Efficient Maintenance of Distance Labelling for Incremental Updates in Large Dynamic Graphs

Muhammad Farhan and Qing Wang

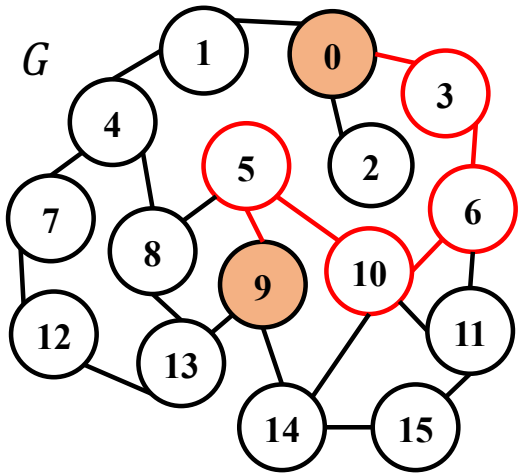
School of Computing, Australian National University
Canberra, Australia

{muhammad.farhan,qing.wang}@anu.edu.au

June 17, 2021

Shortest Path Distance Queries

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

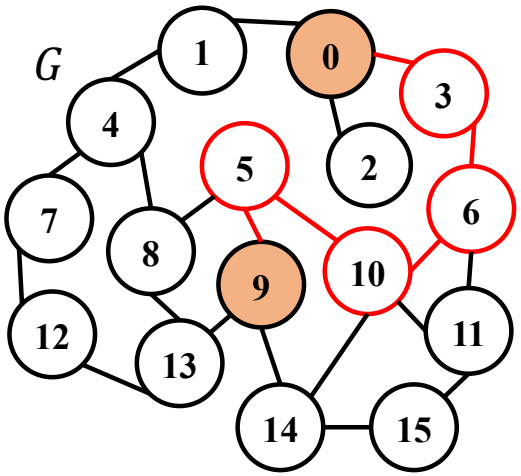


$d_G(0, 9) = 5$

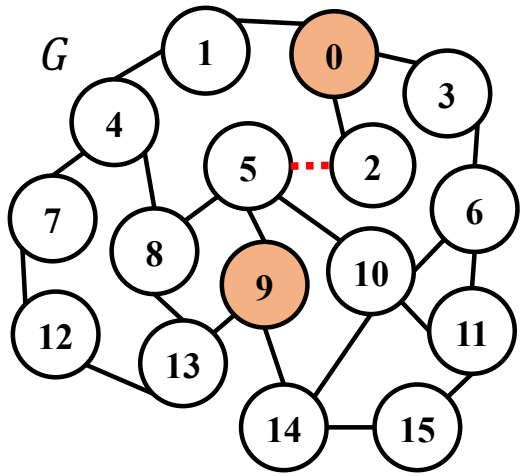
Dynamic Shortest Path Distance Queries

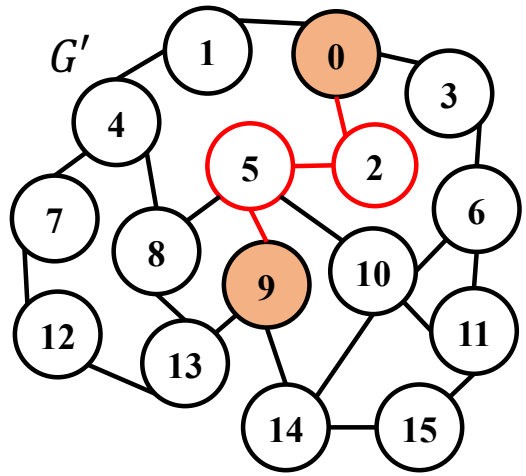
Let $G \hookrightarrow G'$ denote that G is changed to G' by an edge insertion.

- **Problem:** Given a graph $G = (V, E)$, any two vertices $s, t \in V$ and $G \hookrightarrow G'$, to answer shortest path distance $d_{G'}(s, t)$.



$d_G(0, 9) = 5$

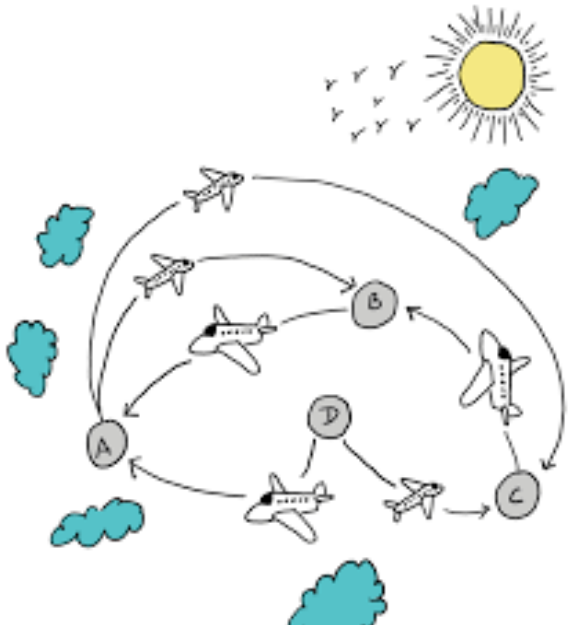




$d_{G'}(0, 9) = 3$

Applications

- Context-aware web search
- Social network analysis
  - Socially sensitive search
  - Closeness centrality
  - Community detection/search
  - ...
- Route navigation
- ...



Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS

Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
  - – no update time
  - – slow query time

Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
  - – no update time
  - – slow query time
- **Labelling-based approaches**
  - Tree decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013]

Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
  - – no update time – slow query time
- **Labelling-based approaches**
  - Tree decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013]
  - – fast query time – slow update time – limited scalability

Related Work

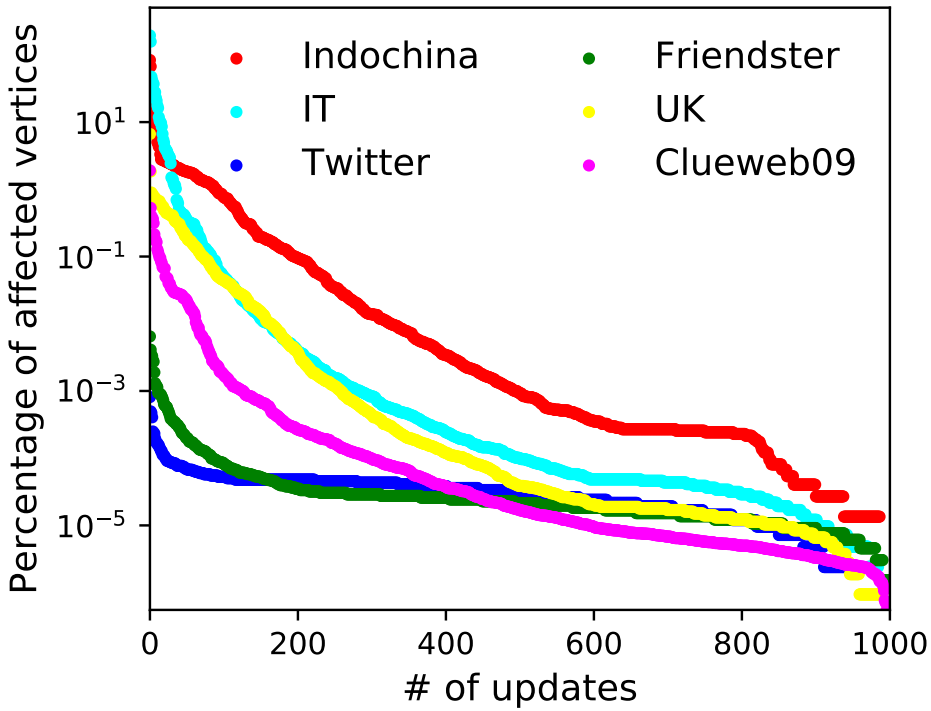
- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
  - – no update time – slow query time
- **Labelling-based approaches**
  - Tree decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013]
  - – fast query time – slow update time – limited scalability
- **Hybrid approaches (Labelling + Online search)**
  - IS-Label (IS-L) method [VLDB2013]
  - Fully Dynamic (FD) method [CIKM2016]
  - Highway labelling (HL) approach [EDBT2019]

Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
  - – no update time – slow query time
- **Labelling-based approaches**
  - Tree decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013]
  - – fast query time – slow update time – limited scalability
- **Hybrid approaches (Labelling + Online search)**
  - IS-Label (IS-L) method [VLDB2013]
  - Fully Dynamic (FD) method [CIKM2016]
  - Highway labelling (HL) approach [EDBT2019]
  - – high scalability – fast query time – slow update time

Maintenance of distance labelling

- very long time to construct a distance labelling
- a very small percentage of vertices is affected



Related Work - Maintenance of distance Labelling

- **Labelling-based approaches**
  - TreE Decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013] - [WWW2014]
- **Hybrid approaches**
  - IS-Label (IS-L) method [VLDB2013]
  - Fully Dynamic (FD) method [CIKM2016]
  - Highway labelling (HL) approach [EDBT2019]

Related Work - Maintenance of distance Labelling

- **Labelling-based approaches**
  - TreE Decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013] - [WWW2014]
- **Hybrid approaches**
  - IS-Label (IS-L) method [VLDB2013]
  - Fully Dynamic (FD) method [CIKM2016]
  - Highway labelling (HL) approach [EDBT2019]

– limited scalability – long update time – no minimality

Related Work - Maintenance of distance Labelling

- **Labelling-based approaches**
  - TreE Decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013] - [WWW2014]
- **Hybrid approaches**
  - IS-Label (IS-L) method [VLDB2013]
  - Fully Dynamic (FD) method [CIKM2016]
  - Highway labelling (HL) approach [EDBT2019]

– high scalability – have minimality – not dynamic

Distance Labelling

- Given a set of landmarks $R \subseteq V$ of G, a label $L(v)$ for each $v \in V$ can be precomputed, i.e.,

$$L(v) = \{(r_1, \delta(r_1, v)), \ldots, (r_n, \delta(r_n, v))\},$$

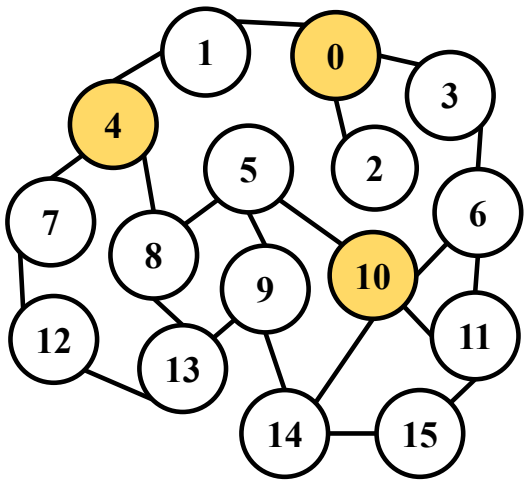
where $r_i \in R$ and $\delta(r_i, v) = d_G(r_i, v)$.

