BatchHL: Answering Distance Queries on Batch-Dynamic Networks at Scale

Muhammad Farhan ¹ Qing Wang ¹ Henning Koehler ²

¹School of Computing, Australian National University
²School of Mathematical and Computational Sciences, Massey University
Problem: Let $G \leftrightarrow G'$ denote that $G$ is changed to $G'$ by a batch update, to answer the shortest path distance $d_{G'}(s, t)$. 
Problem: Let $G \leftrightarrow G'$ denote that $G$ is changed to $G'$ by a batch update, to answer the shortest path distance $d_{G'}(s, t)$.
**Problem:** Let $G \xrightarrow{} G'$ denote that $G$ is changed to $G'$ by a *batch update*, to answer the shortest path distance $d_{G'}(s, t)$.

![Graphs $G$ and $G'$ with edge deletions and insertions](image)

$G$

- $d_G(3, 7) = 5$

$G'$

- $d_{G'}(3, 7) = 6$

**Batch Update**

- (11, 12) removed
- (1, 2) added
- (5, 6) removed
- (8, 10) added
Applications

World Wide Web Networks

Social Networks

Road Networks

Context-Aware Web Search

Socially-Sensitive Search

Social Network Analysis
  centrality, community search

Route Navigation
## Existing Approaches

### Dynamic Methods

| Search-based Methods | Dijkstra’s Search or Bidirectional Dijkstra’s Search |

| Query Time | Storage Memory |
|-------------|----------------|

- Searching

---

Muhammad Farhan  | Distance Queries on Batch-Dynamic Graphs  | 5 / 23
Existing Approaches

**Dynamic Methods**

- **Search-based Methods**
  - Dijkstra's Search or Bidirectional Dijkstra's Search
  - Decremental Method (DECM) (Qin+, WWW'17)
  - Dynamic 2-Hop Cover (FuPLL) (D’angelo+, JEA’19)
  - Maintenance of 2-Hop cover (DWPSL) (Zhang+, ICDE’21)

- **Labelling-based Methods**
  - Slow querying

---

**Diagram:**

- **Query Time**
  - Searching
  - Labelling

- **Storage Memory**
Existing Approaches

**Dynamic Methods**

**Search-based Methods**
- Dijkstra's Search or Bidirectional Dijkstra's Search
- Decremental Method (DECM) (Qin+, WWW'17)
- Dynamic 2-Hop Cover (FulPLL)
  - (D’angelo +, JEA'19)
- Maintenance of 2-Hop cover (DWPSL) (Zhang+, ICDE’21)

**Labelling-based Methods**
- Slow querying
- Limited Scalability
- Slow update time
- No minimality for Incremental updates
  - (Zhang+, ICDE’21)

**Hybrid Methods**
- Fully Dynamic (FD)
  - (Hayashi+, CIKM’16)
- Maintenance of Highway cover (HL)
  - (Farhan+, JVLDB’2021)

---

- Searching
- Hybrid
- Labelling

---

Muhammad Farhan
Distance Queries on Batch-Dynamic Graphs
## Existing Approaches

### Dynamic Methods

#### Search-based Methods
- Slow querying

#### Labelling-based Methods
- Limited Scalability
- Slow update time
- No minimality for Incremental updates
  (Zhang+, ICDE’21)

#### Hybrid Methods
- process single update at a time
- Slow update time
  (Farhan+, JVLDB 2021)

### Hybrid Methods

- Maintenance of Highway cover (HL)
  (Farhan+, JVLDB 2021)

- Fully Dynamic (FD)
  (Hayashi+, CIKM’16)

### Labelling-based Methods

- Maintenance of 2-Hop cover (DWPSL)
  (Zhang+, ICDE’21)

- Dynamic 2-Hop Cover
  (FulPLL)
  (D’angelo +, JEA’19)

### Dynamic Methods

- Dijkstra’s Search or Bidirectional Dijkstra’s Search
- Decremental Method (DECM)
  (Qin+, WWW’17)

### Labelling

- Limited Scalability
- Slow update time
- No minimality for Incremental updates

### Batch-Dynamic Highway Cover Labelling

(Our Work)

### Graph Representation

- Searching
- Hybrid
- Labelling

### Diagram

- Query Time
- Storage Memory

- Batch-Dynamic Highway Cover Labelling
  (Our Work)
Preliminaries

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

\[
\begin{array}{c|c|c}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 1 & 2 & 2 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 2 & 3 & 3 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 3 & 4 & 4 \\
\end{array}
\]

\[
\ldots \ldots \\
\]

\[
d_{G}(8,2) = 3
\]
**Distance Labelling:**

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

\[
R = \{5, 8, 10\}
\]

\[
\begin{align*}
L(1) & : \begin{array}{ccc}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 1 & 2 & 2 \\
\end{array} \\
L(2) & : \begin{array}{ccc}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 2 & 3 & 3 \\
\end{array} \\
L(3) & : \begin{array}{ccc}
\text{Landmark} & 5 & 8 & 10 \\
\text{Distance} & 3 & 4 & 4 \\
\end{array}
\end{align*}
\]

