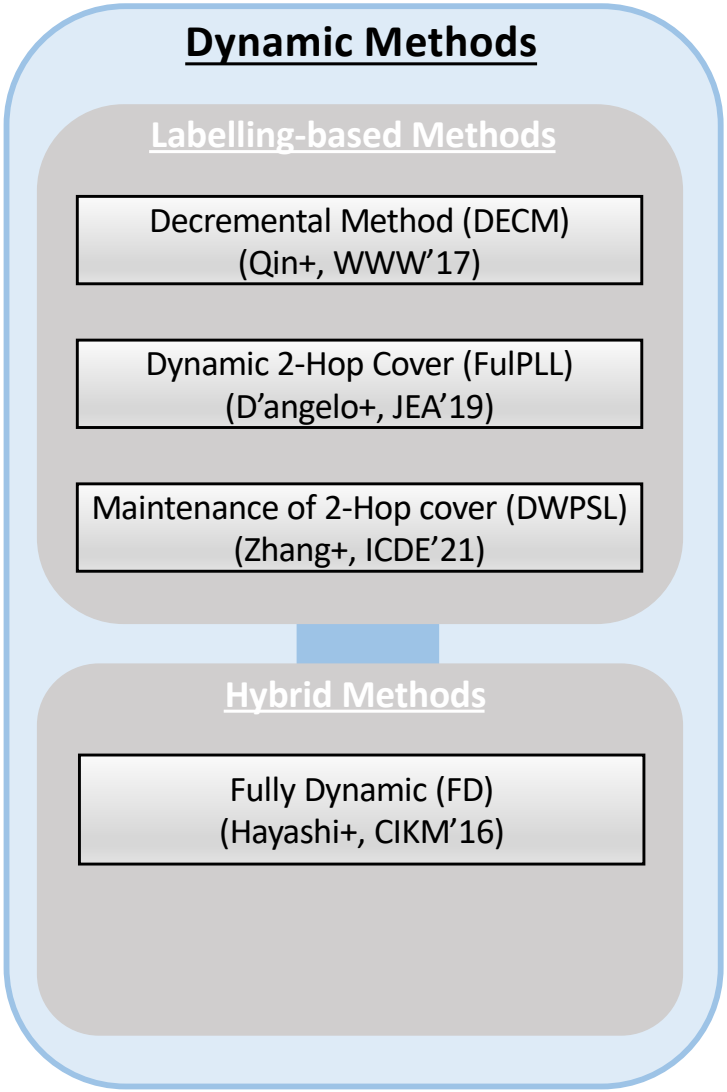


THE AUSTRALIAN NATIONAL UNIVERSITY

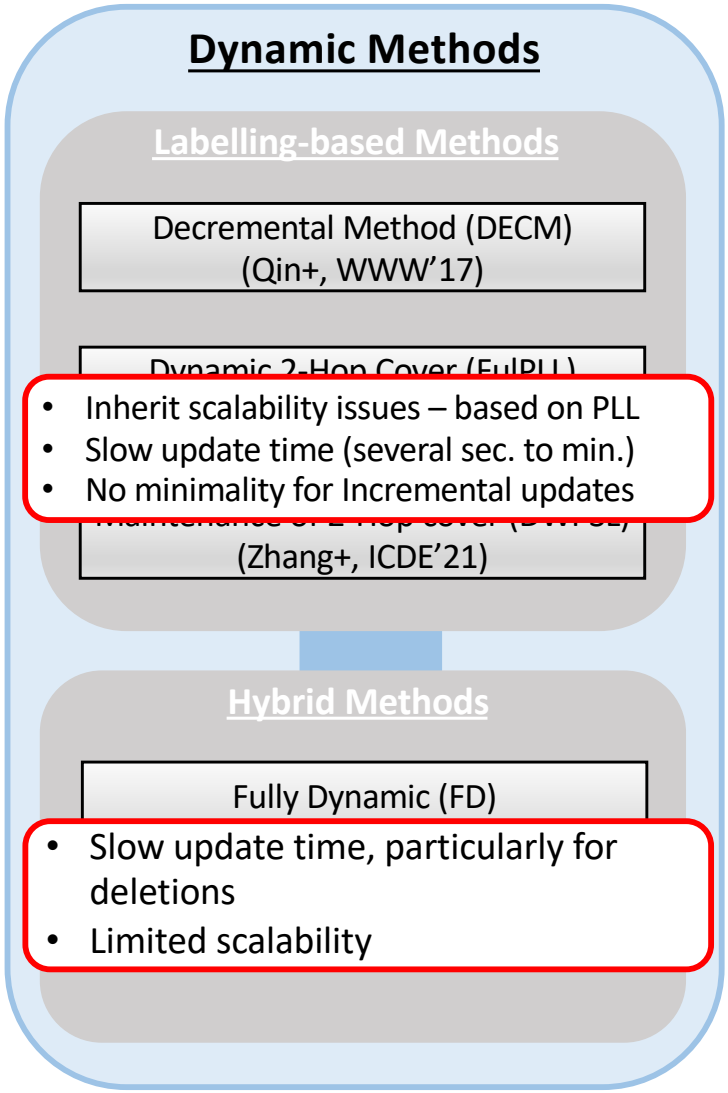
Answering Exact Shortest Path Distance Queries in Dynamic Networks [2]

[2] **Farhan, M.**, Wang, Q., Lin, Y., Mckay, B. "Fast Fully Dynamic Labelling for Distance Queries." The International Journal on Very Large Data Bases (The VLDB Journal).

Existing Methods



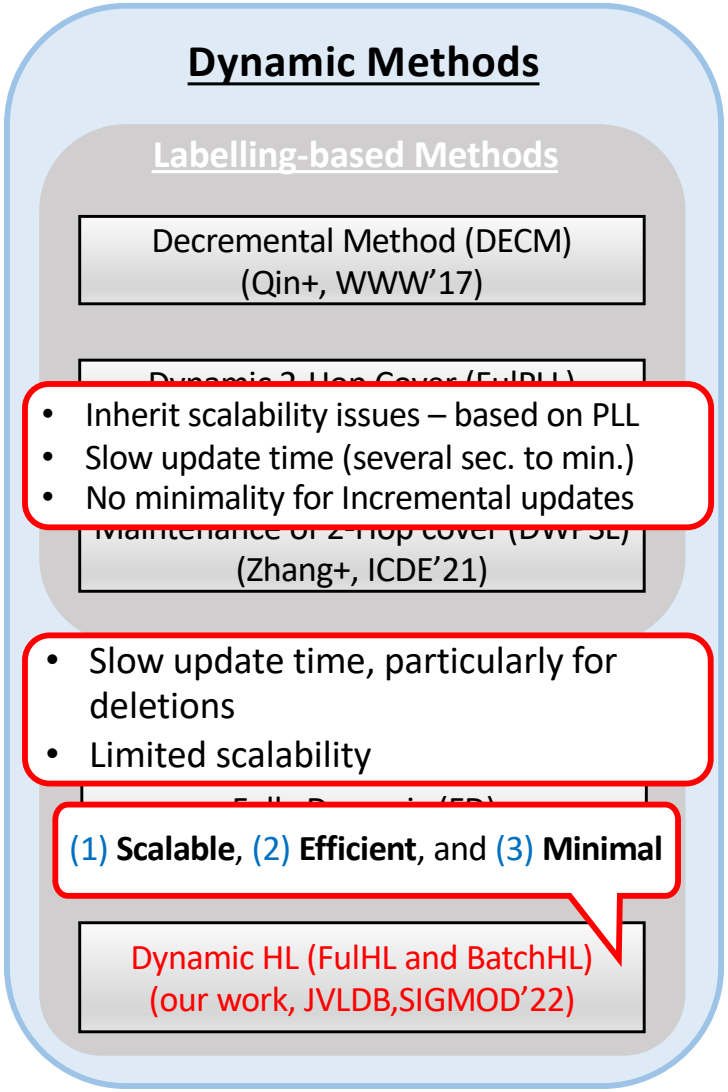
Existing Methods



- Inherit scalability issues – based on PLL
- Slow update time (several sec. to min.)
- No minimality for Incremental updates

- Slow update time, particularly for deletions
- Limited scalability

Existing Methods

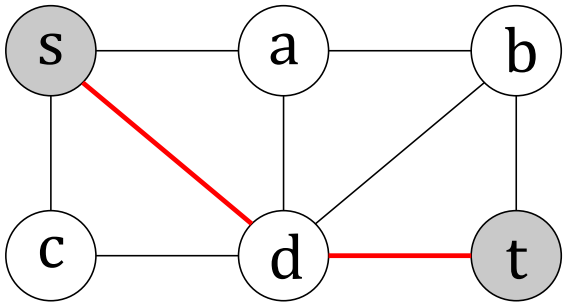


- **Problem:**

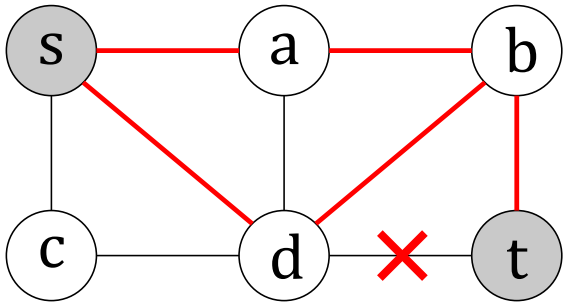
  Let $G \hookrightarrow G'$ denote that G is changed to G' by an edge insertion or deletion, to answer the shortest path distance $d_{G'}(s, t)$.

- **Problem:**

  Let $G \hookrightarrow G'$ denote that G is changed to G' by an edge insertion or deletion, to answer the shortest path distance $d_{G'}(s, t)$.



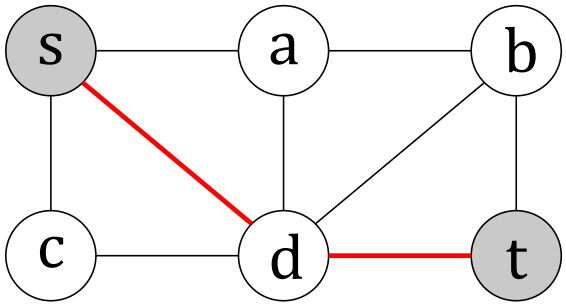
$d_G(s, t) = 2$



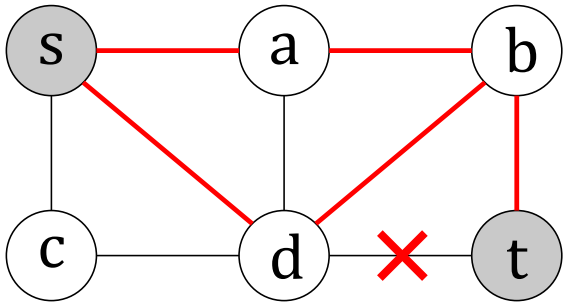
$d_{G'}(s, t) = 3$

- **Problem:**

  Let $G \hookrightarrow G'$ denote that G is changed to G' by an edge insertion or deletion, to answer the shortest path distance $d_{G'}(s,t)$.



$d_G(s,t) = 2$



$d_{G'}(s,t) = 3$

(1) Computing an upper bound $d_{st}^{\top}$ of $d_G(s,t)$ using the highway cover distance labelling;

(2) Computing $d_G(s,t)$ using a distance-bounded shortest-path search over a sparsified graph $G[V \setminus R]$.

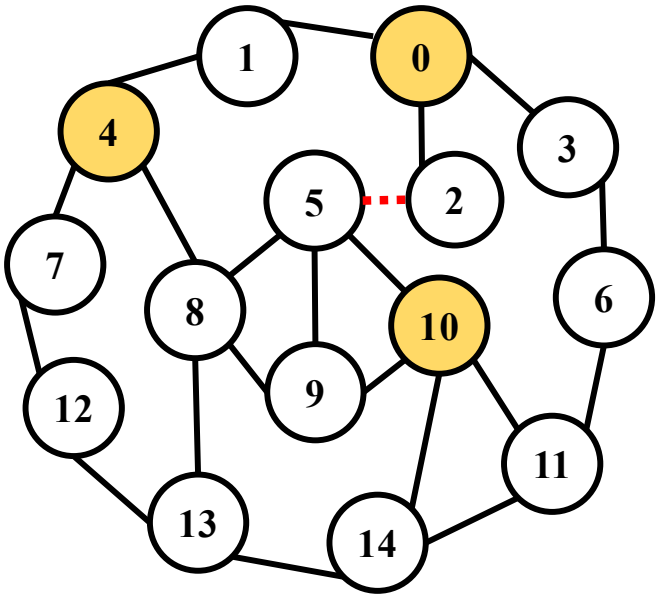
- Incorrect upper bounds
- Overestimated or underestimated distances

- **Challenge:**

  How to efficiently maintain a highway cover labelling for a very large dynamic network?

