A Highly Scalable Labelling Approach for Exact Distance Queries in Complex Networks

Muhammad Farhan, Qing Wang, Yu Lin, and Brendan Mckay

Australian National University
Research School of Computer Science
Canberra, Australia

{muhammad.farhan,qing.wang,yu.lin,brendan.mckay}@anu.edu.au

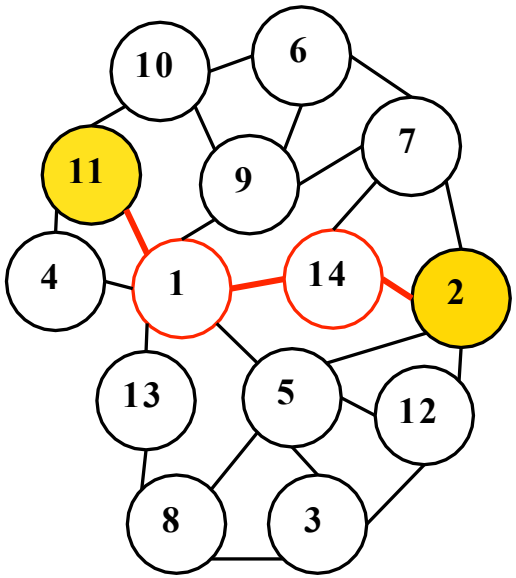
June 17, 2021

Exact 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 $d_G(s, t)$.

Exact 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 $d_G(s, t)$.



$$d_G(2, 11) = 3$$

Exact Shortest Path Distance

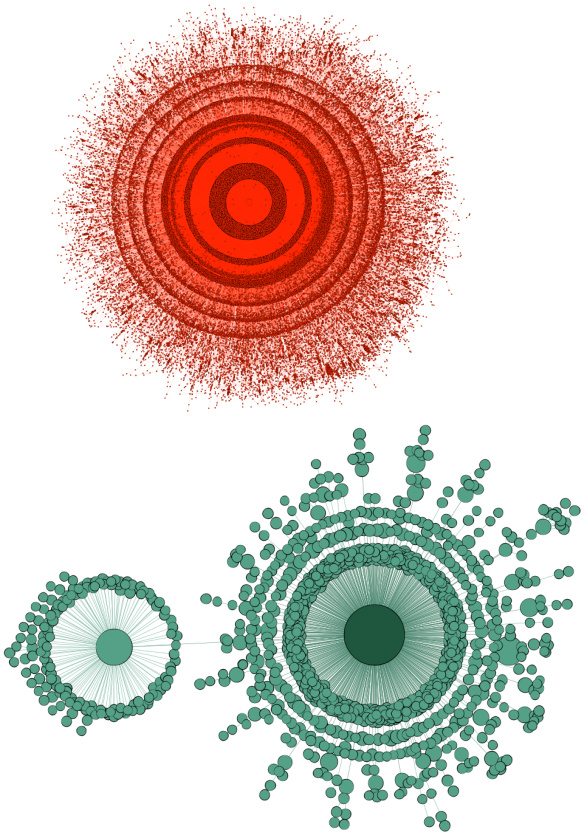
- **Challenge:**

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

Exact Shortest Path Distance

- **Challenge:**

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



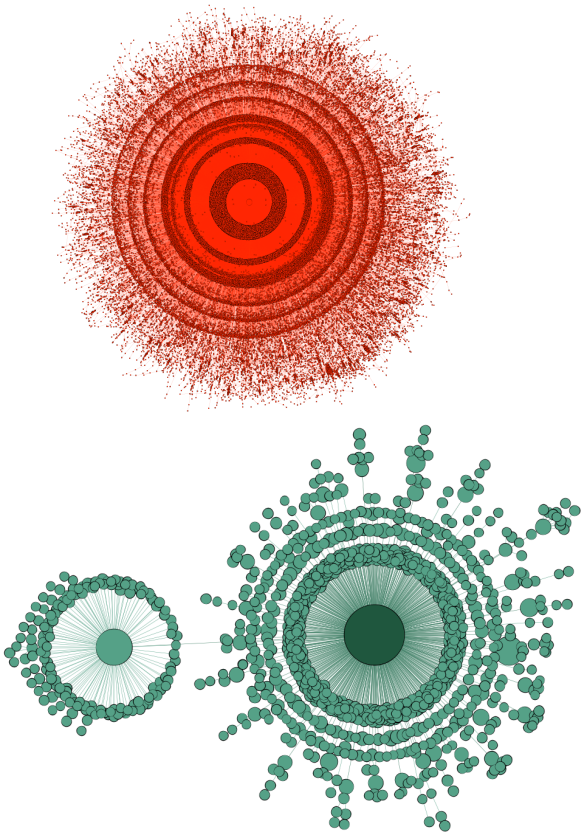
(two complex networks)