Distance Labelling

- Given a set of landmarks $R \subseteq V$ of G, a label $L(v)$ for each $v \in V$ can be precomputed, i.e.,

$$L(v) = \{(r_1, \delta(r_1, v)), \ldots, (r_n, \delta(r_n, v))\},$$

where $r_i \in R$ and $\delta(r_i, v) = d_G(r_i, v)$.



$R = \{0, 4, 10\}$

$L(1)$

Landmark	0	4
Distance	1	1

$L(2)$

Landmark	0
Distance	1

$d_G(0, 2) = 1$

$L(3)$

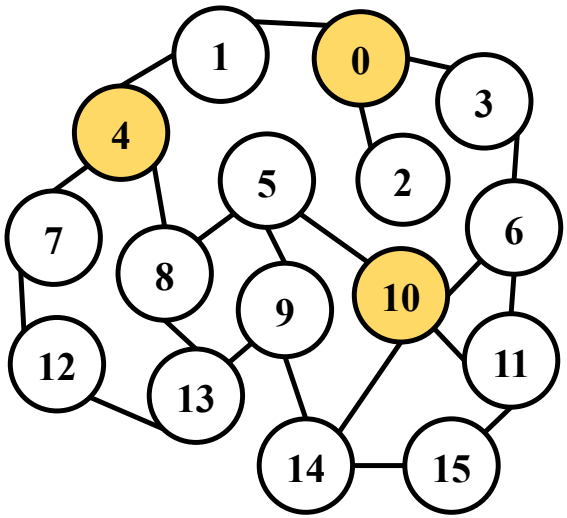
Landmark	0	10
Distance	1	2

.....

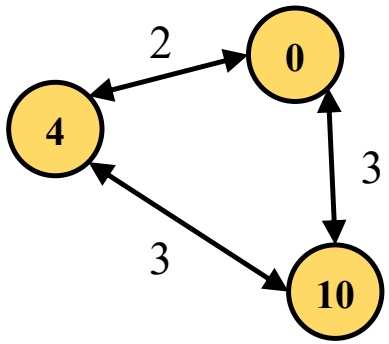
Highway Cover Labelling

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 $\delta_H(r_1, r_2) = d_G(r_1, r_2)$.



Graph

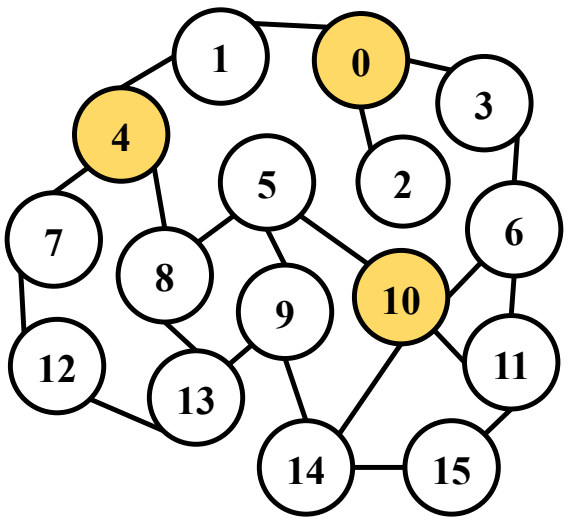


Highway

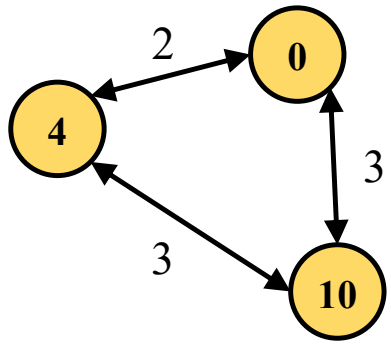
Highway Cover Labelling

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.



Graph



Highway

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)
$L(15)$	(10, 2)

Labelling

Querying Framework

Answering $d_G(s,t)$ in a graph G:

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

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

Querying Framework

Answering $d_G(s,t)$ in a graph G:

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

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

Definition

$$d_G(s,t) = \begin{cases} d_{G[V\backslash R]}(s,t), & \text{if } d_{G[V\backslash R]}(s,t) \leq d_{st}^{\top} \\ d_{st}^{\top}, & \text{otherwise} \end{cases}$$

Computing Upper Bounds

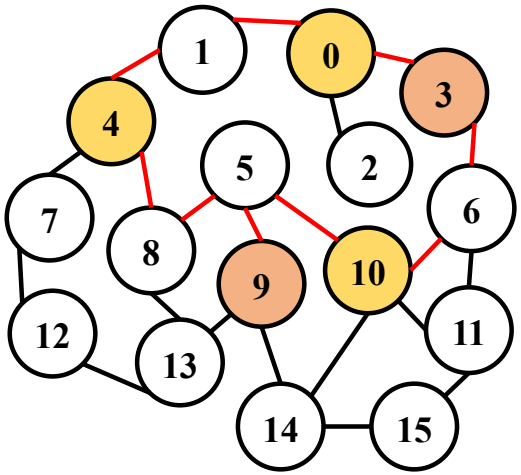
- Find a path of the minimal length through a highway H:

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

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=3 and t=9?