- **Challenge:**

  How to efficiently maintain a highway cover labelling for a very large dynamic network?



Example Graph

	0	4	10
0	0	2	4
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(0, 4) (10, 1)

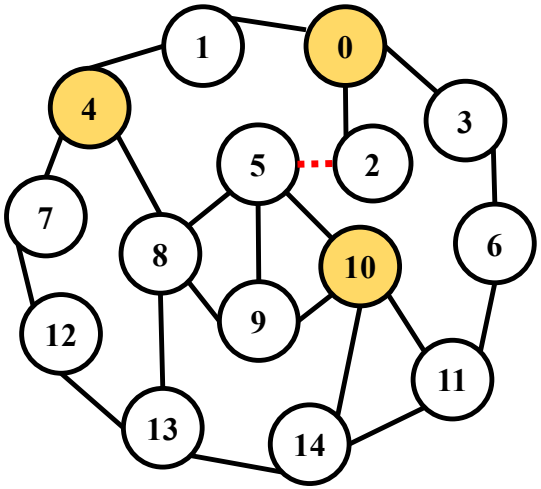
Highway Labelling

Fully Dynamic Framework

- *Incremental algorithm* - to efficiently reflect incremental updates (edge additions) into highway labelling
- *Decremental algorithm* - to efficiently reflect decremental updates (edge deletions) into highway labelling

Incremental Algorithm (IncHL)

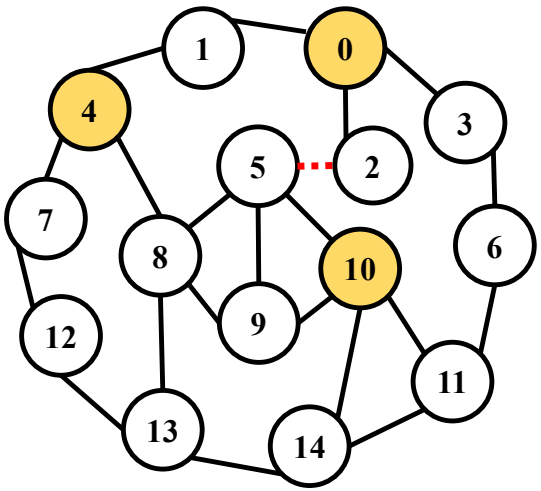
- Performs a Jumped-and-Pruned search from an anchor



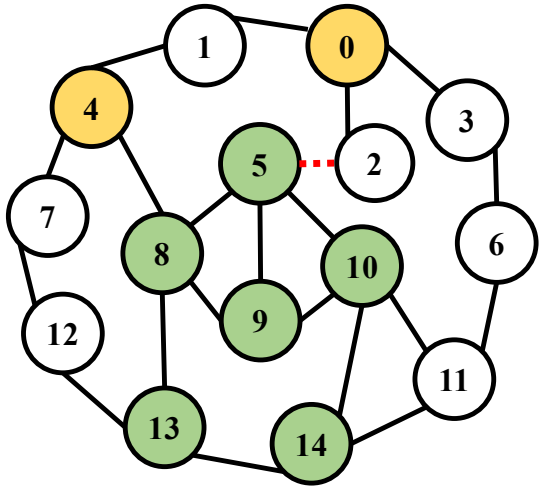
Example Graph

Incremental Algorithm (IncHL)

- Performs a Jumped-and-Pruned search from an anchor



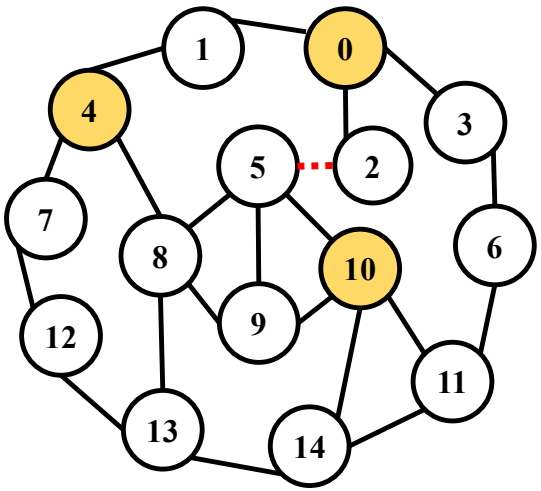
Example Graph



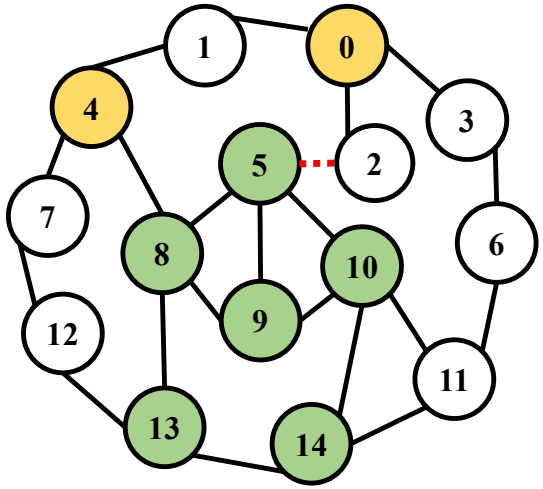
Affected vertices

Incremental Algorithm (IncHL)

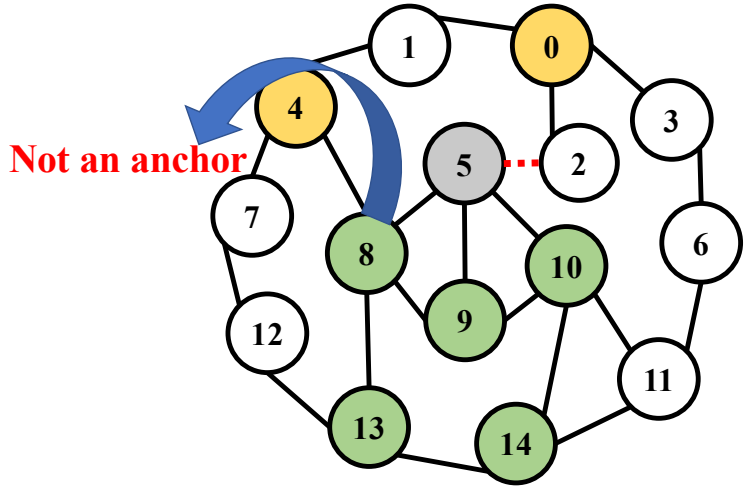
- Performs a Jumped-and-Pruned search from an anchor



Example Graph

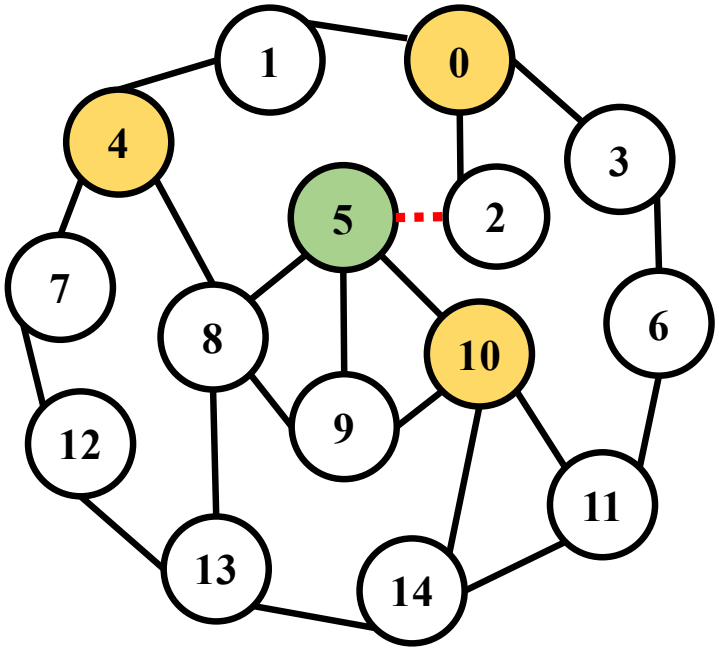


Affected vertices



Anchor vertices

Incremental Algorithm (IncHL)



Example Graph

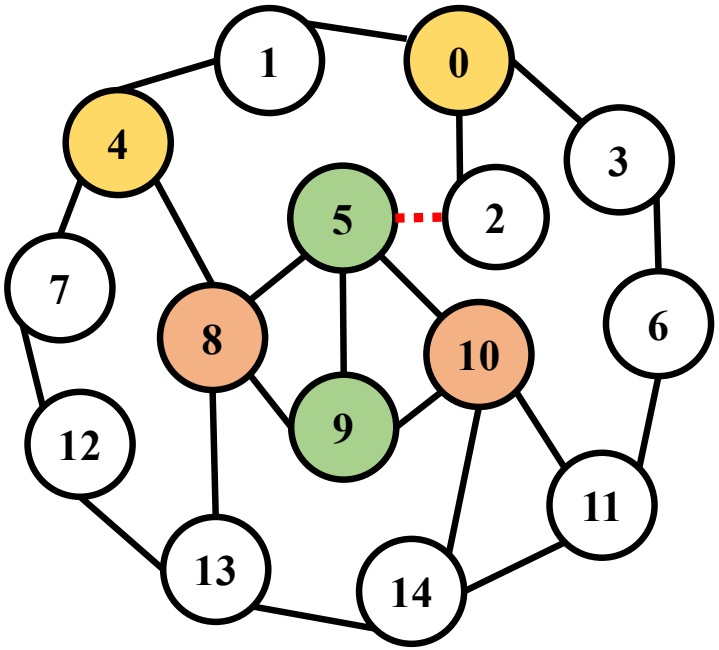
	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(0, 4) (10, 1)

Highway Labelling

(5 is the only anchor vertex after an edge insertion $(2, 5)$)

Incremental Algorithm (IncHL)



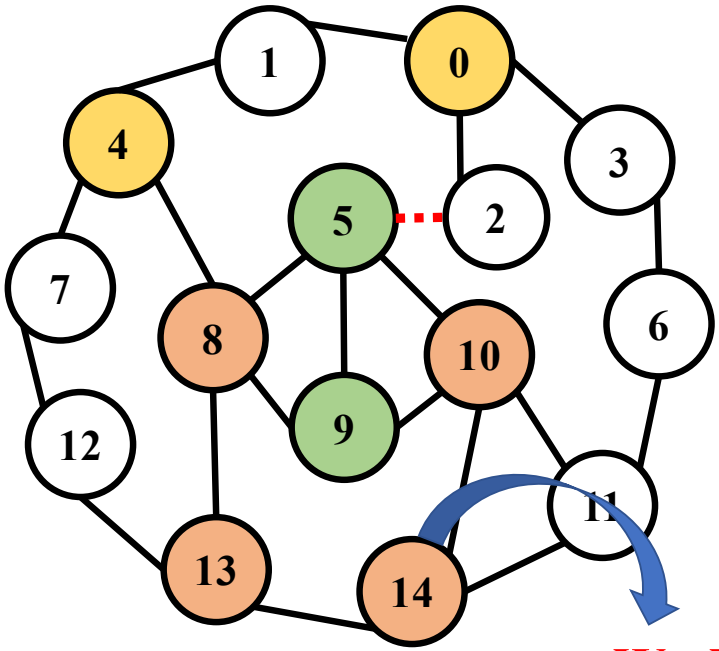
Example Graph

	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(0, 4) (10, 1)

Highway Labelling

Incremental Algorithm (IncHL)



Example Graph

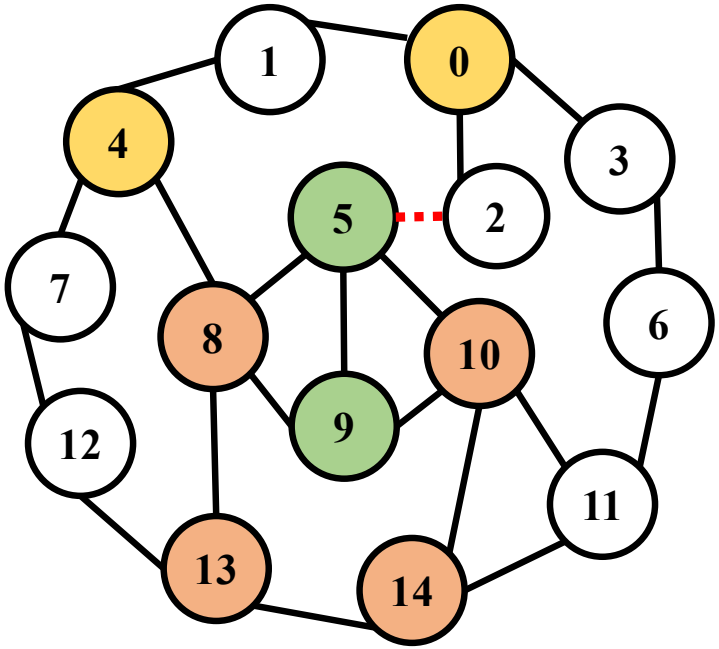
	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(0, 4) (10, 1)

Weekly prunable!