\( d_G(8, 2) = 3 \)
2-Hop Cover Labelling:
Distance through a common landmark in $L(s)$ and $L(t)$.

$$d_G(s, t) = \min_{r \in L(s) \cap L(t)} \left\{ \delta(r, s) + \delta(r, t) \right\}$$
2-Hop Cover Labelling:
Distance through a common landmark in \( L(s) \) and \( L(t) \).

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

\[ R = \{5, 8, 10\} \]

\[ L(3) \]
\[ \begin{array}{|c|c|}
\hline
\text{Landmark} & 5 \\
\hline
\text{Distance} & 3 \\
\hline
\end{array} \]

\[ L(7) \]
\[ \begin{array}{|c|c|c|}
\hline
\text{Landmark} & 5 & 10 \\
\hline
\text{Distance} & 2 & 2 \\
\hline
\end{array} \]

\[
d_G(3, 7) = d_G(3, 5) + d_G(7, 5) = 5
\]
Definition (Highway)

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

Let $G = (V, E)$ be a graph and $H = (R, \delta_H)$ a highway. For any vertex $u \in V \setminus R$ 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).
Two steps for answering $d_G(s, t)$ in a graph $G$:

(1) Computing an upper bound distance using the precomputed (offline) highway cover labelling;

(2) Computing $d_G(s, t)$ using the (online) bidirectional search over a sparsified graph $G[V \setminus R]$. 
Computing Upper Bounds

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

$$d_{st}^\top = \min \{ \delta(r_i, s) + \delta_H(r_i, r_j) + \delta(r_j, t) \}$$

What is $d_{st}^\top$ for $s = 3$ and $t = 7$?
Computing Upper Bounds

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

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

What is $d_{st}^\top$ for $s = 3$ and $t = 7$?

| Landmark | 5 | 10 |
|----------|---|----|
| Distance | 2 | 2 |

| Landmark | 5 |
|----------|---|
| Distance | 3 |
Computing Upper Bounds

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

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

What is $d^\top_{st}$ for $s = 3$ and $t = 7$?

L(7)

| Landmark | 5 | 10 |
|----------|---|----|
| Distance | 2 | 2  |

3 → 5 → 7: length 5

L(3)

| Landmark | 5 |
|----------|---|
| Distance | 3 |

3 → 5 → 10 → 7: length 6

$$d^\top_{st} = 5$$
Bidirectional Search

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

$$d_{G'}(3, 7) = 5$$
Aim:
Efficient maintenance of highway cover distance labelling for a very large dynamic network undergoing batches of updates?
Shortest Path Distance Queries

**Aim:**
Efficient maintenance of highway cover distance labelling for a very large dynamic network undergoing batches of updates?

![Changed Graph G'](image)

| Label | Entries |
|-------|---------|
| L(1)  | (5, 1)  |
| L(6)  | (5, 1) (10, 1) |
| L(7)  | (5, 2) (10,2) |
| L(9)  | (8, 1)  |
| ...   | ...     |

|   | 5 | 8 | 10 |
|---|---|---|----|
| 5 | 0 | 1 | 1  |
| 8 | 1 | 0 | 2  |
| 10| 1 | 2 | 0  |
Proposed Approach - A High-Level Overview

**Landmark**
- **Level Parallelism**

| Label Entries | α | β | γ |
|---------------|---|---|---|
| L(1)          | 5 | 8 | 10
| L(6)          | 5 | 0 | 1 |
| L(7)          | 8 | 1 | 2 |
| L(9)          | 10| 1 | 2 |

**Batch Search**

**Batch Repair**

\[ \Gamma = (L, H) \]

\[ \Gamma' = (L', H') \]

**Landmark-Composite Path Affected**

**Landmark-Distance Affected**

**Repaired**

\[ \mathcal{G} \]

\[ \mathcal{G}' \]

Distance Queries on Batch-Dynamic Graphs
Batch Search

- Unify an edge insertion and deletion \((a, b)\) into a single process
  - both share the same pattern for affected vertices

\[
\begin{align*}
&d_{G(r,v)} \\
&d_{G(r,a)} + 1 \\
d_{G(u,v)}
\end{align*}
\]

Anchor vertex

\((+)\) Inserted edges
\((-)\) Deleted edges
Batch Search

- Unify an edge insertion and deletion \((a, b)\) into a single process
  - both share the same pattern for affected vertices