$L(3)$

Landmark	0	10
Distance	1	2

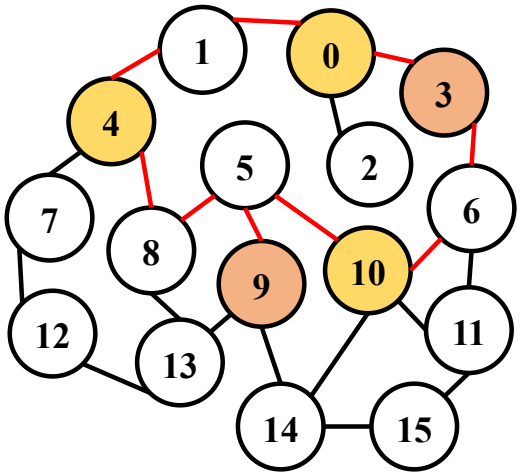
$L(9)$

Landmark	4	10
Distance	3	2

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=3 and t=9?

$L(3)$

Landmark	0	10
Distance	1	2

- $3 \to 0 \to 4 \to 9$: length 6

$L(9)$

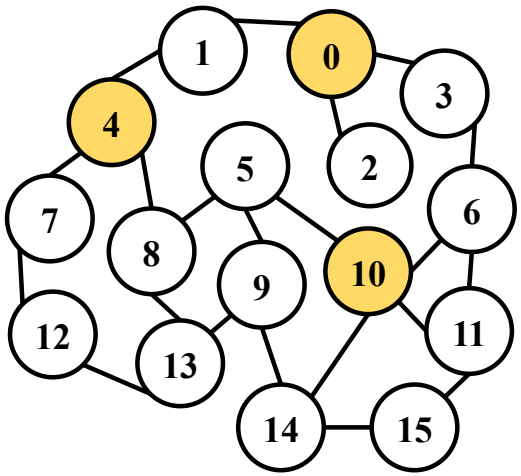
Landmark	4	10
Distance	3	2

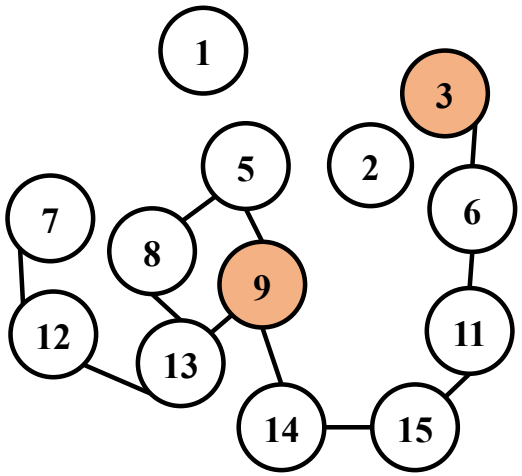
- $3 \to 10 \to 9$: length 4

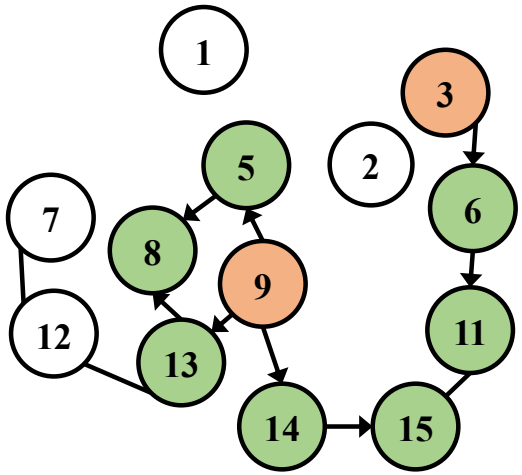
$$d_{st}^{\top} = 4$$

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 $d_{st}^{\top}$



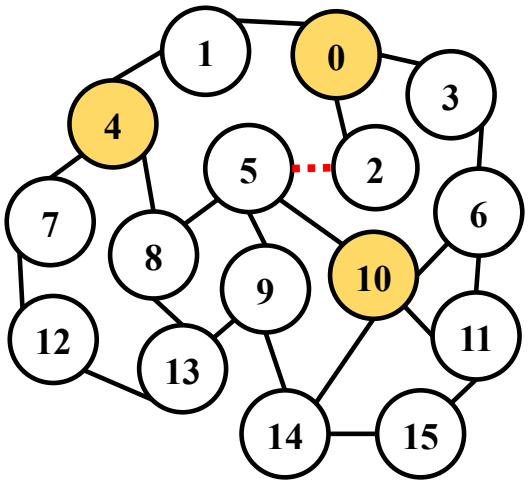




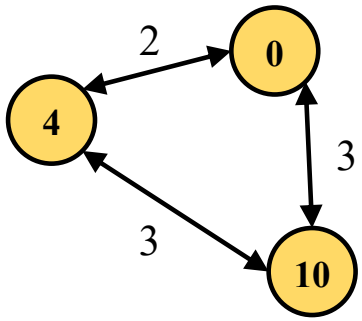
$d_G(3, 9) = 4$

Maintenance of distance labelling

– Graph changes by edge/vertex insertions may result in overestimated distances.



Graph



Highway

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)
$L(15)$	(10, 2)

Labelling

Online Incremental Method - IncHL$^+$

Our IncHL$^+$ maintains a highway cover labelling in two steps:

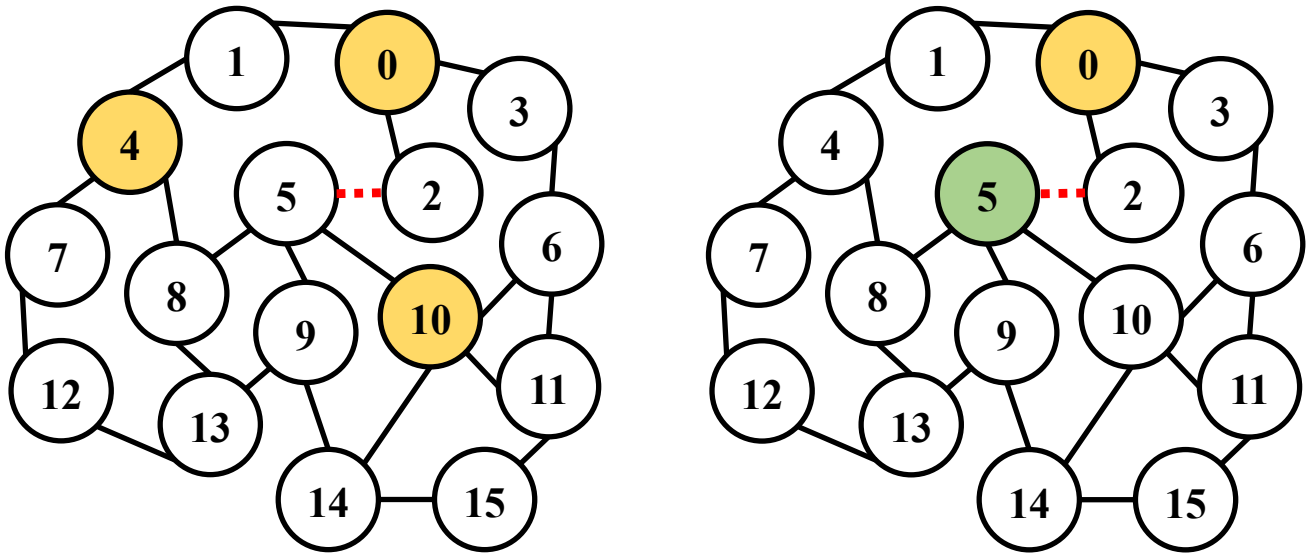
- **Step 1**. Efficiently find the affected vertices caused by an edge insertion
- **Step 2**. Efficiently repair the labels of affected vertices while preserving the minimality of distance labelling

Step 1: Finding Affected Vertices

Definition

A vertex v is *affected* by $G \hookrightarrow G'$ iff $P_G(v, r) \neq P_{G'}(v, r)$ for at least one landmark r; *unaffected* otherwise.

- We conduct partial BFSs to identify affected vertices, e.g., a partial BFS starting from vertex 5 w.r.t. the landmark 0 after inserting the edge $(2, 5)$ is described below, where affected vertices are colored in green.

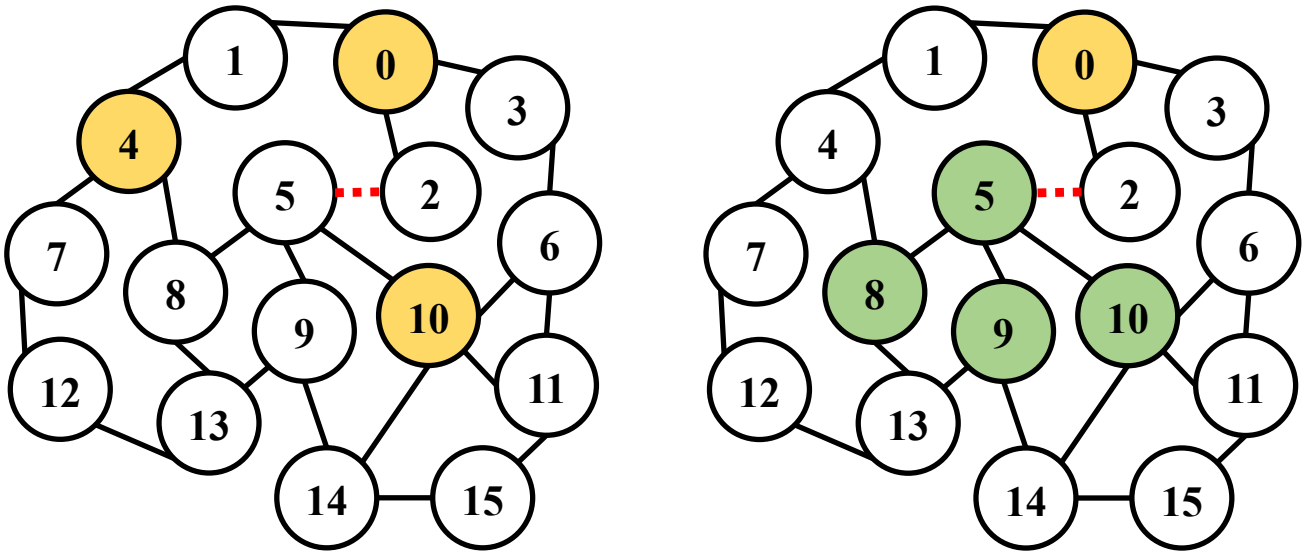


Step 1: Finding Affected Vertices

Definition

A vertex v is *affected* by $G \hookrightarrow G'$ iff $P_G(v, r) \neq P_{G'}(v, r)$ for at least one landmark r; *unaffected* otherwise.

- We conduct partial BFSs to identify affected vertices, e.g., a partial BFS starting from vertex 5 w.r.t. the landmark 0 after inserting the edge $(2, 5)$ is described below, where affected vertices are colored in green.