- *What kinds of networks?*
  - Complex networks

Exact Shortest Path Distance

- **Challenge:**

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



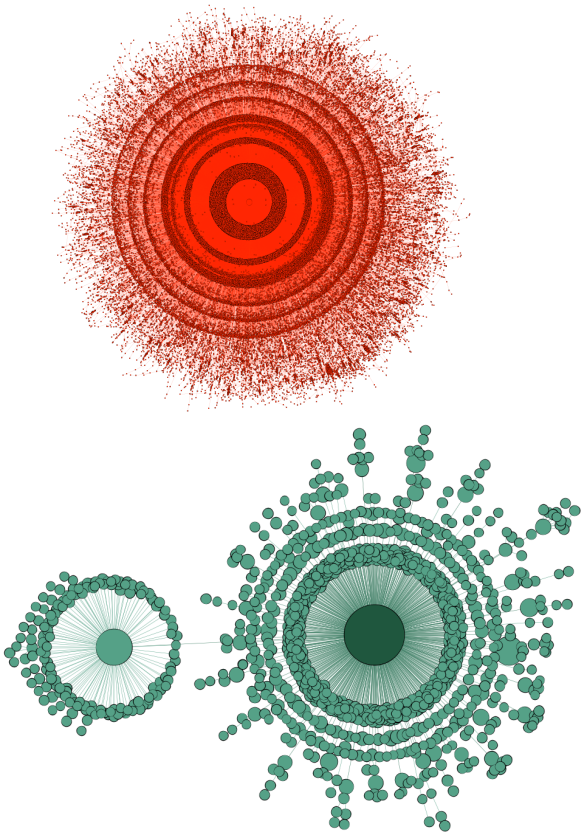
(two complex networks)

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

Exact Shortest Path Distance

- **Challenge:**

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



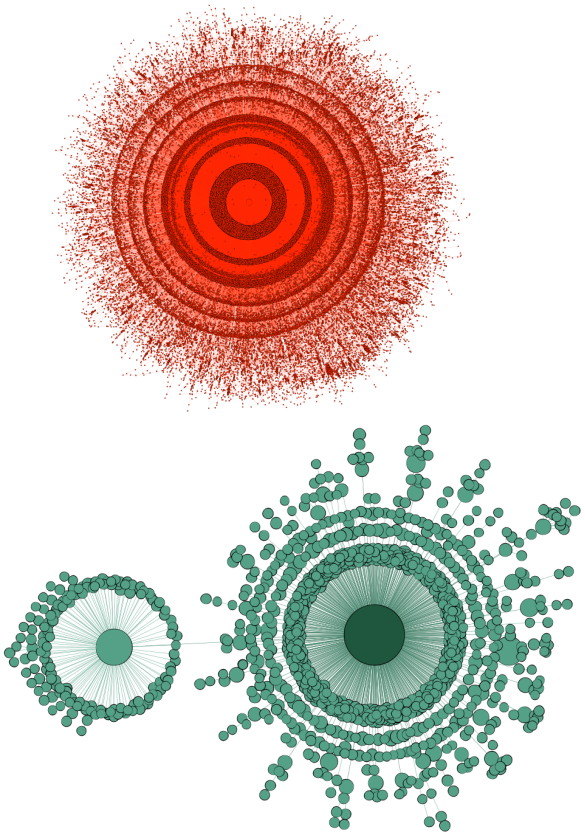
(two complex networks)

- *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

Exact Shortest Path Distance

- **Challenge:**

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

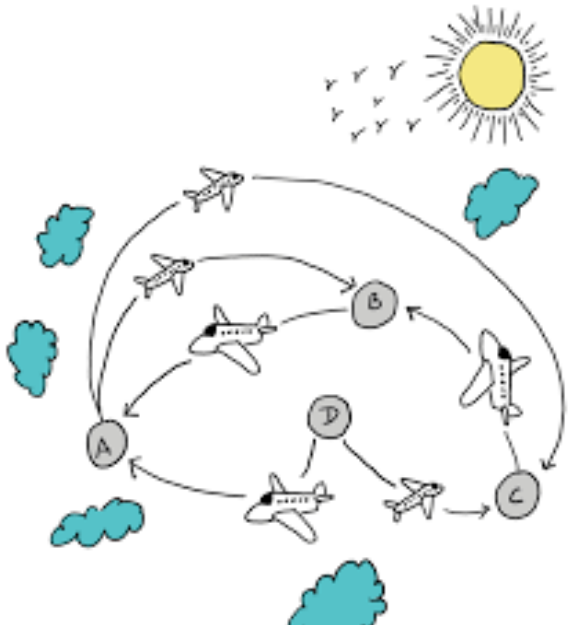


(two complex networks)

- *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
- *Computational resources?*
  - Scalable construction time (of indexing)
  - Scalable labelling size

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
- **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] – outperforming Impoved TEDI and HHL

Related Work

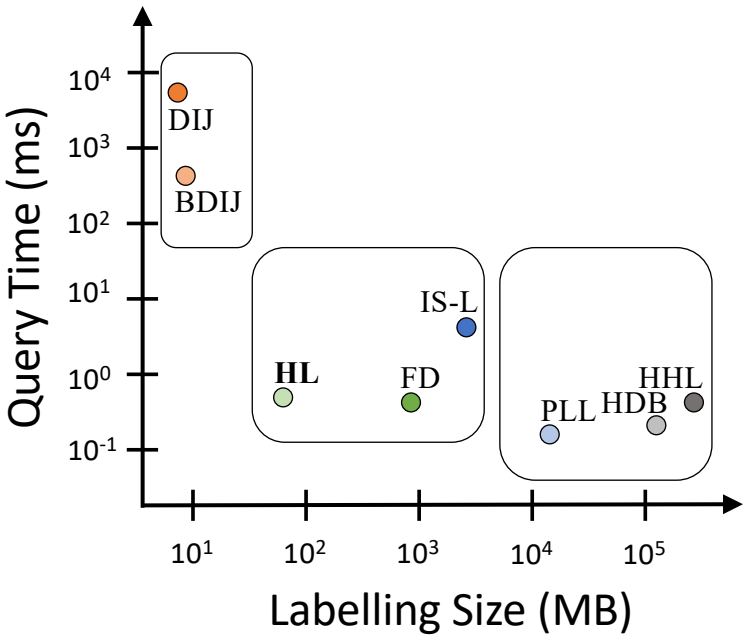
- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
- **Labelling-based approaches**
  - TreE Decomposition based Indexing (TEDI) method [SIGMOD2010]
  - Improved TEDI method [EDBT2012]
  - Hierarchical Hub-Labeling (HHL) algorithm [ESA2012]
  - Pruned Landmark Labeling (PLL) algorithm [SIGMOD2013] – outperforming Impoved TEDI and HHL
- **Hybrid approaches**
  - IS-Label (IS-L) method [VLDB2013] – outperforming HCL and TEDI
  - Fully Dynamic (FD) method [CIKM2016] – outperforming PLL, HDB, RXL and CRXL