Highway Labelling

Incremental Algorithm (IncHL)



Example Graph

	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	, 2) (10, 3)
L(14)	(0, 4) (10, 1)

Redundant entry!

Highway Labelling

- **Minimal Labelling:** remove the pruning condition

Decremental Algorithm (DecHL)

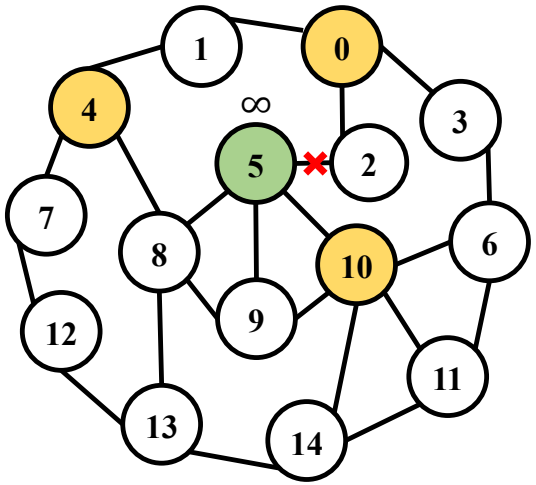
- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge

Decremental Algorithm (DecHL)

- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge
- **First JP-BFS** - to identify affected vertices along with their contingent distances w.r.t. a landmark r

Decremental Algorithm (DecHL)

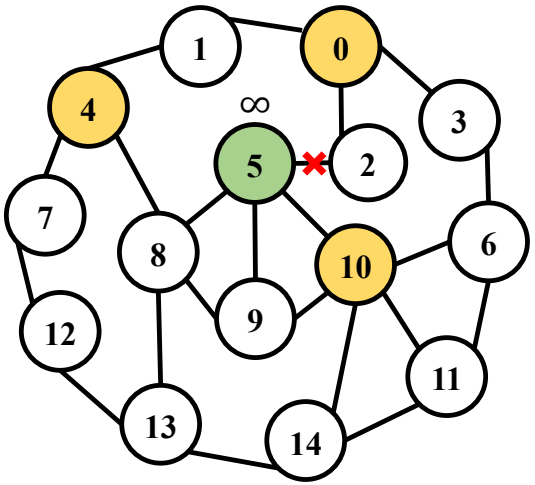
- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge
- **First JP-BFS** - to identify affected vertices along with their contingent distances w.r.t. a landmark r



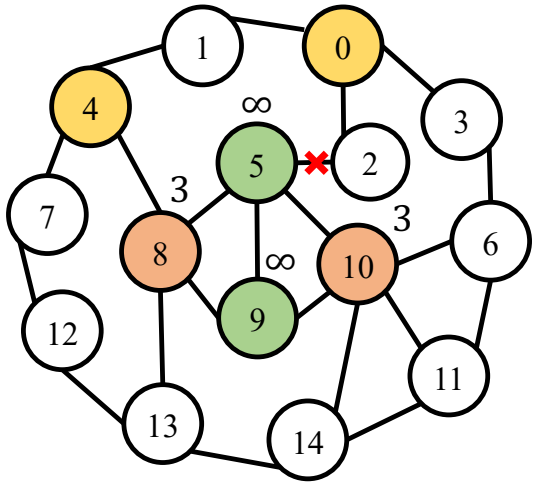
Jumping

Decremental Algorithm (DecHL)

- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge
- **First JP-BFS** - to identify affected vertices along with their contingent distances w.r.t. a landmark r



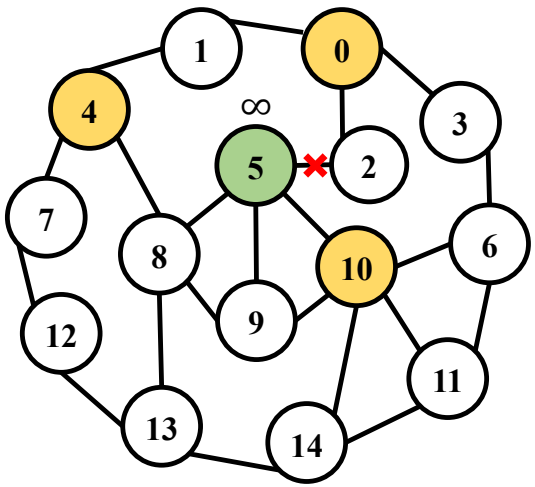
Jumping



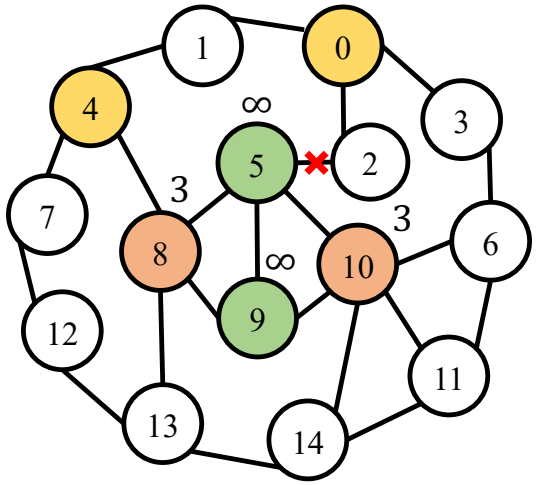
2nd Iteration

Decremental Algorithm (DecHL)

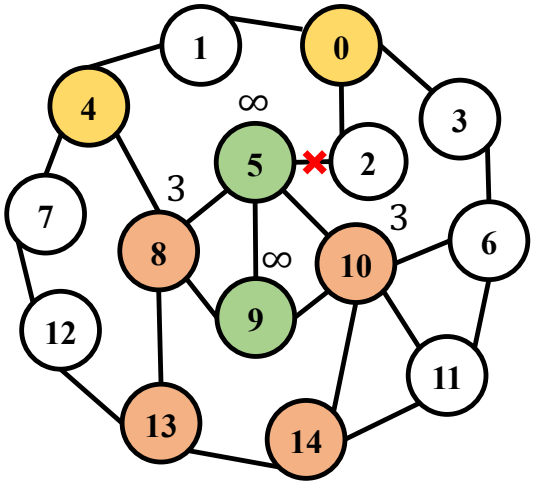
- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge
- **First JP-BFS** - to identify affected vertices along with their contingent distances w.r.t. a landmark r



Jumping



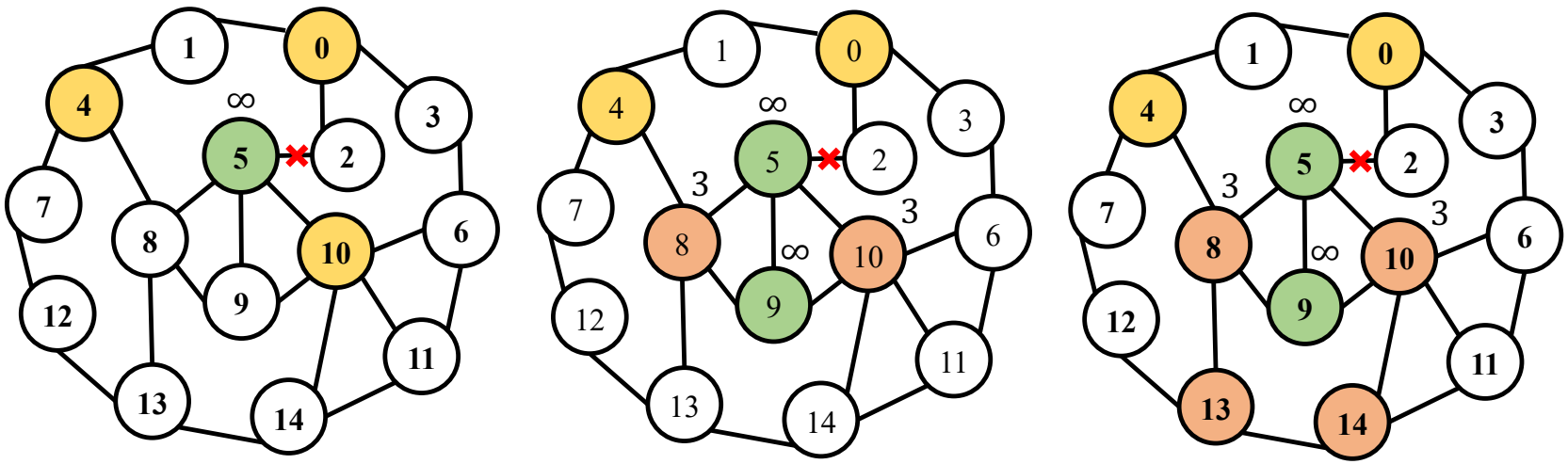
2nd Iteration



3rd iteration

Decremental Algorithm (DecHL)

- Performs two Jumped-and-Pruned (JP-BFS) searches
  - multiple anchor vertices may exist
  - can be far away from the deleted edge
- **First JP-BFS** - to identify affected vertices along with their contingent distances w.r.t. a landmark r



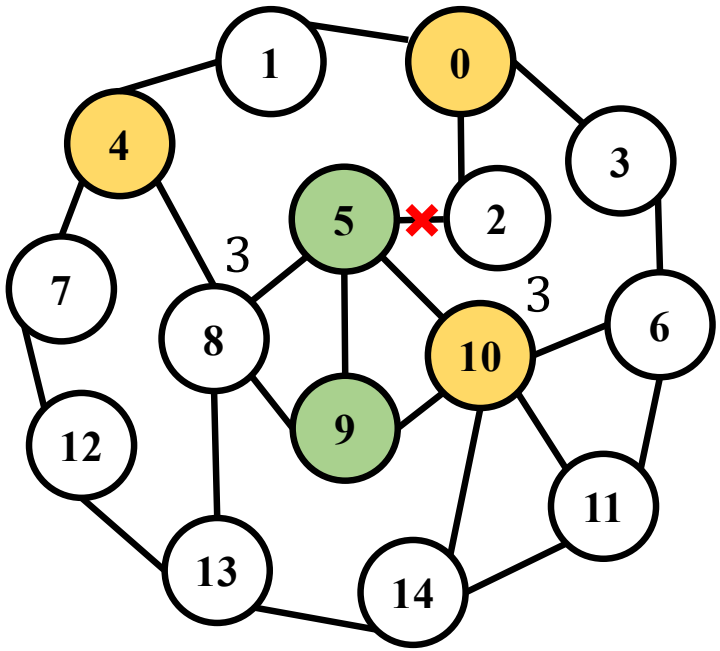
Jumping | 2nd Iteration | 3rd iteration

($\{5, 8, 9, 10\}$ are affected vertices and $\{8, 10\}$ are anchor vertices.)