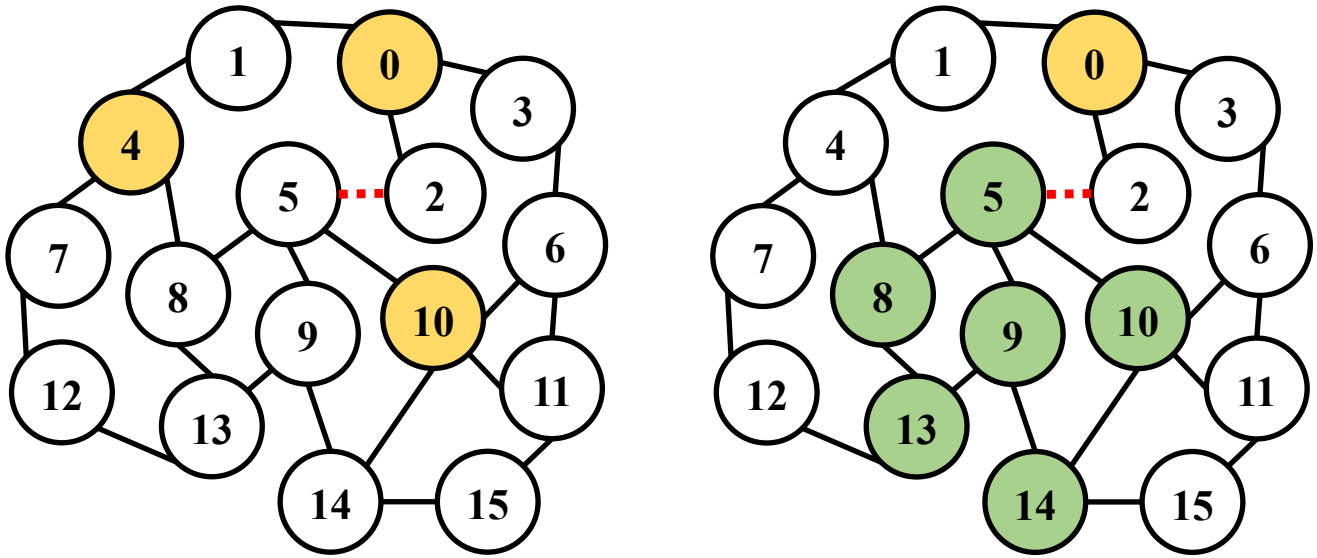


Step 1: Finding Affected Vertices

Definition

A vertex v is *affected* by $G \hookrightarrow G'$ iff $P_G(v, r) \neq P_{G'}(v, r)$ for at least one landmark r; *unaffected* otherwise.

- We conduct partial BFSs to identify affected vertices, e.g., a partial BFS starting from vertex 5 w.r.t. the landmark 0 after inserting the edge $(2, 5)$ is described below, where affected vertices are colored in green.



Step 2: Repairing Affected Vertices

- We also conduct partial BFSs to repair affected vertices.
- We distinguish two types to improve efficiency:
  1. *Covered* - repaired by removing an entry from their labels (if exists)
  2. *Uncovered* - repaired by accurately calculating distances.

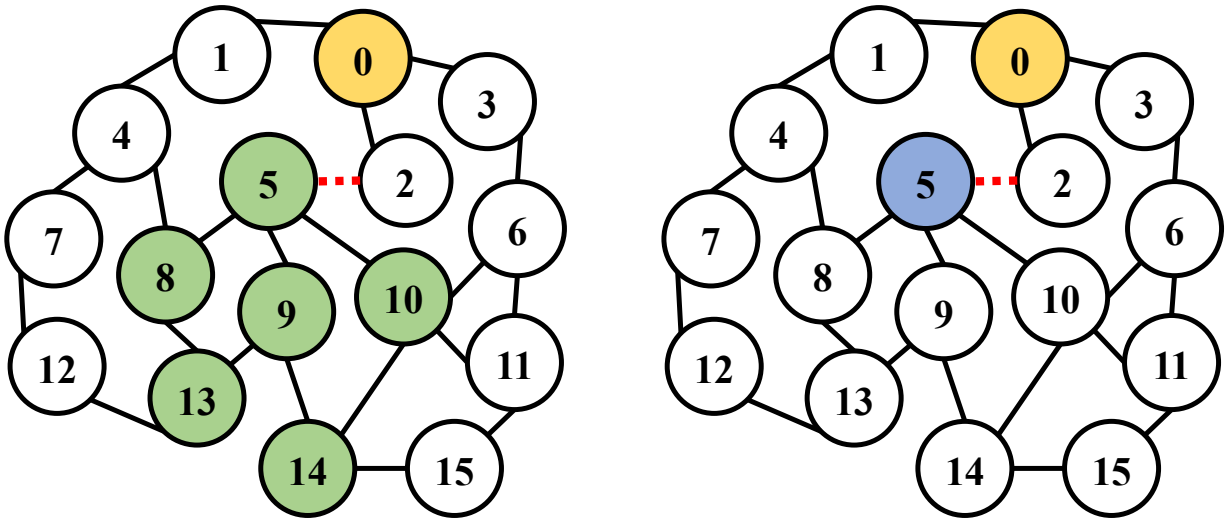
Lemma

Let Λ_r be the set of all affected vertices w.r.t. a landmark r and $v \in \Lambda_r$.