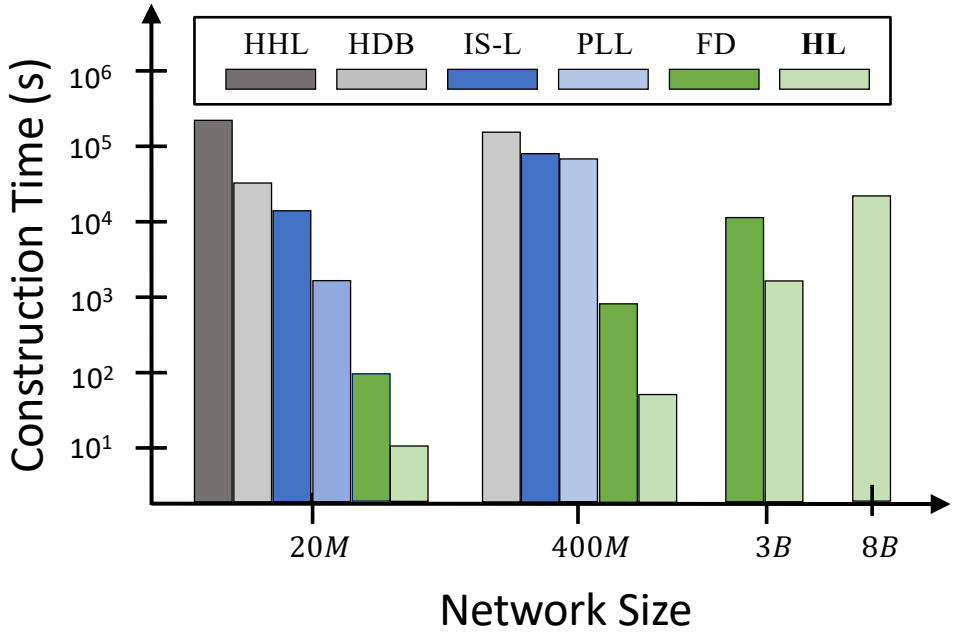
Related Work

- **Search-based approaches**
  - Dijkstra's algorithm
  - BFS or Bidirectional BFS
- **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] – outperforming Impoved TEDI and HHL
- **Hybrid approaches**
  - IS-Label (IS-L) method [VLDB2013] – outperforming HCL and TEDI
  - Fully Dynamic (FD) method [CIKM2016] – outperforming PLL, HDB, RXL and CRXL

No work can handle graphs with billions of vertices and billions of edges.

A High-level Overview





(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.

Our Method - HL

Our method proceeds in two steps:

1. A highly scalable labelling algorithm for constructing the distance labelling using a highway:
   - Relax the 2-hop cover property with a novel property, called the highway cover property
   - Enjoy some other nice properties: order independence, minimality of labelling, and parallel construction
2. A querying framework that supports fast distance bounded searches on a sparsified graph

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.,

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

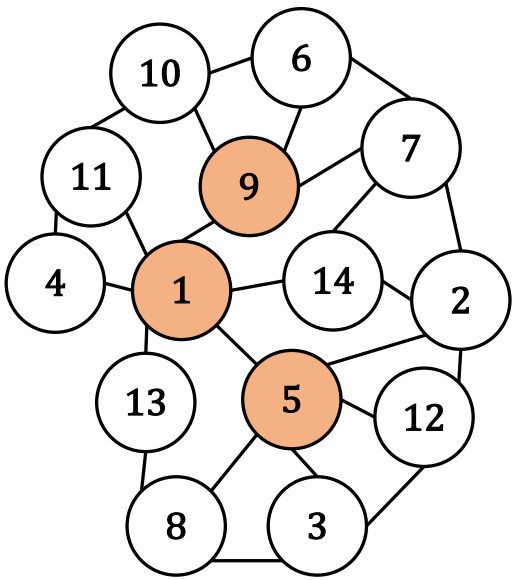
where $r_i \in R$ and $\delta(r_i, v) = d_G(r_i, v)$.

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 $\delta(r_i, v) = d_G(r_i, v)$.



$R = \{1, 5, 9\}$

$L(1)$

Landmark	5	9
Distance	1	1

$L(2)$

Landmark	1	5	9
Distance	2	1	2

$d_G(5, 2) = 1$

$L(3)$

Landmark	1	5	9
Distance	2	1	3

.....

2-Hop Labelling

- **2-Hop cover property**

  For any two vertices $s, t \in V$, there exists at least one landmark $r \in R$ in both $L(s)$ and $L(t)$, which is on the shortest path between s and t.