Decremental Algorithm (DecHL)

- **Second JP-BFS** - to update the labels of affected vertices w.r.t. a landmark r

- **Second JP-BFS** - to update the labels of affected vertices w.r.t. a landmark r



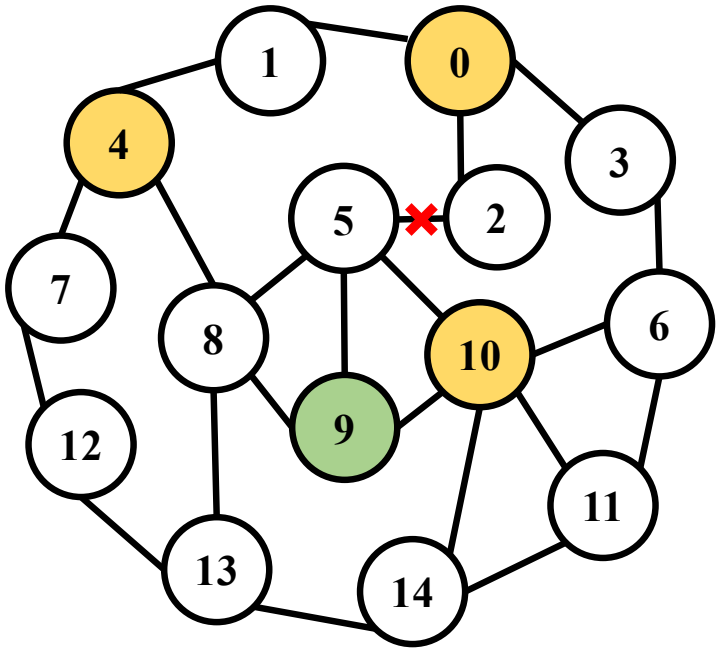
Example Graph

	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(10, 1)

Highway Labelling

- **Second JP-BFS** - to update the labels of affected vertices w.r.t. a landmark r



Example Graph

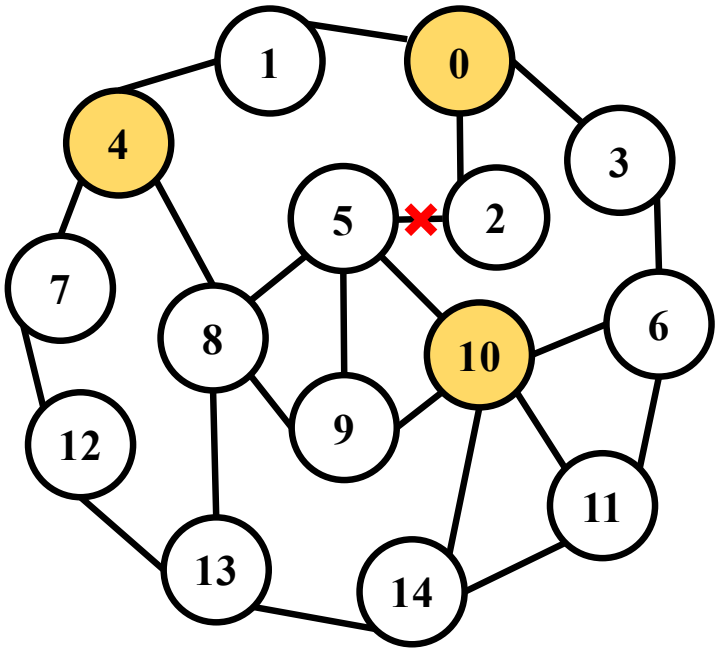
	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(0, 2) (4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(10, 1)

Highway Labelling

Decremental Algorithm (DecHL)

- **Second JP-BFS** - to update the labels of affected vertices w.r.t. a landmark r



Example Graph

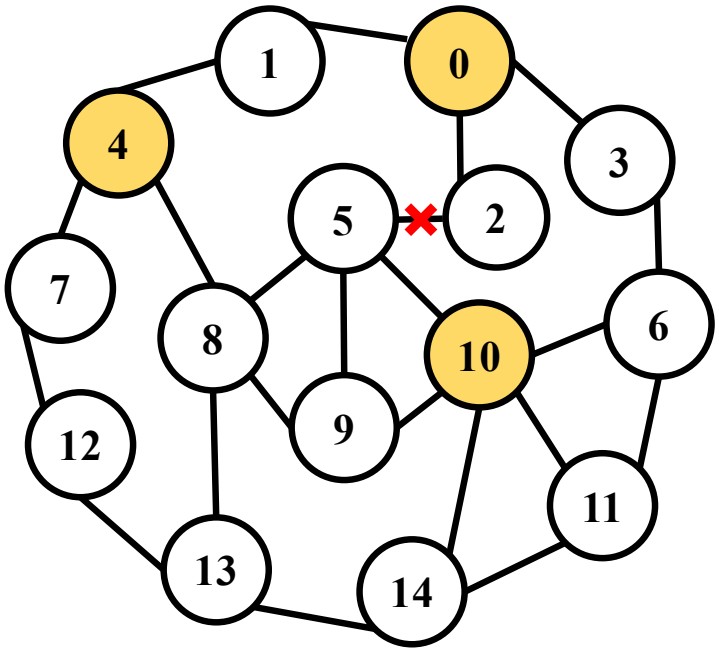
	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(0, 3) (4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(10, 1)

Highway Labelling

Decremental Algorithm (DecHL)

- **Second JP-BFS** - to update the labels of affected vertices w.r.t. a landmark r



Example Graph

	0	4	10
0	0	2	3
4	2	0	3
10	4	3	0

Label	Distance Entries
L(0)	(0, 0)
...	...
L(4)	(4, 0)
L(5)	(4, 2) (10, 1)
...	...
L(8)	(4, 1) (10, 2)
L(9)	(4, 3) (10, 2)
L(10)	(10, 0)
...	...
L(13)	(4, 2) (10, 3)
L(14)	(10, 1)

Highway Labelling

Empirical Evaluation

THE AUSTRALIAN NATIONAL UNIVERSITY

Dataset	$\|V\|$	$\|E\|$	Update Time (ms)				Labelling Size			Query Time (ms)		
			FulHL-M	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL
Skitter	1.7M	11M	**1.096**	**1.019**	10.67	20400	**42 MB**	153 MB	11.6 GB	**0.027**	0.019	0.006
Flickr	1.7M	16M	**0.055**	**0.053**	7.655	6810	**34 MB**	152 MB	12.7 GB	**0.007**	0.012	0.009
Hollywood	1.1M	114M	**0.223**	**0.212**	10.54	-	**27 MB**	262 MB	-	**0.027**	0.037	-
Orkut	3.1M	117M	**1.234**	**1.075**	40.12	-	**70 MB**	711 MB	-	**0.101**	0.103	-
Enwiki	4.2M	101M	**1.488**	**1.459**	88.54	-	**82 MB**	608 MB	-	**0.054**	0.035	-
Livejournal	4.8M	69M	**0.275**	**0.179**	2.564	-	**122 MB**	662 MB	-	**0.044**	0.046	-
Indochina	7.4M	194M	**1.414**	**0.598**	107.2	-	**87 MB**	840 MB	-	**0.737**	0.839	-
IT	41M	1.2B	**22.9**	**10.62**	160.3	-	**862 MB**	4.74 GB	-	**1.069**	1.013	-
Twitter	42M	1.5B	**73.37**	**72.76**	2512	-	**1.14 GB**	3.83 GB	-	**0.863**	0.177	-
Friendster	66M	1.8B	**2.131**	**2.097**	21.64	-	**2.43 GB**	9.14 GB	-	**0.814**	0.904	-
UK	106M	3.7B	**2.755**	**1.075**	337.6	-	**1.78 GB**	11.8 GB	-	**3.443**	5.858	-
Clueweb09	1.7B	7.8B	**103.1**	**56.25**	-	-	**163 GB**	-	-	**16.93**	-	-
Clueweb12	~1B	43B	**15950**	**1796**	-	-	**49.1 GB**	-	-	**9.375**	-	-

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Update Time (ms)				Labelling Size			Query Time (ms)		
			FulHL-M	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL
Skitter	1.7M	11M	**1.096**	**1.019**	10.67	20400	**42 MB**	153 MB	11.6 GB	**0.027**	0.019	0.006
Flickr	1.7M	16M	**0.055**	**0.053**	7.655	6810	**34 MB**	152 MB	12.7 GB	**0.007**	0.012	0.009
Hollywood	1.1M	114M	**0.223**	**0.212**	10.54	-	**27 MB**	262 MB	-	**0.027**	0.037	-
Orkut	3.1M	117M	**1.234**	**1.075**	40.12	-	**70 MB**	711 MB	-	**0.101**	0.103	-
Enwiki	4.2M	101M	**1.488**	**1.459**	88.54	-	**82 MB**	608 MB	-	**0.054**	0.035	-
Livejournal	4.8M	69M	**0.275**	**0.179**	2.564	-	**122 MB**	662 MB	-	**0.044**	0.046	-
Indochina	7.4M	194M	**1.414**	**0.598**	107.2	-	**87 MB**	840 MB	-	**0.737**	0.839	-
IT	41M	1.2B	**22.9**	**10.62**	160.3	-	**862 MB**	4.74 GB	-	**1.069**	1.013	-
Twitter	42M	1.5B	**73.37**	**72.76**	2512	-	**1.14 GB**	3.83 GB	-	**0.863**	0.177	-
Friendster	66M	1.8B	**2.131**	**2.097**	21.64	-	**2.43 GB**	9.14 GB	-	**0.814**	0.904	-
UK	106M	3.7B	**2.755**	**1.075**	337.6	-	**1.78 GB**	11.8 GB	-	**3.443**	5.858	-
Clueweb09	1.7B	7.8B	**103.1**	**56.25**	-	-	**163 GB**	-	-	**16.93**	-	-
Clueweb12	~1B	43B	**15950**	**1796**	-	-	**49.1 GB**	-	-	**9.375**	-	-

- Significantly faster than FulFD and several orders of magnitude faster than FulPLL.

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Update Time (ms)				Labelling Size			Query Time (ms)		
			FulHL-M	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL
Skitter	1.7M	11M	**1.096**	**1.019**	10.67	20400	**42 MB**	153 MB	11.6 GB	**0.027**	0.019	0.006
Flickr	1.7M	16M	**0.055**	**0.053**	7.655	6810	**34 MB**	152 MB	12.7 GB	**0.007**	0.012	0.009
Hollywood	1.1M	114M	**0.223**	**0.212**	10.54	-	**27 MB**	262 MB	-	**0.027**	0.037	-
Orkut	3.1M	117M	**1.234**	**1.075**	40.12	-	**70 MB**	711 MB	-	**0.101**	0.103	-
Enwiki	4.2M	101M	**1.488**	**1.459**	88.54	-	**82 MB**	608 MB	-	**0.054**	0.035	-
Livejournal	4.8M	69M	**0.275**	**0.179**	2.564	-	**122 MB**	662 MB	-	**0.044**	0.046	-
Indochina	7.4M	194M	**1.414**	**0.598**	107.2	-	**87 MB**	840 MB	-	**0.737**	0.839	-
IT	41M	1.2B	**22.9**	**10.62**	160.3	-	**862 MB**	4.74 GB	-	**1.069**	1.013	-
Twitter	42M	1.5B	**73.37**	**72.76**	2512	-	**1.14 GB**	3.83 GB	-	**0.863**	0.177	-
Friendster	66M	1.8B	**2.131**	**2.097**	21.64	-	**2.43 GB**	9.14 GB	-	**0.814**	0.904	-
UK	106M	3.7B	**2.755**	**1.075**	337.6	-	**1.78 GB**	11.8 GB	-	**3.443**	5.858	-
Clueweb09	1.7B	7.8B	**103.1**	**56.25**	-	-	**163 GB**	-	-	**16.93**	-	-
Clueweb12	~1B	43B	**15950**	**1796**	-	-	**49.1 GB**	-	-	**9.375**	-	-

- Significantly faster than FulFD and several orders of magnitude faster than FulPLL.
- Much smaller (save up to 90% of space)

Empirical Evaluation

THE AUSTRALIAN NATIONAL UNIVERSITY

Dataset	$\|V\|$	$\|E\|$	Update Time (ms)				Labelling Size			Query Time (ms)		
			FulHL-M	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL	FulHL	FulFD	FulPLL
Skitter	1.7M	11M	**1.096**	**1.019**	10.67	20400	**42 MB**	153 MB	11.6 GB	**0.027**	0.019	0.006
Flickr	1.7M	16M	**0.055**	**0.053**	7.655	6810	**34 MB**	152 MB	12.7 GB	**0.007**	0.012	0.009
Hollywood	1.1M	114M	**0.223**	**0.212**	10.54	-	**27 MB**	262 MB	-	**0.027**	0.037	-
Orkut	3.1M	117M	**1.234**	**1.075**	40.12	-	**70 MB**	711 MB	-	**0.101**	0.103	-
Enwiki	4.2M	101M	**1.488**	**1.459**	88.54	-	**82 MB**	608 MB	-	**0.054**	0.035	-
Livejournal	4.8M	69M	**0.275**	**0.179**	2.564	-	**122 MB**	662 MB	-	**0.044**	0.046	-
Indochina	7.4M	194M	**1.414**	**0.598**	107.2	-	**87 MB**	840 MB	-	**0.737**	0.839	-
IT	41M	1.2B	**22.9**	**10.62**	160.3	-	**862 MB**	4.74 GB	-	**1.069**	1.013	-
Twitter	42M	1.5B	**73.37**	**72.76**	2512	-	**1.14 GB**	3.83 GB	-	**0.863**	0.177	-
Friendster	66M	1.8B	**2.131**	**2.097**	21.64	-	**2.43 GB**	9.14 GB	-	**0.814**	0.904	-
UK	106M	3.7B	**2.755**	**1.075**	337.6	-	**1.78 GB**	11.8 GB	-	**3.443**	5.858	-
Clueweb09	1.7B	7.8B	**103.1**	**56.25**	-	-	**163 GB**	-	-	**16.93**	-	-
Clueweb12	~1B	43B	**15950**	**1796**	-	-	**49.1 GB**	-	-	**9.375**	-	-

- Significantly faster than FulFD and several orders of magnitude faster than FulPLL.
- Much smaller (save up to 90% of space)
- Comparable (answer queries in less than 10ms on a graph with 43B edges)