- *v is covered by a landmark $r' \in R \backslash \{r\}$ iff r' exists in $P_{G'}(v, r)$.*
- *If v is covered by r', then any $v' \in \Lambda_r$ satisfying $d_{G'}(r, v') = d_{G'}(r, v) + d_{G'}(v, v')$ must also be covered by r'.*

Step 2: Repairing Affected Vertices

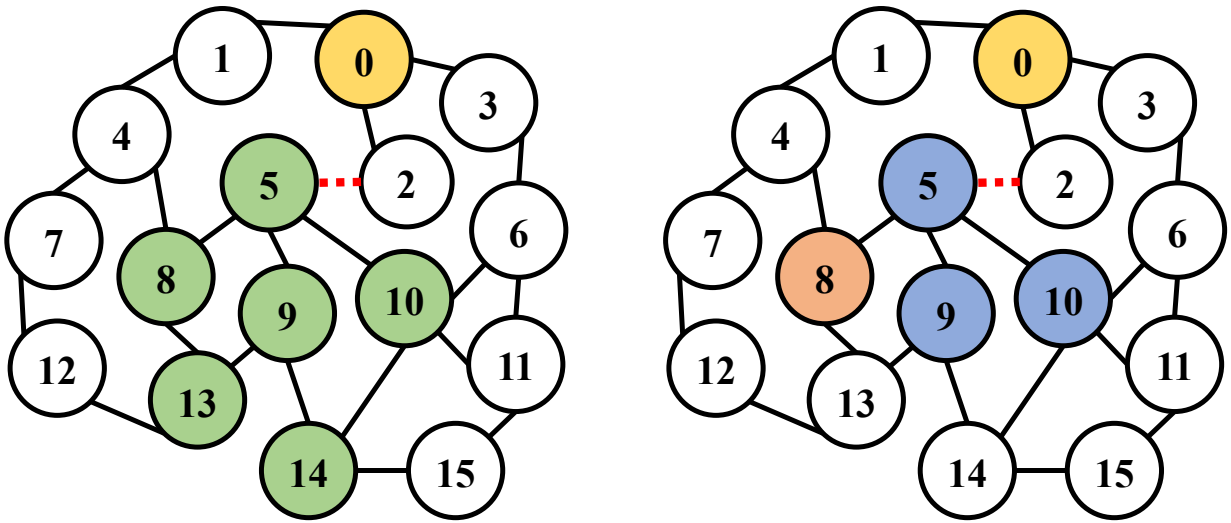
– A partial BFS starting from vertex 5 w.r.t. the landmark 0 is described below.



- Blue: repaired vertices with added/modified entries

Step 2: Repairing Affected Vertices

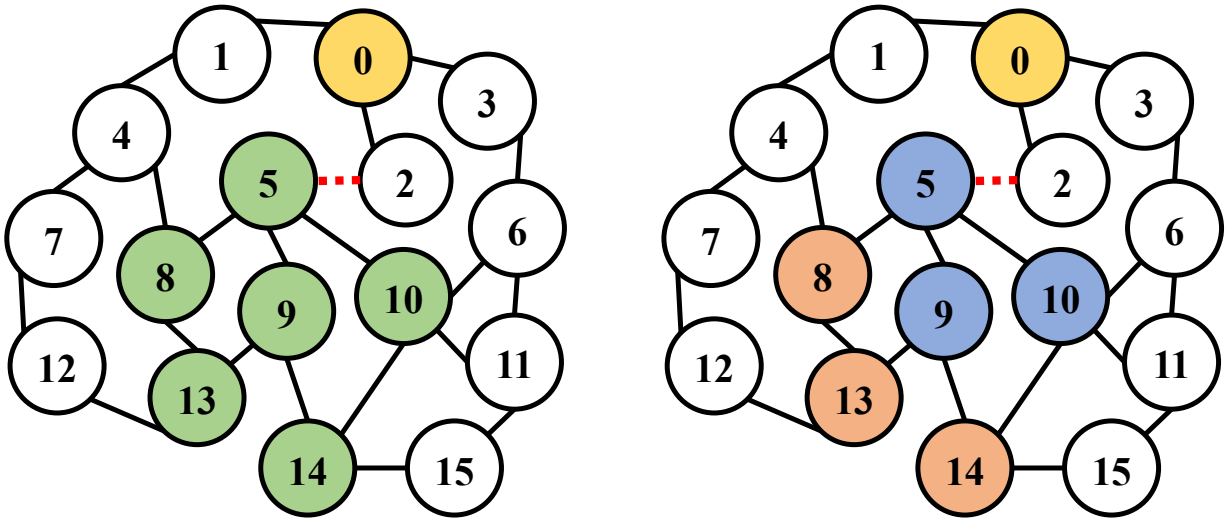
– A partial BFS starting from vertex 5 w.r.t. the landmark 0 is described below.



- Red: repaired vertices with removed entries
- Blue: repaired vertices with added/modified entries

Step 2: Repairing Affected Vertices

– A partial BFS starting from vertex 5 w.r.t. the landmark 0 is described below.



- Red: repaired vertices with removed entries
- Blue: repaired vertices with added/modified entries

Experiments

- **12 large-scale real-world networks**
  - Undirected and unweighted graphs
  - Sizes from $|V|$=1.7M & $|E|$=11M to $|V|$=2B & $|E|$=8B
  - Domains: social networks, web networks, etc.
- **Graph updates**
  - Randomly sample 1,000 pairs of vertices in each network as edge insertions to evaluate the average update time.
- **Distance queries**
  - Randomly sample 100,000 pairs of vertices from all pairs of vertices in each network