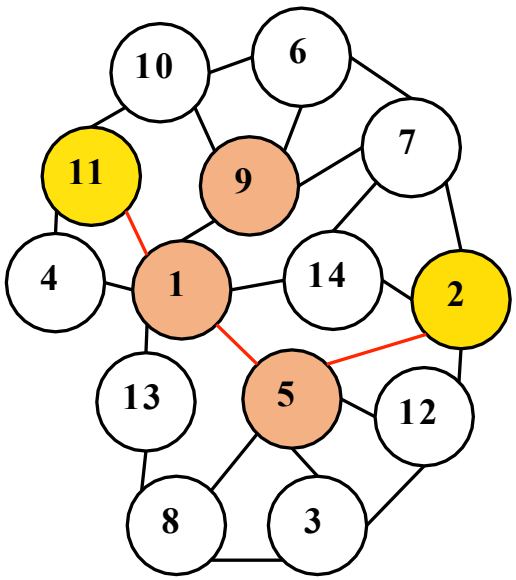
$$d_G(s,t) = \min_{r \in L(s) \cap L(t)} \{\delta(r,s) + \delta(r,t)\}$$

2-Hop Labelling

- **2-Hop cover property**

  For any two vertices $s, t \in V$, there exists at least one landmark $r \in R$ in both $L(s)$ and $L(t)$, which is on the shortest path between s and t.

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



$R = \{1, 5, 9\}$

$L(2)$

Landmark	1	5	9
Distance	2	1	2

$L(11)$

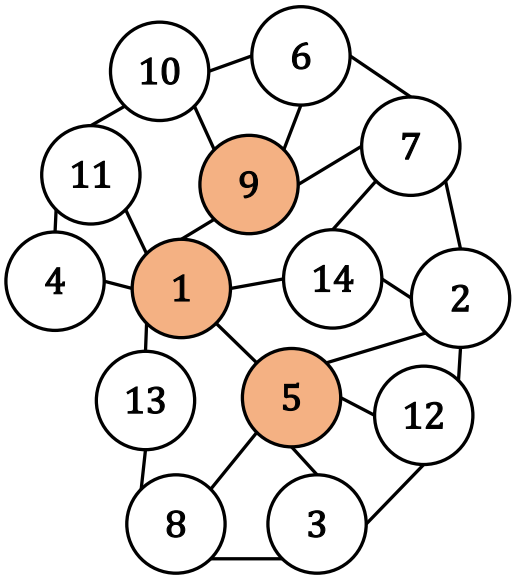
Landmark	1
Distance	1

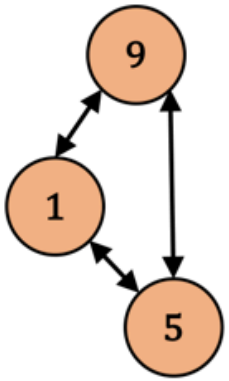
$$d_G(2,11) = d_G(1,2) + d_G(1,11) = 3$$

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

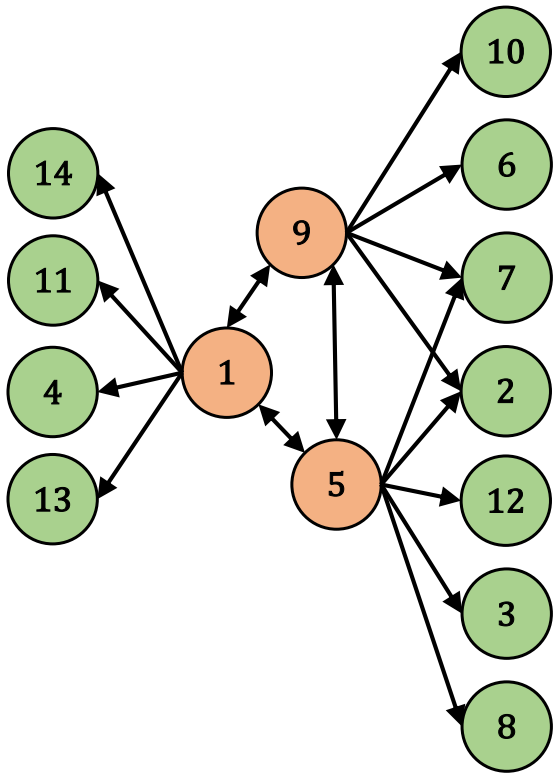




Highway Cover Labelling

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 the shortest path between u and r (r and r' may be the same).



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)

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.

Highway Cover Labelling

Definition (Highway Cover Labelling Problem)

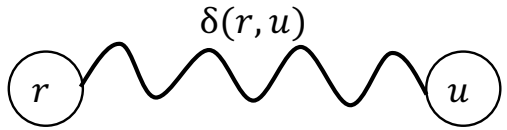
Given a graph G and a highway H over G, the *highway cover labelling problem* is to efficiently construct a highway cover labelling L.