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Incremental Algorithms					Decremental Algorithms		
			IncHL-M (ms)	IncHL (ms)	IncHL$^+$ (ms)	IncFD (ms)	IncPLL (ms)	DecHL (ms)	DecFD (ms)	DecPLL (sec.)
Skitter	1.7M	11M	**0.133**	**0.075**	0.194	0.447	2.189	**1.443**	19.48	21.3
Flickr	1.7M	16M	**0.005**	**0.005**	0.006	0.046	1.869	**0.152**	17.71	11.7
Hollywood	1.1M	114M	**0.027**	**0.026**	0.031	0.078	48.97	**0.265**	21.03	-
Orkut	3.1M	117M	**1.687**	**1.423**	2.026	2.039	-	**0.418**	48.12	-
Enwiki	4.2M	101M	**0.119**	**0.105**	0.134	0.129	6.596	**2.969**	163.8	-
Livejournal	4.8M	69M	**0.201**	**0.122**	0.245	0.225	-	**0.300**	7.406	-
Indochina	7.4M	194M	**2.587**	**1.187**	5.443	167.7	2021	**0.233**	60.60	-
IT	41M	1.2B	**49.77**	**21.34**	95.92	241.8	-	**5.843**	210.5	-
Twitter	42M	1.5B	**0.017**	**0.015**	0.027	0.106	-	**192.6**	5126	-
Friendster	66M	1.8B	**0.119**	**0.119**	0.159	0.396	-	**2.409**	42.92	-
UK	106M	3.7B	**4.071**	**2.132**	11.49	397.7	-	**0.267**	151.5	-
Clueweb09	1.7B	7.8B	**27.04**	**9.205**	40.68	-	-	**131.8**	-	-
Clueweb12	~1B	43B	**26365**	**2061**	61661	-	-	**2129**	-	-

Incremental Updates

- significant improvement Indochina, IT, Clueweb09 and Clueweb12

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Incremental Algorithms					Decremental Algorithms		
			IncHL-M (ms)	IncHL (ms)	IncHL$^+$ (ms)	IncFD (ms)	IncPLL (ms)	DecHL (ms)	DecFD (ms)	DecPLL (sec.)
Skitter	1.7M	11M	**0.133**	**0.075**	0.194	0.447	2.189	**1.443**	19.48	21.3
Flickr	1.7M	16M	**0.005**	**0.005**	0.006	0.046	1.869	**0.152**	17.71	11.7
Hollywood	1.1M	114M	**0.027**	**0.026**	0.031	0.078	48.97	**0.265**	21.03	-
Orkut	3.1M	117M	**1.687**	**1.423**	2.026	2.039	-	**0.418**	48.12	-
Enwiki	4.2M	101M	**0.119**	**0.105**	0.134	0.129	6.596	**2.969**	163.8	-
Livejournal	4.8M	69M	**0.201**	**0.122**	0.245	0.225	-	**0.300**	7.406	-
Indochina	7.4M	194M	**2.587**	**1.187**	5.443	167.7	2021	**0.233**	60.60	-
IT	41M	1.2B	**49.77**	**21.34**	95.92	241.8	-	**5.843**	210.5	-
Twitter	42M	1.5B	**0.017**	**0.015**	0.027	0.106	-	**192.6**	5126	-
Friendster	66M	1.8B	**0.119**	**0.119**	0.159	0.396	-	**2.409**	42.92	-
UK	106M	3.7B	**4.071**	**2.132**	11.49	397.7	-	**0.267**	151.5	-
Clueweb09	1.7B	7.8B	**27.04**	**9.205**	40.68	-	-	**131.8**	-	-
Clueweb12	~1B	43B	**26365**	**2061**	61661	-	-	**2129**	-	-

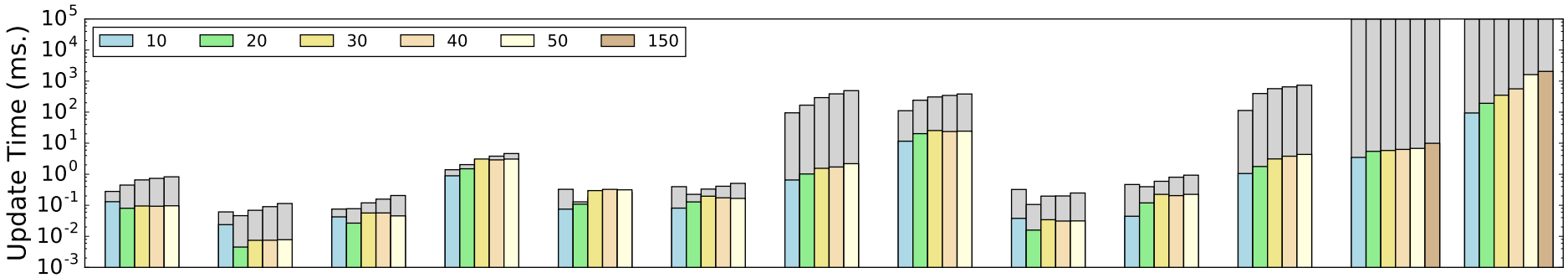
Incremental Updates

- significant improvement Indochina, IT, Clueweb09 and Clueweb12

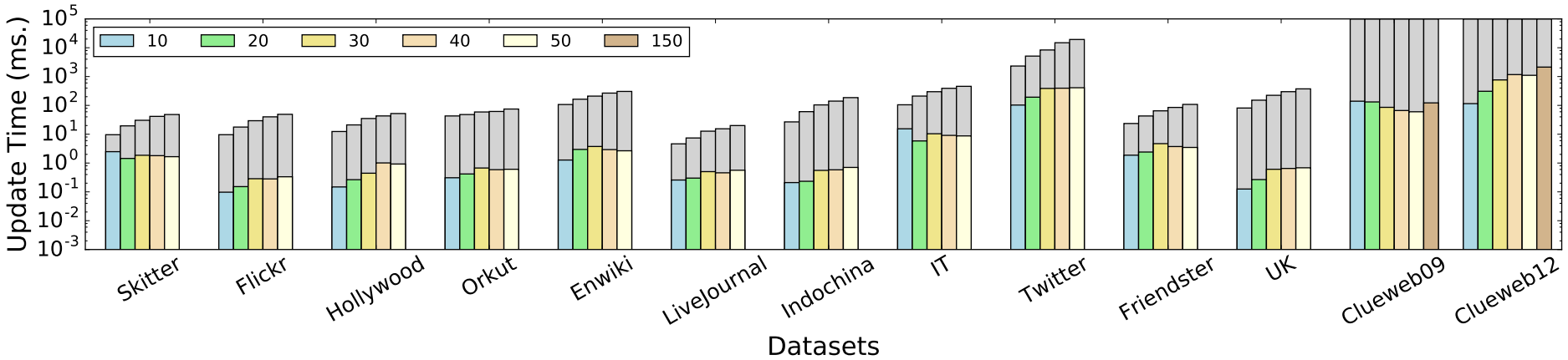
Decremental Updates

- significantly faster overall

Empirical Evaluation



Average update time for incremental updates where colored bars represent IncHL and colored plus grey bars represent IncFD.

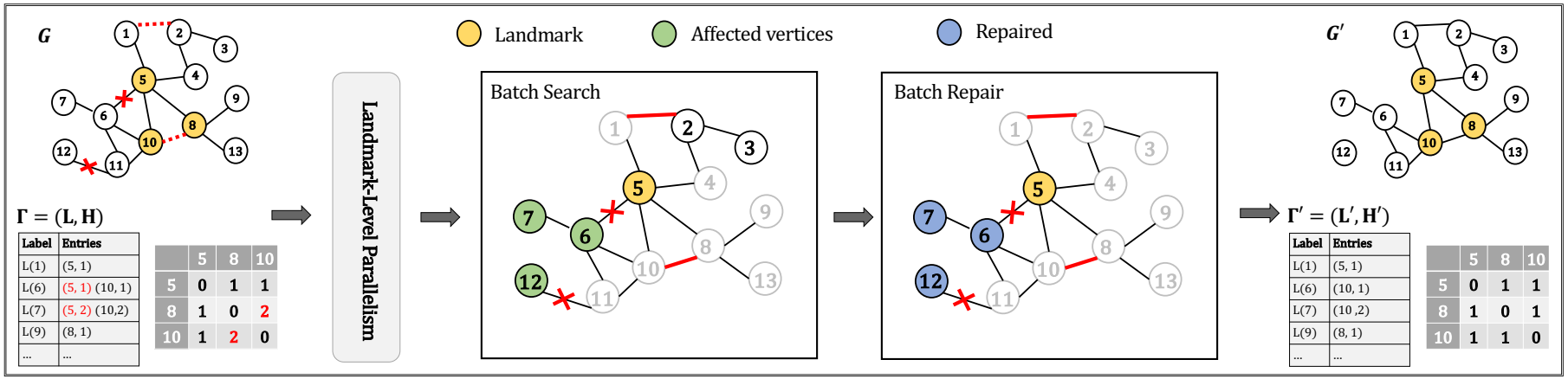


Average update time for decremental updates where colored bars represent DecHL and colored plus grey bars represent DecFD.

Answering Exact Shortest Path Distance Queries in Batch-Dynamic Networks [3]

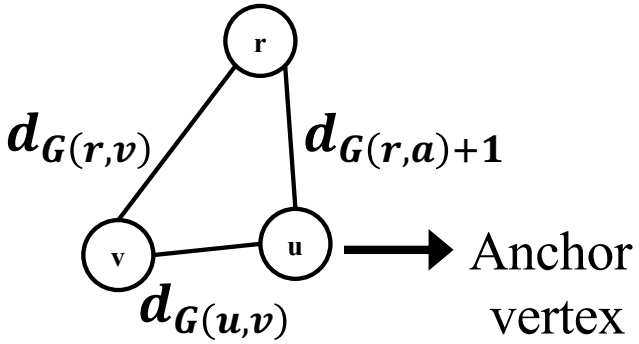
[3] **Farhan, M.**, Wang, Q., Koehler, H. "BatchHL: Answering Distance Queries on Batch-Dynamic Networks at Scale." Under review in The ACM SIGMOD/POD International Conference on Management of Data, 2022.

Proposed Approach - A High-Level Overview



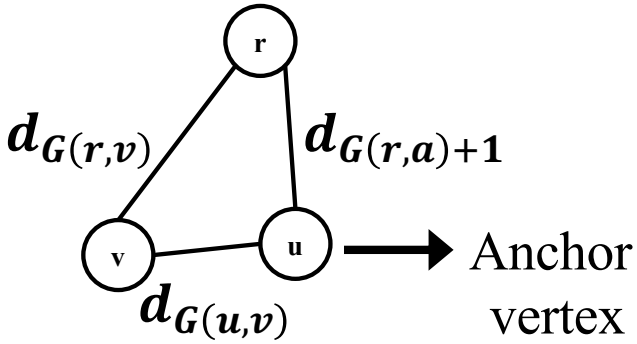
Batch Search

- Unify an update (a, b) into one process
  - both share the same pattern for affected vertices

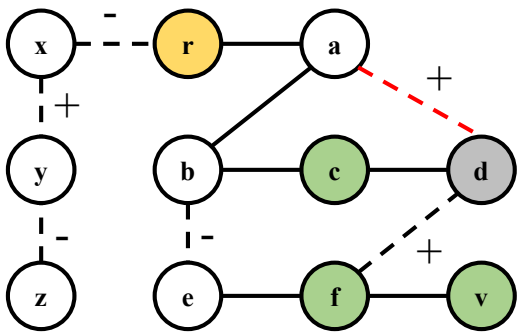


Batch Search

- Unify an update (a, b) into one process
  - both share the same pattern for affected vertices



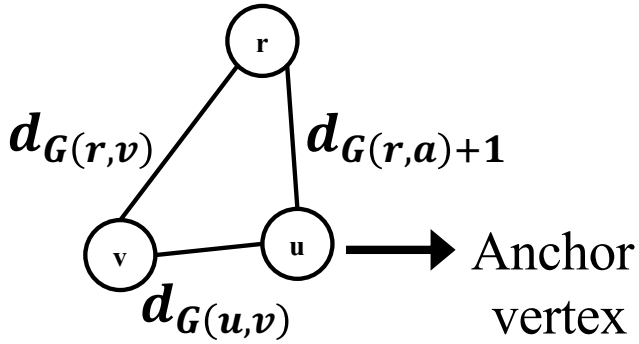
- Searches for vertices affected by different updates in a batch combine in a single search