- Searches for vertices affected by edge insertions and deletions in a batch combine in a single search

\[
d_G(r, v) \quad d_G(r, a) + 1
\]

\[
d_G(u, v)
\]

Anchor vertex

(+). Inserted, (-). Deleted

(+) Inserted, (-) Deleted

(f, v) is traversed only once!
Batch Search

- Unify an edge insertion and deletion \((a, b)\) into a single process
  - both share the same pattern for affected vertices

\[
d_G(r,v) \quad d_G(r,a) + 1
\]

- Searches for vertices affected by edge insertions and deletions in a batch combine in a single search

\((+)\) Inserted, \((-)\) Deleted

Insertion of \((a, d)\)
Batch Search

- Unify an edge insertion and deletion $(a, b)$ into a single process
  - both share the same pattern for affected vertices

- Searches for vertices affected by edge insertions and deletions in a batch combine in a single search

$$d_G(r,v) \quad d_G(r,a)+1$$

$$d_G(u,v)$$

$\rightarrow$ Anchor vertex

(+ Inserted, (-) Deleted

(+) Inserted edges
(-) Deleted edges

Insertion of $(a, d)$

Insertion of $(d, f)$

$(f, v)$ is traversed only once!
Batch Search

- Unify an edge insertion and deletion \((a, b)\) into a single process
  - both share the same pattern for affected vertices

\[
\begin{align*}
\delta_G(r, v) & \quad \delta_G(r, a) + 1 \\
\delta_G(u, v) & \quad \text{Anchor vertex}
\end{align*}
\]

- Searches for vertices affected by edge insertions and deletions in a batch combine in a single search

\((-+)\) Inserted, \((-)\) Deleted
Insertion of \((a, d)\)
Insertion of \((d, f)\)
Deletion of \((b, e)\)

(f, v) is traversed only once!
Improved Batch Search

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

Case 1: no change

Case 2: change
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.

**Extended Landmark Length**

\[ \text{Extended Landmark Length} \leq \text{Landmark Length} \]

**Addition Case - Pruning**

\[ (2, F, F) \]
\[ (2, F, T) \]

\[ (2, T, F) \]
\[ (2, T, T) \]

\[ (2, T, F) \]
\[ (2, T, T) \]

**Deletion Case - Pruning**

\[ (2, F, T) \]
\[ (2, F, T) \]

\[ (2, T, F) \]
\[ (2, T, T) \]

\[ (2, T, F) \]
\[ (2, T, T) \]
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 returned by batch search

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

$r - f - e$ is a composite path

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

\[
L = \begin{array}{cccccccc}
& a & b & c & d & e & f & g & h & i \\
(r, 1) & (r, 1) & (r, 1) & (r, 2) & (r, 1) & (r, 2) & (r, 1) & (r, 2) & (r, 1) & (r, 2) \\
(r', 1) & (r', 1) & (r', 1) & (r', 2) & (r', 2) & (r', 1) & (r', 2) & (r', 2) & (r', 2) & (r', 2) \\
\end{array}
\]
Batch Repair

- Repair the labels of vertices returned by batch search

![Diagram](image)

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

\[
L = 
\begin{array}{cccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\
(r, 1) & (r, 1) & (r, 1) & (r, 1) & (r, 2) & (r, 2) & (r, 2) & (r, 3) & (r, 2) \\
\end{array}
\]

Basic batch search algorithm returns \{r', d, e, f, g, h, i\}
• Repair the labels of vertices returned by batch search

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

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

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

\[
L = \begin{array}{cccccccc}
|   & a & b & c & d & e & f & g & h & i \\
---&---&---&---&---&---&---&---&---&---
(r, 1) & (r, 1) & (r', 1) & (r', 1) & (r, 2) & (r, 1) & (r, 2) & (r, 3) & (r', 1) & (r', 2)
\end{array}
\]
Repair the labels of vertices returned by batch search

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

$L = \begin{array}{cccccccc}
    & a & b & c & d & e & f & g & h & i \\
\hline
    r & (r, 1) & (r, 1) & (r', 1) & (r', 1) & (r, 2) & (r, 1) & (r, 2) & (r, 3) & (r', 2) \\
    r' & (r', 1) & (r', 2) & (r', 1) & (r', 1) & (r', 2) & (r', 2) & (r', 2) & (r', 2) & (r', 2) \\
\end{array}$

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

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

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

$L =
\begin{array}{cccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\
\hline
(r, 1) & (r, 1) & (r, 1) & (r, 2) & (r, 1) & (r, 2) & (r, 3) & (r', 1) & (r', 2) \\
\end{array}$
Batch Repair

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

\[
L = \begin{array}{cccccccc}
  a & b & c & d & e & f & g & h & i \\
  (r, 1) & (r, 1) & (r, 1) & (r', 1) & (r, 2) & (r, 1) & (r, 2) & (r, 3) & (r', 2)
\end{array}
\]