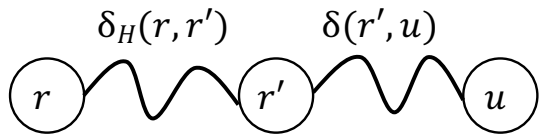
- Some properties of labelling methods:

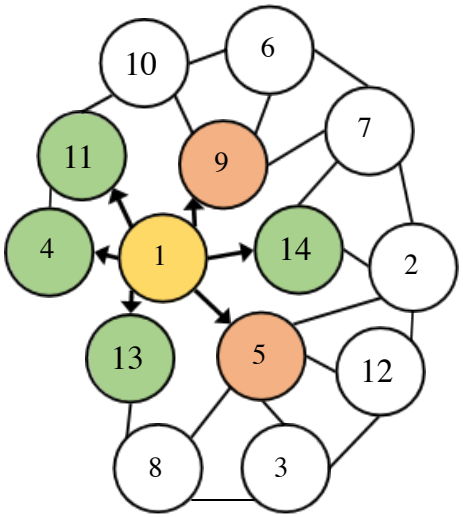
Methods	Ordering dependent?	Highway Cover minimal?	Parallel?
HL (ours)	**no**	**yes**	**landmarks**
FD [CIKM2016]	no	no	neighbours
IS-L [VLDB2013]	yes	no	no
PLL [SIGMOD2013]	yes	no	neighbours
HDB [VLDB2014]	yes	no	no
HHL [ESA2012]	yes	no	no

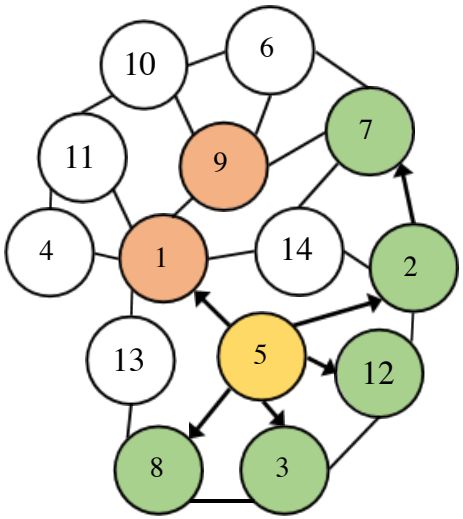
Highway Cover Labelling: Our Algorithm

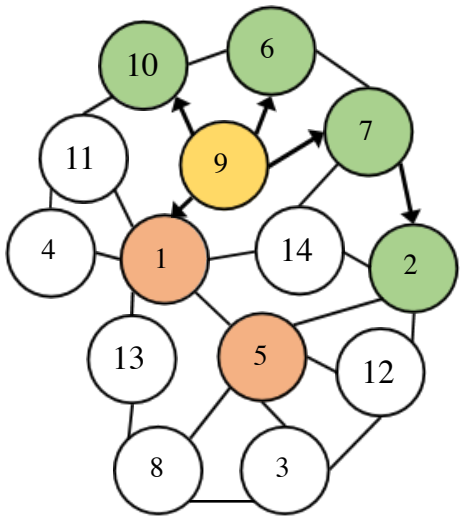
1. Conduct a BFS from each landmark $r \in R$
   - Add a distance entry $(r, \delta(r, v))$ into the label of $v \in V \setminus R$ iff there does not exist any other landmark $r' \in R$ that appears in the shortest path between r and u.











Order Independence

Lemma

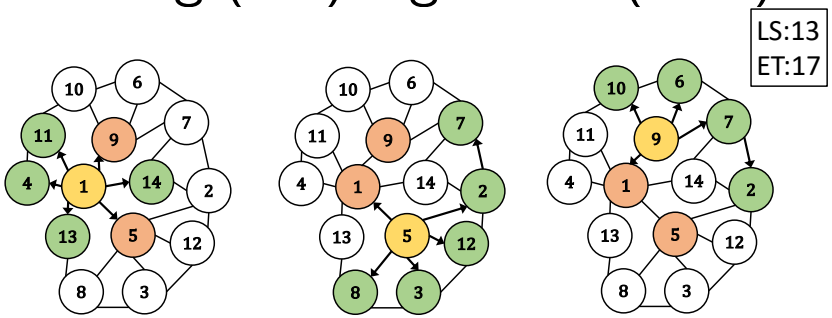
Let L_1 and L_2 be the highway cover labellings constructed by our HL Algorithm using two different labelling orders, then $L_1(v) = L_2(v)$ for every $v \in V \backslash R$.

Order Independence

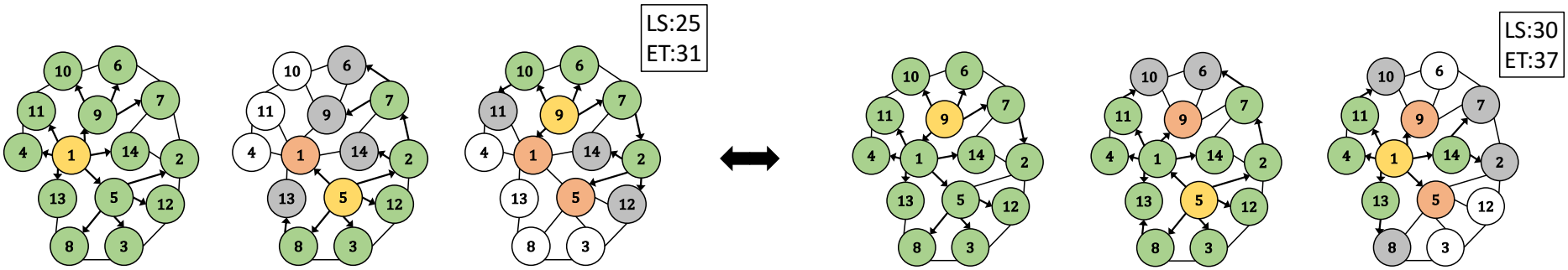
Lemma

Let L_1 and L_2 be the highway cover labellings constructed by our HL Algorithm using two different labelling orders, then $L_1(v) = L_2(v)$ for every $v \in V \setminus R$.