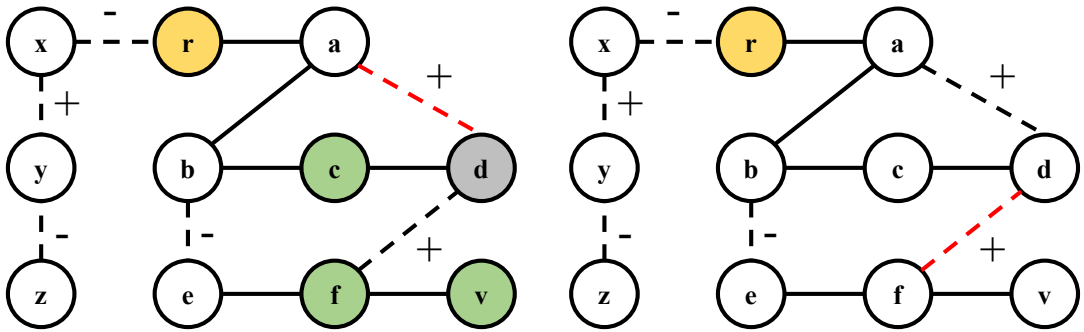
Insertion of (a, d)

Batch Search

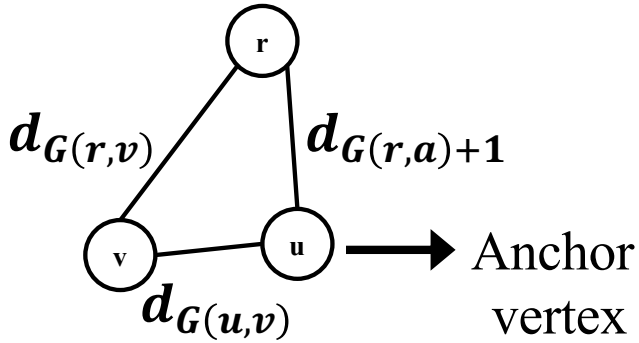
- Unify an update (a, b) into one process
  - both share the same pattern for affected vertices



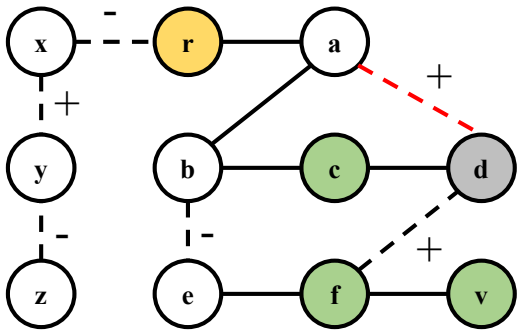
- Searches for vertices affected by different updates in a batch combine in a single search



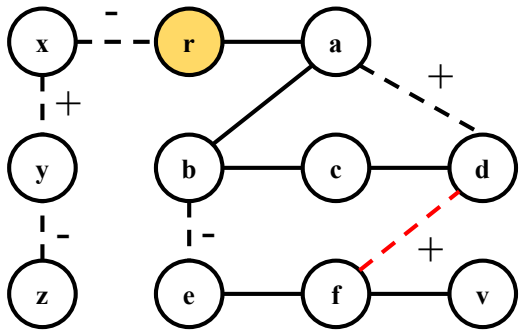
- Unify an update (a, b) into one process
  - both share the same pattern for affected vertices



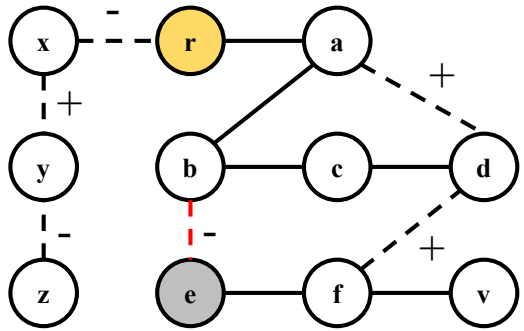
- Searches for vertices affected by different updates in a batch combine in a single search



Insertion of (a, d)

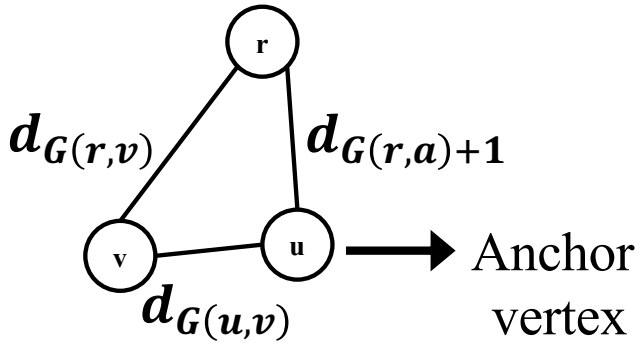


Insertion of (d, f)

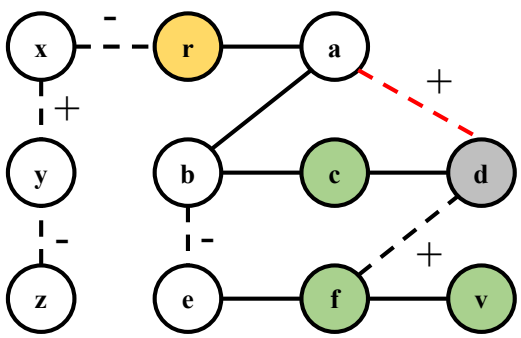


Deletion of (b, e)

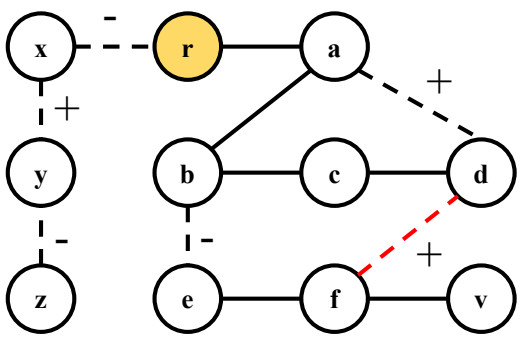
- Unify an update (a, b) into one process
  - both share the same pattern for affected vertices



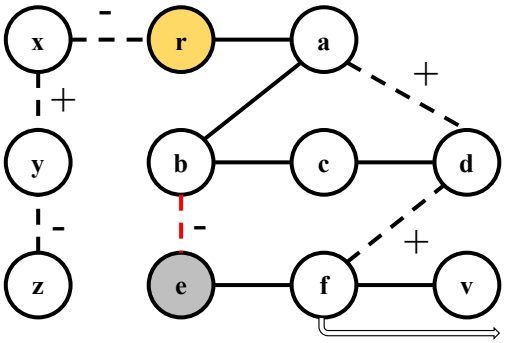
- Searches for vertices affected by different updates in a batch combine in a single search



Insertion of (a, d)



Insertion of (d, f)

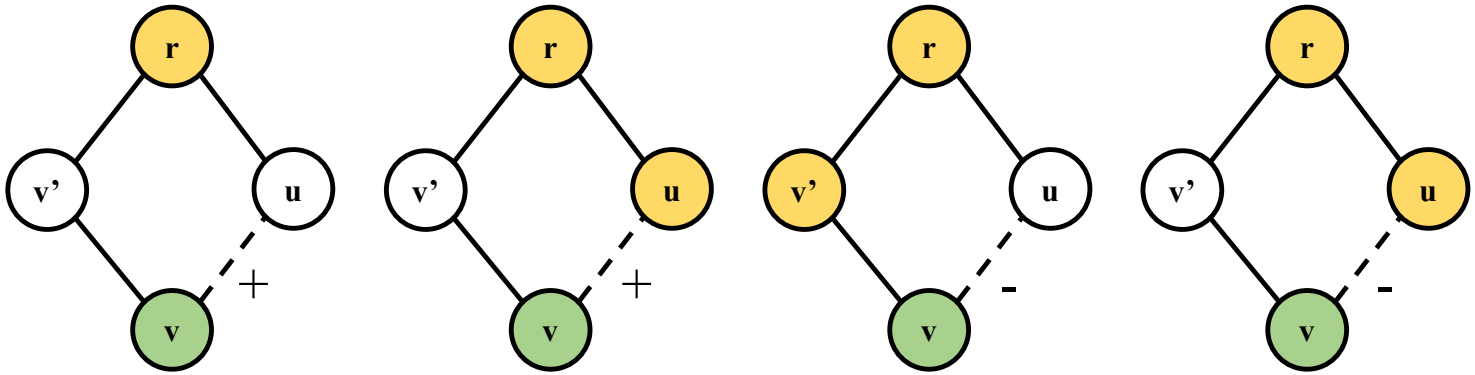


Deletion of (b, e)

Already found! (f,v) is traversed only once.

Improved Batch Search

- New shortest path of the same length as existing ones won't change the distance

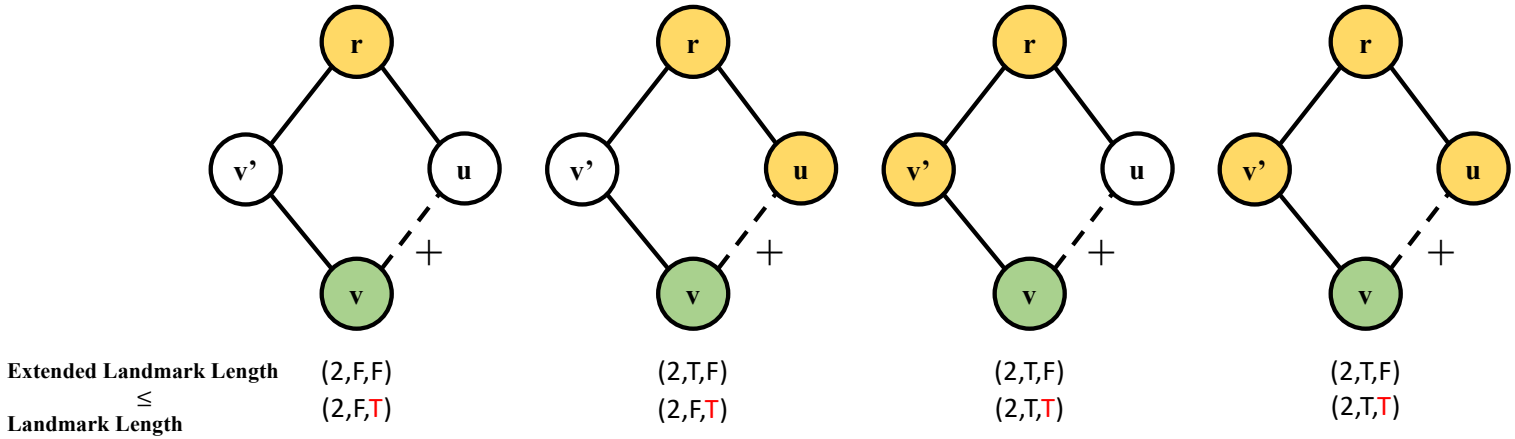


Improved Batch Search

- To identify such cases, we track whether a shortest path to r passes through (1) another landmark and (2) a deleted edge

Improved Batch Search

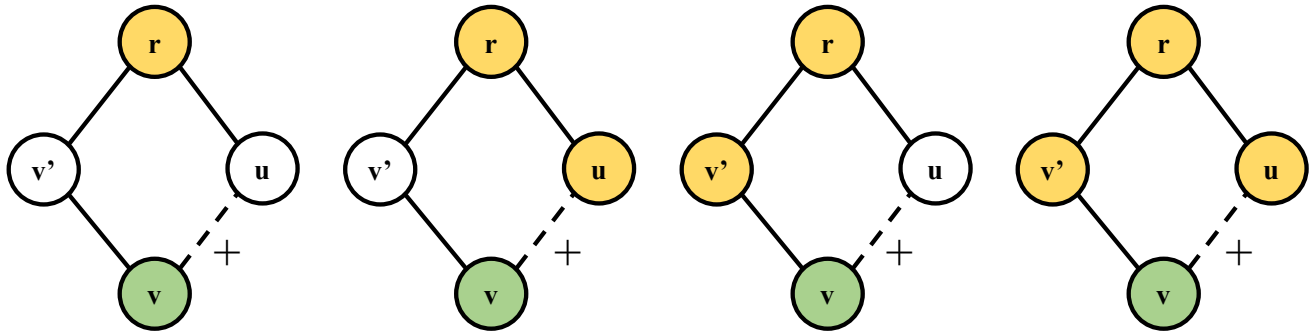
- To identify such cases, we track whether a shortest path to r passes through (1) another landmark and (2) a deleted edge