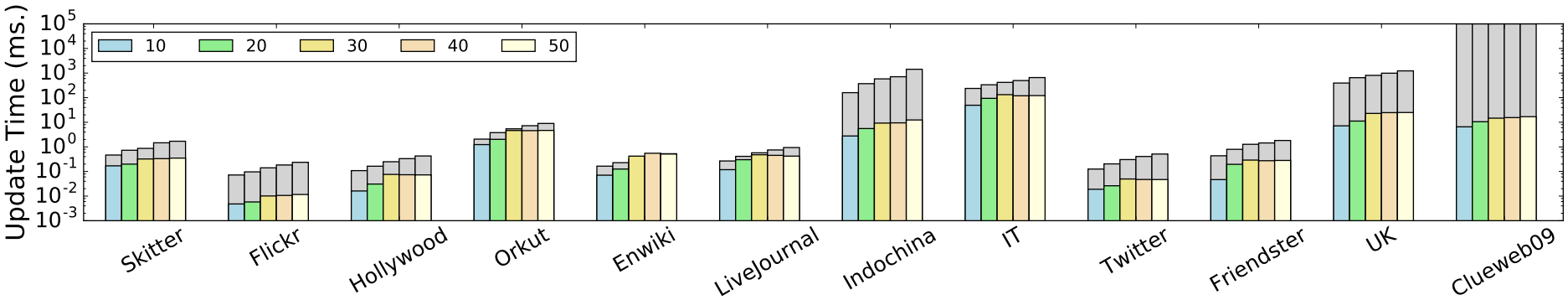
Efficiency

How efficiently can our method perform against state-of-the-art methods?

Dataset	Update Time (ms)			Query Time (ms)			Labelling Size		
	IncHL$^+$	IncFD	IncPLL	IncHL$^+$	IncFD	IncPLL	IncHL$^+$	IncFD	IncPLL
Skitter	0.194	0.444	2.05	0.027	0.019	0.047	42 MB	153 MB	2.44 GB
Flickr	0.006	0.074	1.73	0.007	0.012	0.064	34 MB	152 MB	3.69 GB
Hollywood	0.031	0.101	48	0.027	0.037	0.109	27 MB	263 MB	12.58 GB
Orkut	2.026	2.049	-	0.101	0.103	-	70 MB	711 MB	-
Enwiki	0.134	0.163	5.91	0.054	0.035	0.071	82 MB	608 MB	12.57 GB
Livejournal	0.245	0.268	-	0.044	0.046	-	122 MB	663 MB	-
Indochina	5.443	158	2018	0.737	0.839	0.063	81 MB	838 MB	18.64 GB
IT	95.92	224	-	1.069	1.013	-	854 MB	4.74 GB	-
Twitter	0.027	0.134	-	0.863	0.177	-	1.14 GB	3.83 GB	-
Friendster	0.159	0.419	-	0.814	0.904	-	2.43 GB	9.14 GB	-
UK	11.49	384	-	3.443	5.858	-	1.78 GB	11.8 GB	-
Clueweb09	40.68	-	-	16.93	-	-	163 GB	-	-

Performance under varying Landmarks

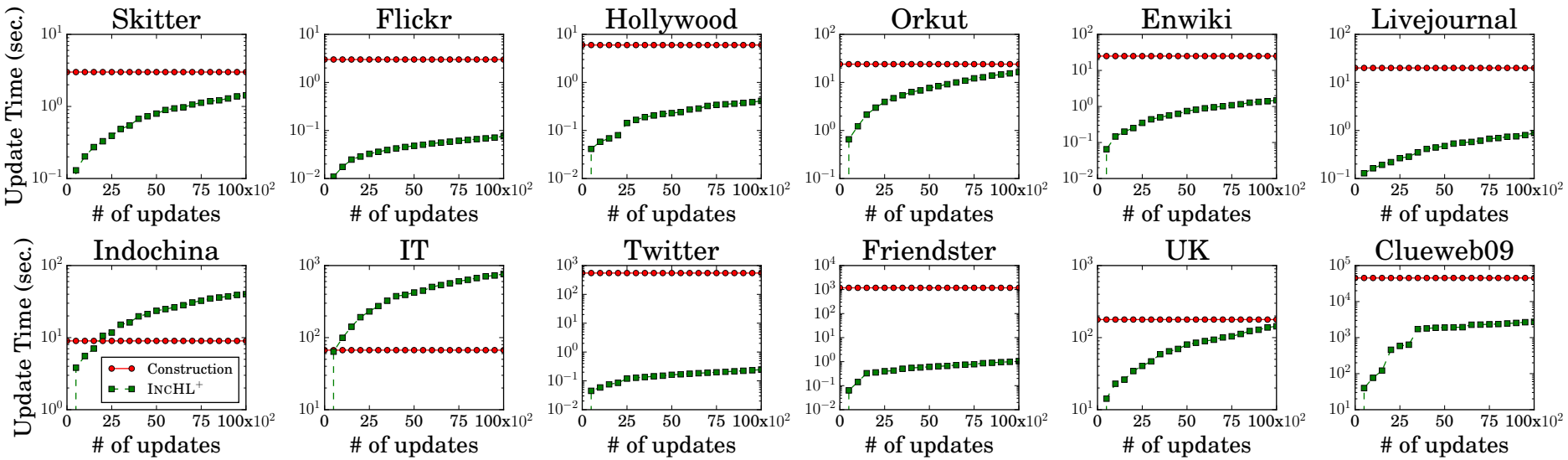
How does the number of landmarks affect the performance of our method?



- Colored bars: our online incremental method IncHL$^+$
- Colored plus grey bars: fully dynamic method IncFD

Scalability

How does our method scale to perform updates occurring rapidly in large dynamic networks?



Conclusion

- **Main contributions**

  An online incremental algorithm that

  (1) efficiently reflect incremental updates in billion-scale graphs;

  (2) preserve minimality of highway cover labelling.

Conclusion

- **Main contributions**

  An online incremental algorithm that

  (1) efficiently reflect incremental updates in billion-scale graphs;

  (2) preserve minimality of highway cover labelling.

- **Future works**
  - Decremental updates - edge/vertex deletions
  - Batch of updates in dynamic graphs
  - Landmark selection strategies