- Highway cover Labelling (HL) algorithm (ours)



- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]



Minimality

Theorem

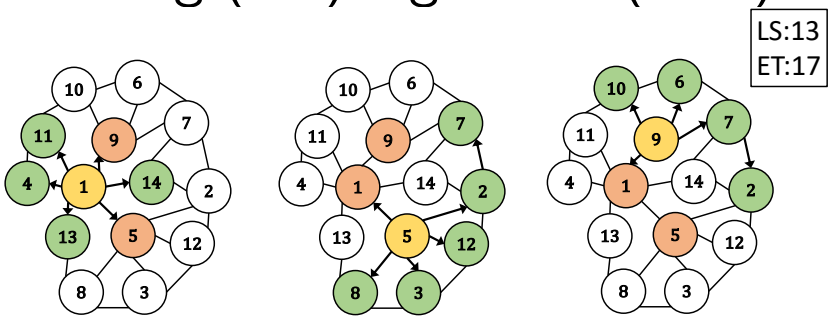
The highway cover labelling L over (G, H) constructed by our algorithm is minimal, i.e., for any highway cover distance labelling L' over (G, H), $size(L') \geq size(L)$ must hold.

Minimality

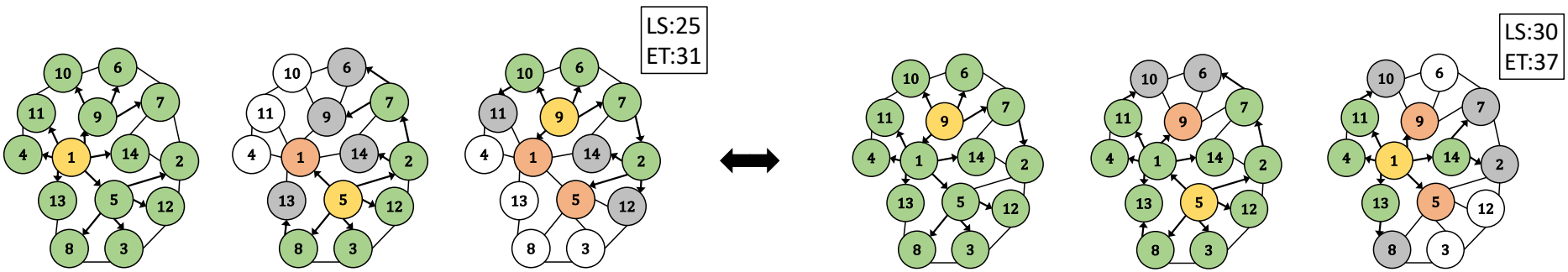
Theorem

The highway cover labelling L over (G, H) constructed by our algorithm is minimal, i.e., for any highway cover distance labelling L' over (G, H), $size(L') \geq size(L)$ must hold.

- Highway cover Labelling (HL) algorithm (ours)



- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]



Parallelism

- Highway cover Labelling (HL) algorithm (ours)
  - **Landmark Parallelism** (LP):
    Run parallel BFSs from multiple landmarks (depending on the number of processors) to construct labelling in an extremely efficient way for massive networks

- Pruned Landmark Labelling (PLL) algorithm [SIGMOD2013]
  - **Bit-Parallelism** (BP):
    Perform BFSs from a given landmark r and up to 64 of its neighbors simultaneously, and encode the relative distances (-1, 0 or 1) of these neighbors w.r.t. the shortest paths between r and each vertex v into a 64-bit unsigned integer.

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 $G[V \backslash R]$.

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

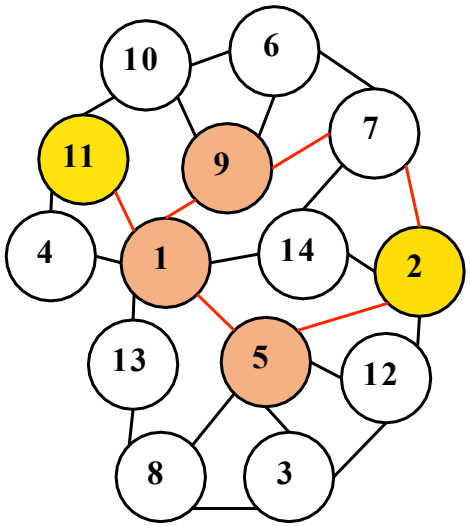
- Find a path of the minimal length through a highway H:

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

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=2 and t=11?

$L(2)$

Landmark	5	9
Distance	1	2

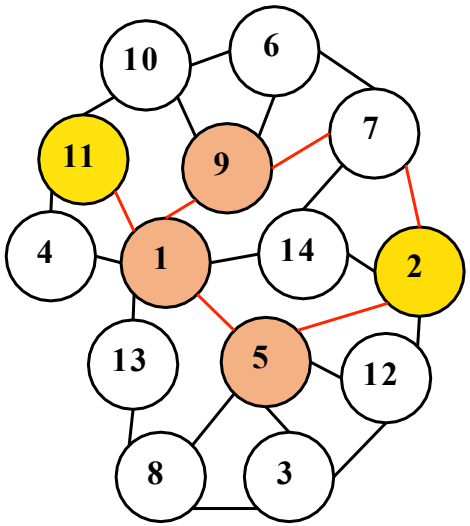
$L(11)$

Landmark	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=2 and t=11?