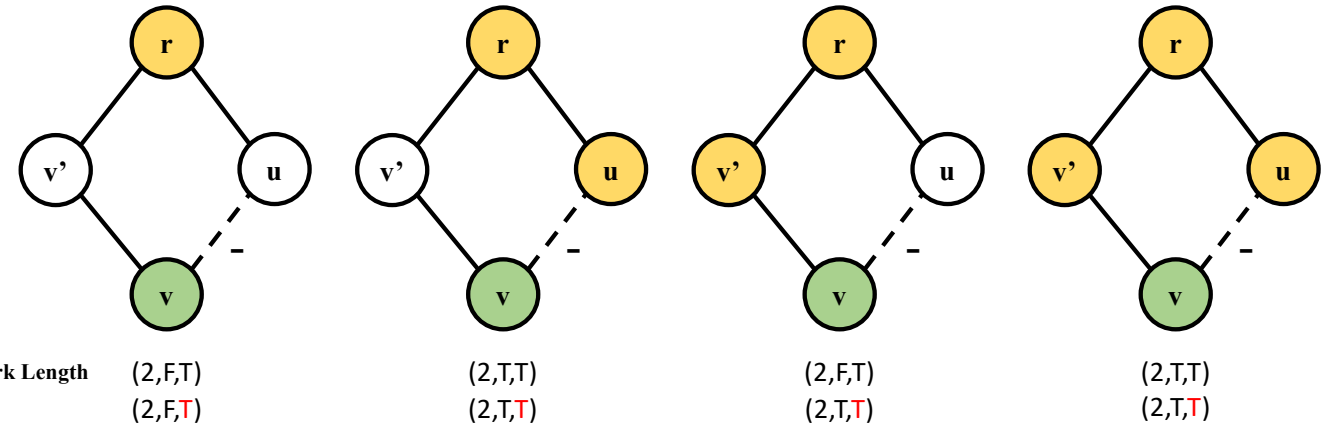
Addition Case - Pruning

Improved Batch Search

- To identify such cases, we track whether a shortest path to r passes through (1) another landmark and (2) a deleted edge



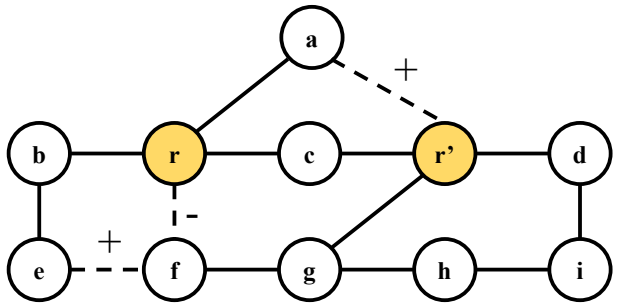
Addition Case - Pruning



Deletion Case - Pruning

Batch Repair

- Repair the labels of vertices return by Batch Search



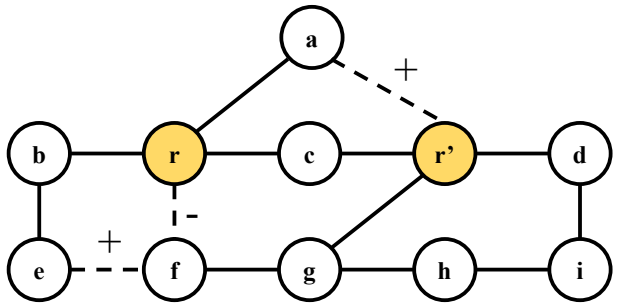
$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)

Batch Repair

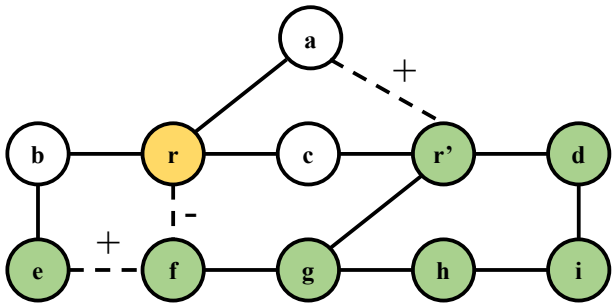
- Repair the labels of vertices return by Batch Search



$H = \{\delta_H(r, r') = 2\}$

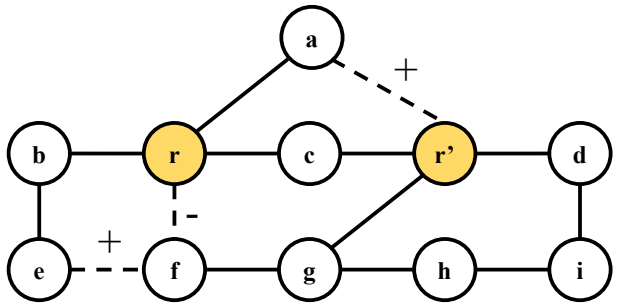
L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



Basic batch search algorithm
returns {r', d, e, f, g, h, i}

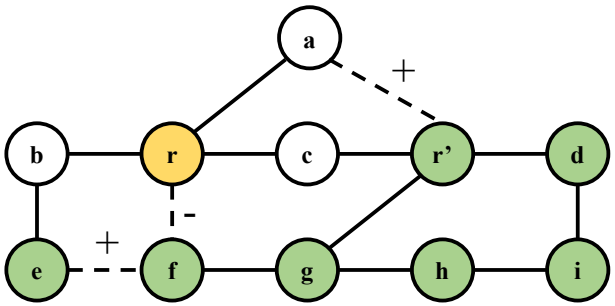
- Repair the labels of vertices return by Batch Search



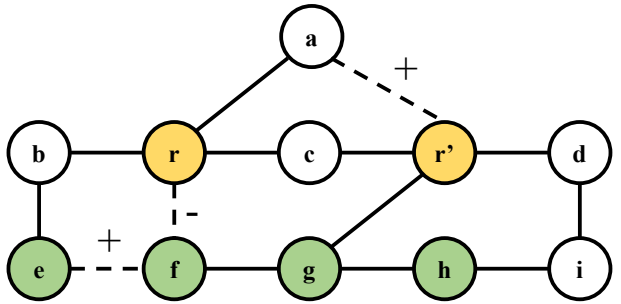
$H = \{\delta_H(r, r') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)

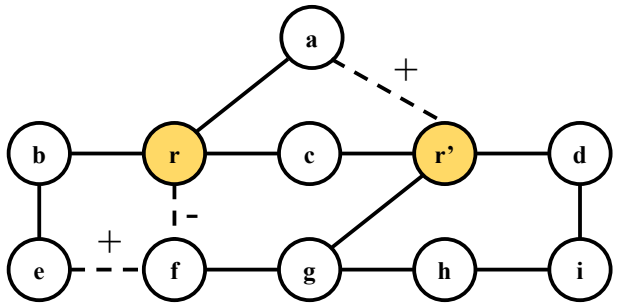


Basic batch search algorithm returns {r', d, e, f, g, h, i}



Improved batch search algorithm returns {e, f, g, h}

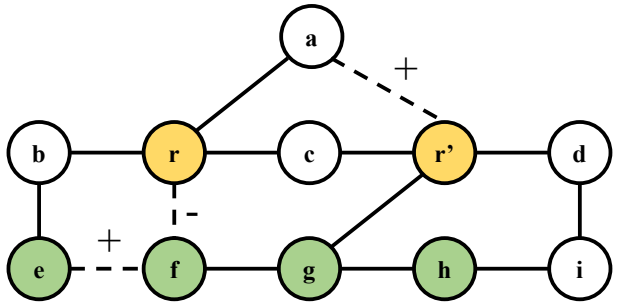
- Repair the labels of vertices return by Batch Search



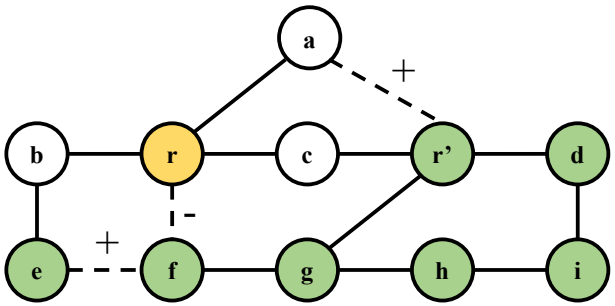
$H = \{\delta_H(r, r') = 2\}$

L =

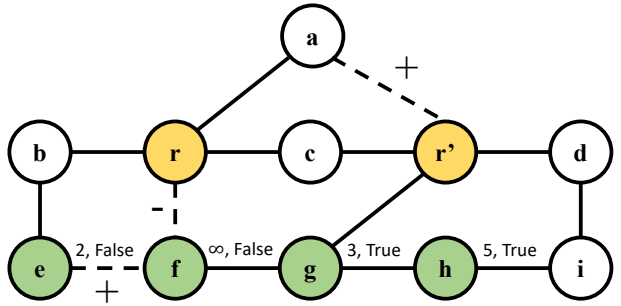
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



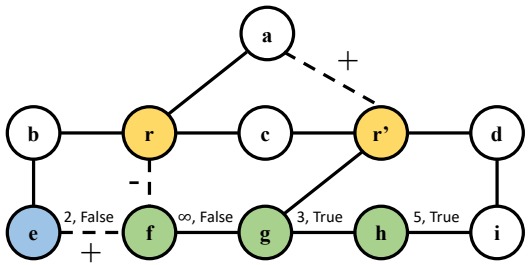
Improved batch search algorithm returns {e, f, g, h}



Basic batch search algorithm returns {r', d, e, f, g, h, i}



Batch Repair

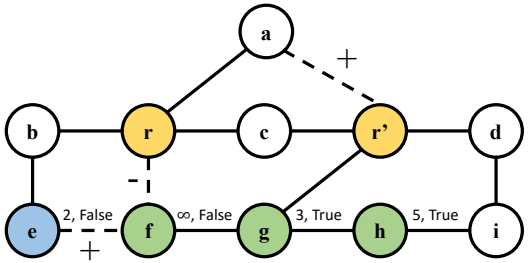


$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)

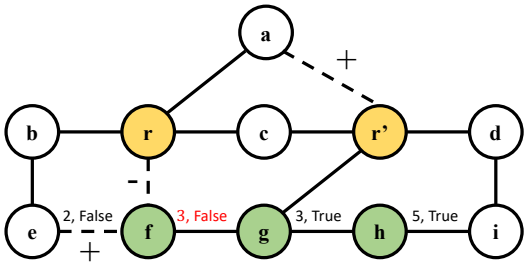
Batch Repair



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)

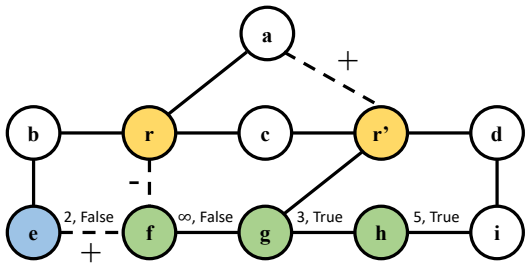


$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)

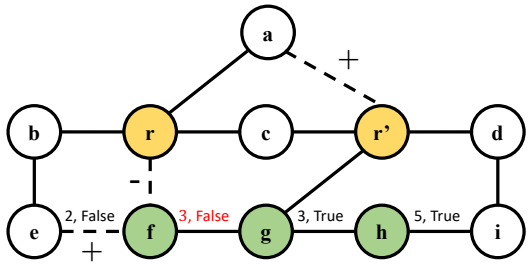
Batch Repair



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

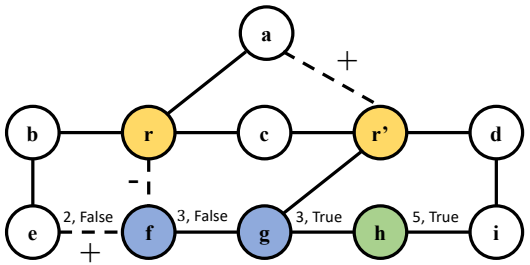
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

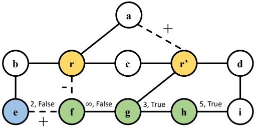
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

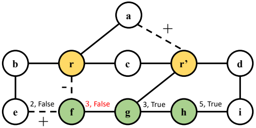
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(r, r') = 2\}$

L =

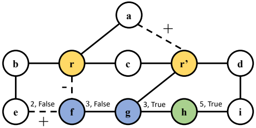
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(r, r') = 2\}$

L =

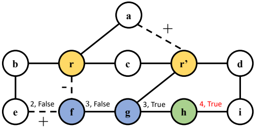
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(r, r') = 2\}$

L =

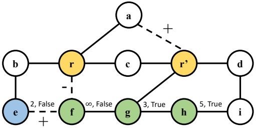
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(r, r') = 2\}$

L =

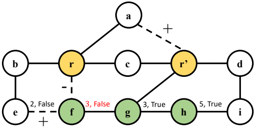
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 3) (r', 2)	~~(r, 2)~~ (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