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

\[
L = \begin{array}{cccccccc}
  a & b & c & d & e & f & g & h & i \\
  (r, 1) & (r, 1) & (r, 1) & (r', 1) & (r, 2) & (r, 3) & (r, 2) & (r, 3) & (r', 2)
\end{array}
\]
Batch Repair

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

\[
L = \begin{array}{cccccccc}
  a & b & c & d & e & f & g & h & i \\
\text{(r, 1)} & (r, 1) & (r, 1) & (r', 1) & (r, 2) & (r, 1) & (r', 2) & (r', 1) & (r', 2) \\
\end{array}
\]

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

\[
L = \begin{array}{cccccccc}
  a & b & c & d & e & f & g & h & i \\
\text{(r, 1)} & (r, 1) & (r, 1) & (r', 1) & (r, 2) & (r, 1) & (r', 2) & (r', 1) & (r', 2) \\
\end{array}
\]
Batch Repair

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

$L = \begin{array}{cccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\
(r, 1) & (r, 1) & (r, 1) & (r', 1) & (r, 2) & (r, 2) & (r, 3) & (r', 2) & (r', 2)
\end{array}$

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

$L = \begin{array}{cccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\
(r, 1) & (r, 1) & (r, 1) & (r', 2) & (r, 2) & (r, 3) & (r', 1) & (r', 2) & (r', 2)
\end{array}$
### 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 | 9.110  | 0.003    | 0.008    | 0.048   | 0.154 | 0.194 | 0.070    | 0.169    | 0.239   | 3.181 | 9.850 |
| Skitter   | 1.7M | 11M  | 0.147    | 0.601    | 0.902 | 1.587   | 5.986 | 8.770  | 0.002    | 0.006    | 0.069   | 0.117 | 1.312 | 0.163    | 0.751    | 2.382   | 14.15 | 31.50 |
| Flickr    | 1.7M | 16M  | 0.024    | 0.026    | 0.130 | 0.099   | 2.152 | 6.300  | 0.003    | 0.008    | 0.072   | 0.053 | 1.259 | 0.030    | 0.041    | 0.107   | 3.364 | 13.40 |
| Wikitalk  | 2.4M | 5M   | 0.029    | 0.025    | 0.101 | 0.134   | 2.926 | 4.550  | 0.002    | 0.005    | 0.097   | 0.029 | 0.081 | 0.046    | 0.044    | 0.147   | 5.674 | 9.820 |
| 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 | 2.193   | 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 | -      |
| Italianwiki| 1.2M | 35M  | 0.001    | 0.001    | 0.025 | 0.051   | 6.623 | -      | -       | -       | -      | -     | -      | -       | -       | -      | -     | -      |
| Frenchwiki| 2.2M | 59M  | 0.003    | 0.004    | 0.067 | 0.098   | 5.289 | -      | -       | -       | -      | -     | -      | -       | -       | -      | -     | -      |

- Much improved update time for all three settings.
- Decremental batch updates are much more faster.
### 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  | 9.110  | 0.003  | 0.008  | 0.048  | 0.154  | 0.194  | 0.070  | 0.169  | 0.239  | 3.181  | 9.850  |
| Skitter   | 1.7M | 11M | 0.147  | 0.601  | 0.902  | 1.587  | 5.986  | 8.770  | 0.002  | 0.006  | 0.069  | 0.117  | 1.312  | 0.163  | 0.751  | 2.382  | 14.15  | 31.50  |
| Flickr    | 1.7M | 16M | 0.024  | 0.026  | 0.130  | 0.099  | 2.152  | 6.300  | 0.003  | 0.008  | 0.072  | 0.053  | 1.259  | 0.030  | 0.041  | 0.107  | 3.364  | 13.40  |
| Wkitalk   | 2.4M | 5M  | 0.029  | 0.025  | 0.101  | 0.134  | 2.926  | 4.550  | 0.002  | 0.005  | 0.097  | 0.029  | 0.081  | 0.046  | 0.044  | 0.147  | 5.674  | 9.820  |
| 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  | -      |
| Italianwiki| 1.2M | 35M | 0.001  | 0.001  | 0.025  | 0.051  | 6.623  | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      |
| Frenchwiki| 2.2M | 59M | 0.003  | 0.004  | 0.067  | 0.098  | 5.289  | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      |

- Much improved update time for all three settings.
- Decremental batch updates are much more faster.
Comparison of the total time of querying and updating the labelling with the baseline methods.
Conclusion

Contributions
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 works
- Extension of the proposed approaches to road networks
- Selection of highly central vertices
- Guided search by investigating properties of the proposed approaches
Questions

Thank You