$L(2)$

Landmark	5	9
Distance	1	2

$L(11)$

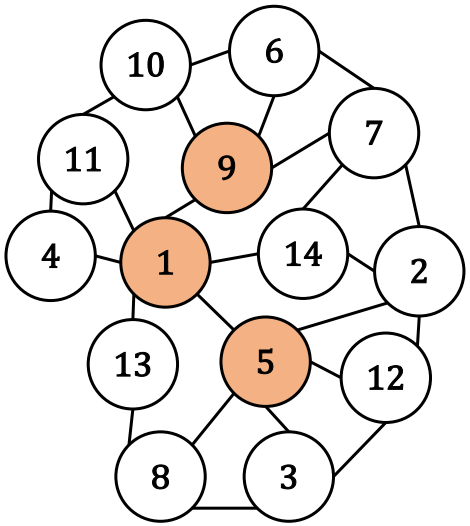
Landmark	1
Distance	1

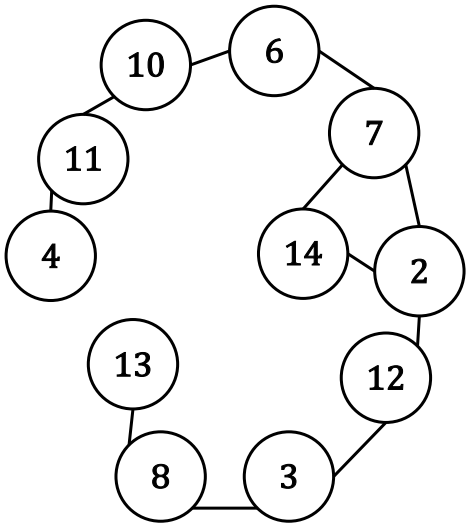
- $2 \to 5 \to 1 \to 11$: length 3
- $2 \to 9 \to 1 \to 11$: length 4

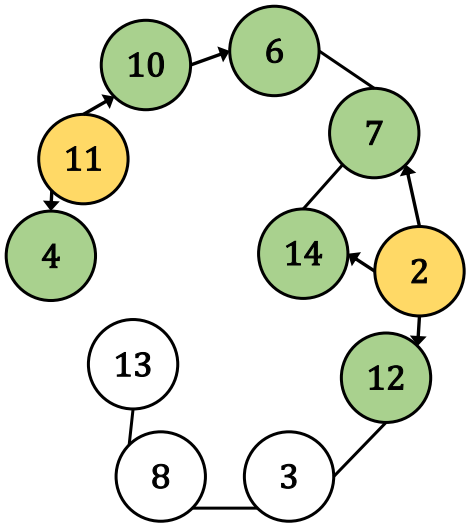
$$d_{st}^{\top} = 3$$

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 $d_{st}^{\top}$







$$d_G(2, 11) = 3$$

Experiments

- **12 large-scale real-world networks**
  - Complex networks
  - Undirected and unweighted graphs
  - Sizes ranging from $|V|$=1.7M & $|E|$=11M to $|V|$=2B & $|E|$=8B
  - Various domains: social networks, web networks, etc.
- **Evaluation measures**
  - Construction Time of Indexing (CT)
  - Query Time (QT)
  - Labelling Size (LS)
- **Distance queries**
  - Randomly sample 100,000 pairs of vertices from all pairs of vertices in each network

Construction Time (sec.)

Dataset	|V|	|E|	HL-P (this work)	HL (this work)	FD (Hayashi+'16)	PLL (Akiba+'13)	IS-L (Fu+'13)
Skitter	1.7M	11M	**2**	**13**	30	638	1042
Flickr	1.7M	16M	**2**	**14**	41	1330	8359
Hollywood	1.1M	114M	**3**	**17**	107	31855	DNF
Orkut	3.1M	117M	**10**	**62**	366	DNF	DNF
enwiki2013	4.2M	101M	**9**	**77**	308	22080	DNF
LiveJournal	4.8M	69M	**9**	**77**	166	DNF	20583
Indochina	7.4M	194M	**8**	**50**	144	9456	DNF
it2004	41M	1.2B	**66**	**304**	1623	DNF	DNF
Twitter	42M	1.5B	**133**	**1380**	1838	DNF	DNF
Friendster	66M	1.8B	**135**	**2229**	9661	DNF	DNF
uk2007	106M	3.7B	**110**	**1124**	6201	DNF	DNF
ClueWeb09	2B	8B	**4236**	**28124**	DNF	DNF	DNF

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

Query Time (ms)

Dataset	|V|	|E|	HL (this work)	FD (Hayashi+'16)	PLL (Akiba+'13)	IS-L (Fu+'13)
Skitter	1.7M	11M	**0.067**	0.043	0.008	3.556
Flickr	1.7M	16M	**0.015**	0.028	0.01	33.760
Hollywood	1.1M	114M	**0.047**	0.075	0.051	-
Orkut	3.1M	117M	**0.224**	0.251	-	-
enwiki2013	4.2M	101M	**0.190**	0.131	0.027	-
LiveJournal	4.8M	69M	**0.088**	0.111	-	56.847
Indochina	7.4M	194M	**1.905**	1.803	0.02	-
it2004	41M	1.2B	**2.684**	2.118	-	-
Twitter	42M	1.5B	**1.424**	0.432	-	-
Friendster	66M	1.8B	**1.091**	1.435	-	-
uk2007	106M	3.7B	**11.841**	18.979	-	-
ClueWeb09	2B	8B	**0.309**	-	-	-