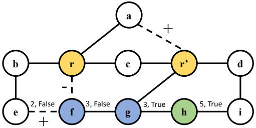
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1)		(r, 2)	(r, 1)	(r, 2)	(r, 3)	
		(r',1)	(r', 1)		(r', 2)	(r', 1)	(r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

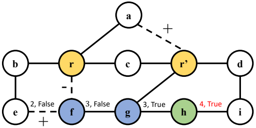
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1)		(r, 2)	(r, 1)	(r, 2)	(r, 3)	
		(r',1)	(r', 1)		(r', 2)	(r', 1)	(r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

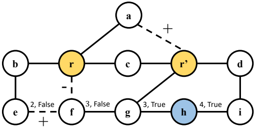
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1)		(r, 2)	(r, 1)	(r, 2)	(r, 3)	
		(r',1)	(r', 1)		(r', 2)	(r', 1)	(r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

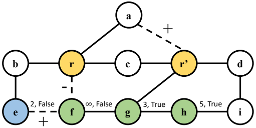
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1)		(r, 2)	(r, 3)	~~(r, 2)~~	(r, 3)	
		(r',1)	(r', 1)		(r', 2)	(r', 1)	(r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

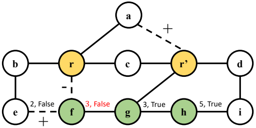
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1)		(r, 2)	(r, 3)		~~(r, 3)~~	
		(r',1)	(r', 1)		(r', 2)	(r', 1)	(r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

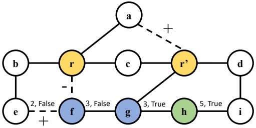
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

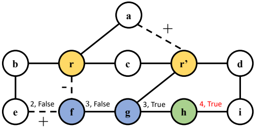
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

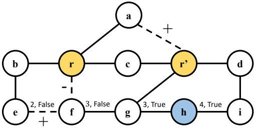
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 1) (r', 2)	(r, 2) (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

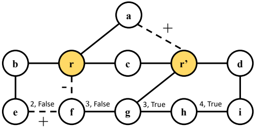
a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 3) (r', 2)	~~(r, 2)~~ (r', 1)	(r, 3) (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 3) (r', 2)	(r', 1)	~~(r, 3)~~ (r', 2)	(r', 2)



$H = \{\delta_H(\mathrm{r}, \mathrm{r}') = 2\}$

L =

a	b	c	d	e	f	g	h	i
(r, 1)	(r, 1)	(r, 1) (r',1)	(r', 1)	(r, 2)	(r, 3) (r', 2)	(r', 1)	(r', 2)	(r', 2)

Empirical Evaluation

Dataset	$\|V\|$	$\|E\|$	Fully Dynamic Batch Update Time (sec.)						Incremental Batch Update Time (sec.)					Decremental Batch Update Time (sec.)				
			BHL^p	BHL$^+$	BHL	UHL$^+$	FulFD	FulPLL	BHL^p	BHL$^+$	UHL$^+$	IncFD	IncPLL	BHL^p	BHL$^+$	UHL$^+$	DecFD	DecPLL
Youtube	1.1M	3M	**0.046**	**0.070**	**0.208**	0.091	1.249	9110	**0.003**	**0.008**	0.048	0.154	0.194	**0.070**	**0.169**	0.239	3.181	9850
Skitter	1.7M	11M	**0.147**	**0.601**	**0.902**	1.587	5.986	8770	**0.002**	**0.006**	0.069	0.117	1.312	**0.163**	**0.751**	2.382	14.15	31500
Flickr	1.7M	16M	**0.024**	**0.026**	**0.130**	0.099	2.152	6300	**0.003**	**0.008**	0.072	0.053	1.259	**0.030**	**0.041**	0.107	3.364	13400
Wikitalk	2.4M	5M	**0.029**	**0.025**	**0.101**	0.134	2.926	4550	**0.002**	**0.005**	0.097	0.029	0.081	**0.046**	**0.044**	0.147	5.674	9820
Hollywood	1.1M	114M	**0.008**	**0.014**	**0.115**	0.056	4.423	-	**0.001**	**0.002**	0.046	0.090	27.53	**0.017**	**0.031**	0.071	8.401	-
Orkut	3.1M	117M	**0.537**	**1.775**	**5.855**	4.539	13.30	-	**0.005**	**0.014**	0.127	0.367	-	**0.677**	**0.035**	5.921	23.94	-
Enwiki	4.3M	101M	**0.508**	**1.681**	**10.50**	3.952	121.7	-	**0.008**	**0.012**	0.168	0.316	4.916	**0.770**	**3.079**	8.194	251.2	-
Livejournal	4.8M	68M	**0.221**	**0.306**	**0.873**	0.379	4.736	-	**0.006**	**0.010**	0.202	0.244	-	**0.299**	**0.570**	0.731	4.736	-
Indochina	7.4M	194M	**0.543**	**1.181**	**1.547**	9.575	20.63	-	**0.015**	**0.011**	0.308	0.141	4.680	**0.553**	**1.346**	19.20	44.92	-
Twitter	42M	1.5B	**13.29**	**49.62**	**115.7**	125.6	5103	-	**0.125**	**0.024**	13.09	0.263	-	**19.17**	**68.85**	231.8	9460	-
Friendster	66M	1.8B	**0.409**	**0.410**	**0.811**	21.93	23.27	-	**0.163**	**0.035**	20.96	0.254	-	**0.420**	**0.738**	21.87	30.38	-
UK	106M	3.7B	**14.45**	**41.46**	**40.79**	56.50	110.1	-	**0.218**	**0.055**	4.349	0.258	-	**14.99**	**42.29**	75.20	257.3	-

Empirical Evaluation

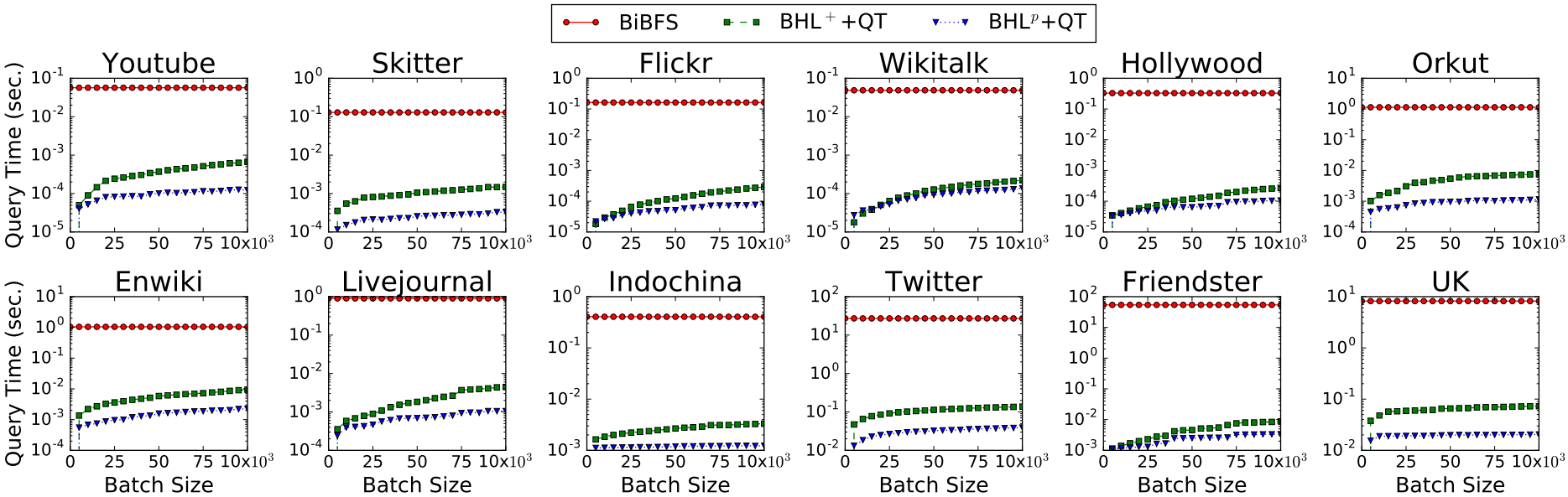
Dataset	$\|V\|$	$\|E\|$	Fully Dynamic Batch Update Time (sec.)						Incremental Batch Update Time (sec.)					Decremental Batch Update Time (sec.)				
			BHL^p	BHL$^+$	BHL	UHL$^+$	FulFD	FulPLL	BHL^p	BHL$^+$	UHL$^+$	IncFD	IncPLL	BHL^p	BHL$^+$	UHL$^+$	DecFD	DecPLL
Youtube	1.1M	3M	**0.046**	**0.070**	**0.208**	0.091	1.249	9110	**0.003**	**0.008**	0.048	0.154	0.194	**0.070**	**0.169**	0.239	3.181	9850
Skitter	1.7M	11M	**0.147**	**0.601**	**0.902**	1.587	5.986	8770	**0.002**	**0.006**	0.069	0.117	1.312	**0.163**	**0.751**	2.382	14.15	31500
Flickr	1.7M	16M	**0.024**	**0.026**	**0.130**	0.099	2.152	6300	**0.003**	**0.008**	0.072	0.053	1.259	**0.030**	**0.041**	0.107	3.364	13400
Wikitalk	2.4M	5M	**0.029**	**0.025**	**0.101**	0.134	2.926	4550	**0.002**	**0.005**	0.097	0.029	0.081	**0.046**	**0.044**	0.147	5.674	9820
Hollywood	1.1M	114M	**0.008**	**0.014**	**0.115**	0.056	4.423	-	**0.001**	**0.002**	0.046	0.090	27.53	**0.017**	**0.031**	0.071	8.401	-
Orkut	3.1M	117M	**0.537**	**1.775**	**5.855**	4.539	13.30	-	**0.005**	**0.014**	0.127	0.367	-	**0.677**	**0.035**	5.921	23.94	-
Enwiki	4.3M	101M	**0.508**	**1.681**	**10.50**	3.952	121.7	-	**0.008**	**0.012**	0.168	0.316	4.916	**0.770**	**3.079**	8.194	251.2	-
Livejournal	4.8M	68M	**0.221**	**0.306**	**0.873**	0.379	4.736	-	**0.006**	**0.010**	0.202	0.244	-	**0.299**	**0.570**	0.731	4.736	-
Indochina	7.4M	194M	**0.543**	**1.181**	**1.547**	9.575	20.63	-	**0.015**	**0.011**	0.308	0.141	4.680	**0.553**	**1.346**	19.20	44.92	-
Twitter	42M	1.5B	**13.29**	**49.62**	**115.7**	125.6	5103	-	**0.125**	**0.024**	13.09	0.263	-	**19.17**	**68.85**	231.8	9460	-
Friendster	66M	1.8B	**0.409**	**0.410**	**0.811**	21.93	23.27	-	**0.163**	**0.035**	20.96	0.254	-	**0.420**	**0.738**	21.87	30.38	-
UK	106M	3.7B	**14.45**	**41.46**	**40.79**	56.50	110.1	-	**0.218**	**0.055**	4.349	0.258	-	**14.99**	**42.29**	75.20	257.3	-

- Much improved update time for all three settings.
- Decremental batch updates are much more faster.

Empirical Evaluation

Average affected vertices after performing batch updates collectively (in a batch) and individually (one by one).

	Method	Type	Youtube	Skitter	Flickr	Wikitalk	Hollywood	Orkut	Enwiki	Livejournal	Indochina	Twitter	Friendster	UK
affected	BHL$^+$	Delete	366 K	971 K	55 K	127 K	14 K	503 K	1,220 K	276 K	2,079 K	10,622 K	66 K	54,515 K
		Add	23 K	11 K	22 K	16 K	2 K	3 K	4 K	12 K	15 K	2 K	6 K	12 K
		Mix	166 K	834 K	42 K	81 K	7 K	293 K	712 K	156 K	200 K	8,341 K	36 K	54,026 K
	BHL	Mix	476 K	1,266 K	157 K	474 K	41 K	982 K	3,587 K	454 K	3,085 K	20,705 K	80 K	54,864 K



Query time of the proposed methods against online search methods.

Conclusion

1. A scalable algorithm (HL) for answering distance queries on static graphs, which has several nice properties:
   (1) Minimal labelling
   (2) Order independence
   (3) Parallel construction
2. Efficient methods to process graph changes in the unit update setting for answering distance queries on very large dynamic graphs.
3. An efficient batch update method for answering distance queries on graphs undergoing batch updates, which has the following benefits
   (1) Unifying edge insertion and deletion
   (2) Avoiding unnecessary and repeated computations
   (3) Exploiting the potential for parallelism

Future Work

- Extension of the proposed approaches to road networks
  - different structure than complex networks
- Selection of highly central vertices
  - improved query time
  - reduced labelling size
- Guided search by investigating properties of the proposed approaches
  - improved query time

Questions

Thank You