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

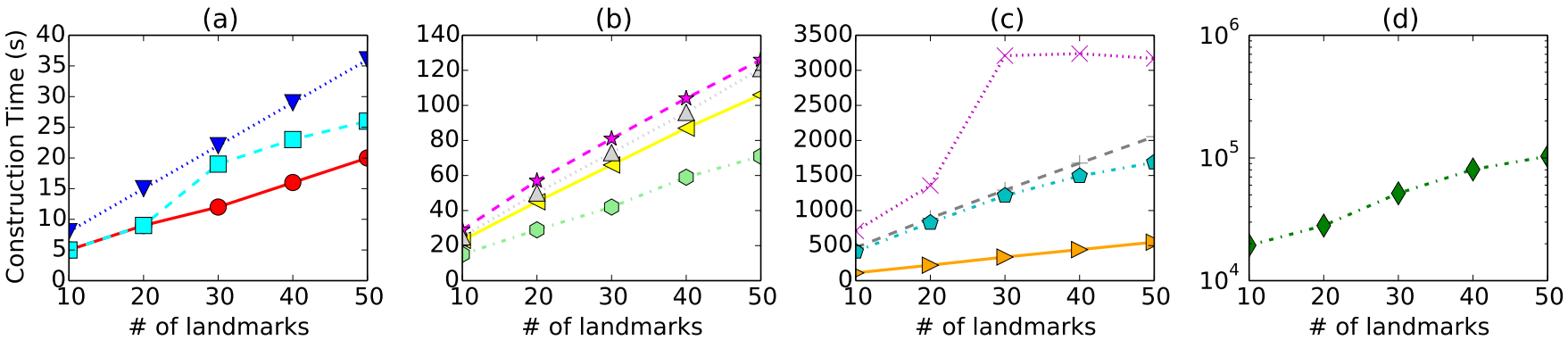
Labelling Size

Dataset	|V|	|E|	HL(8) (this work)	HL (this work)	FD (Hayashi+'16)	PLL (Akiba+'13)	IS-L (Fu+'13)
Skitter	1.7M	11M	**42MB**	**102MB**	202MB	2.5	507MB
Flickr	1.7M	16M	**34MB**	**81MB**	178MB	3.7GB	679MB
Hollywood	1.1M	114M	**28MB**	**67MB**	293MB	13GB	-
Orkut	3.1M	117M	**70MB**	**170MB**	756MB	-	-
enwiki2013	4.2M	101M	**83MB**	**200MB**	743MB	12GB	-
LiveJournal	4.8M	69M	**123MB**	**299MB**	778MB	-	3.8GB
Indochina	7.4M	194M	**81MB**	**192MB**	999MB	21GB	-
it2004	41M	1.2B	**855MB**	**2GB**	5.6GB	-	-
Twitter	42M	1.5B	**1.2GB**	**2.8GB**	4.8GB	-	-
Friendster	66M	1.8B	**2.5GB**	**5.2GB**	11.8GB	-	-
uk2007	106M	3.7B	**1.8GB**	**4.3GB**	14.1GB	-	-
ClueWeb09	2B	8B	**4.7GB**	**9GB**	-	-	-

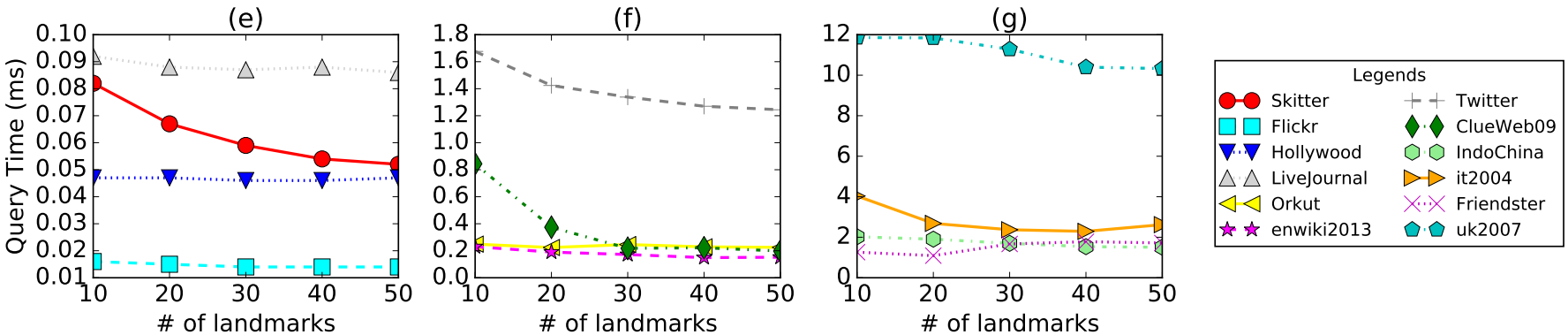
- Much smaller (may save up to 90% of space)

Performance under varying Landmarks

- Construction time using our method HL under varying landmarks



- Query time using our method HL under varying landmarks



Conclusion

- **Main contributions**

  A highly scalable labelling algorithm that can scale to billion-scale graphs, which also has several nice properties:

  (1) Minimal labelling

  (2) Order independence

  (3) Parallel construction

Conclusion

- **Main contributions**

  A highly scalable labelling algorithm that can scale to billion-scale graphs, which also has several nice properties:

  (1) Minimal labelling

  (2) Order independence

  (3) Parallel construction

- **Future works**
  - Dynamic graphs
  - Landmark selection strategies