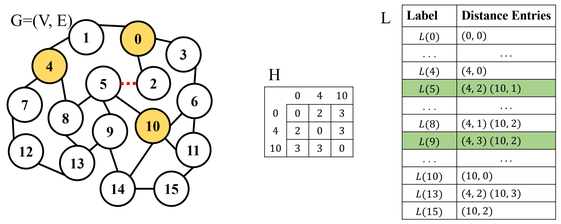
EFFICIENT MAINTENANCE OF DISTANCE LABELLING FOR INCREMENTAL UPDATES IN LARGE DYNAMIC GRAPHS

Muhammad Farhan and Qing Wang

School of Computing, Australian National University; {muhammad.farhan, qing.wang}@anu.edu.au

Introduction

- The shortest-path distance query is a fundamental problem in graph theory.
- Current research is limited to static graphs and suffers from:
  1) scaling to billion-scale networks;
  2) efficiently updating the labelling.
- In real-world applications, dynamic networks are more vulnerable to edge insertions than edge removals.
- Graph changes by edge insertions may result in overestimated distances.



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

Label	Distance Entries
$L(0)$	$(0, 0)$
...	...
$L(4)$	$(4, 0)$
$L(5)$	$(4, 2)$ $(10, 1)$
...	...
$L(8)$	$(4, 1)$ $(10, 2)$
$L(9)$	$(4, 3)$ $(10, 2)$
...	...
$L(10)$	$(10, 0)$
$L(13)$	$(4, 2)$ $(10, 3)$
$L(15)$	$(10, 2)$

Fig. 1: (left) A graph G with three landmarks (colored in yellow) and an inserted edge $(2, 5)$; (middle) A highway H over G; (right) A highway cover labelling L over G.

Preliminaries

Let $d_G(u, v)$ denote the length of the shortest path and $P_G(u, v)$ the set of all shortest paths between u and v in G.

Given a set of *landmarks* $R \subseteq V$,

1. *Distance labelling*: $L(v) = \{(r_1, \delta_L(r_1, v)), \ldots, (r_{|R|}, \delta_L(r_{|R|}, v))\}$ $\forall v \in V$, where $r_i \in R$ and $\delta_L(r_i, v) = d_G(r_i, v)$.
2. *Highway*: $H = (R, \delta_H)$ consisting of landmarks R and a distance decoding function $\delta_H : R \times R \to \mathbb{N}^+$ s.t. for any $r_1, r_2 \in R$, $\delta_H(r_1, r_2) = d_G(r_1, r_2)$.

Highway cover labelling. A *highway cover labelling* is a pair $\Gamma = (H, L)$ where H is a highway and L is a distance labelling s.t. for any $v \in V \backslash R$ and $r \in R$, we have:

$$d_G(r, v) = \texttt{min}\{\delta_L(r_i, v) + \delta_H(r, r_i) | (r_i, \delta_L(r_i, v)) \in L(v)\}. \quad (1)$$

Distance querying. Given a highway cover labeling $\Gamma = (H, L)$, an upper bound on the distance between two vertices $u, v \in V \backslash R$ is computed:

$$d^{\top}_{uv} = \texttt{min}\{\delta_L(r_i, u) + \delta_H(r_i, r_j) + \delta_L(r_j, v) | (r_i, \delta_L(r_i, u)) \in L(u), (r_j, \delta_L(r_j, v)) \in L(v)\} \quad (2)$$

An exact distance query $Q(u, v, \Gamma)$ is answered by conducting a distance-bounded shortest-path search over a sparsified graph $G[V \backslash R]$ (i.e., removing all landmarks in R from G) s.t.

$$Q(u, v, \Gamma) = \begin{cases} d_{G[V \backslash R]}(u, v) & \text{if } d_{G[V \backslash R]}(u, v) \leq d^{\top}_{uv}, \\ d^{\top}_{uv} & \text{otherwise.} \end{cases}$$

Problem Statement

Let $G \hookrightarrow G'$ denote that a graph G is changed to a graph G' by an edge insertion. The *dynamic distance querying* problem is, given any two vertices u and v in G', to efficiently compute $d_{G'}(u, v)$.

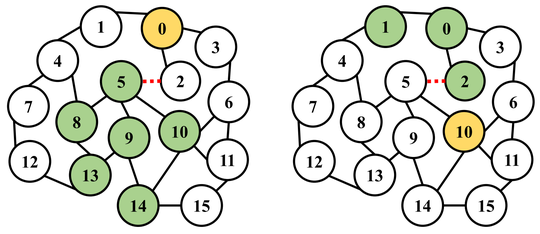
Online Incremental Algorithm

Our online incremental algorithm, called INCHL$^+$, incrementally updates labelling to reflect graph changes in two steps, as described below.

Step 1: Finding Affected Vertices

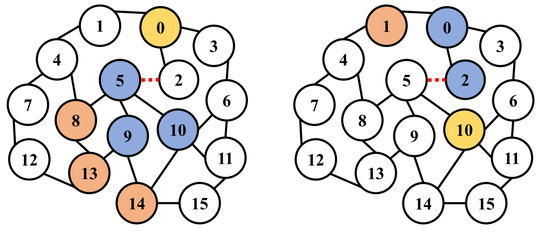
Affected vertex. A vertex $v \in V$ is *affected* by $G \hookrightarrow G'$ iff $P_G(v, r) \neq P_{G'}(v, r)$ for at least one $r \in R$; *unaffected* otherwise.

We conduct a partial BFS to identify affected vertices, e.g., below, partial BFSs starting from vertices 5 and 2 w.r.t. landmarks 0 and 10, respectively, where affected vertices are colored in green.



Step 2: Repairing Affected Vertices

We conduct another partial BFS to repair affected vertices, e.g., below, partial BFSs starting from vertices 5 and 2 w.r.t. landmarks 0 and 10, respectively, where repaired vertices with added/modified entries are colored in blue, and ones with removed entries are colored in red.



Two types of vertices are distinguished to improve efficiency:

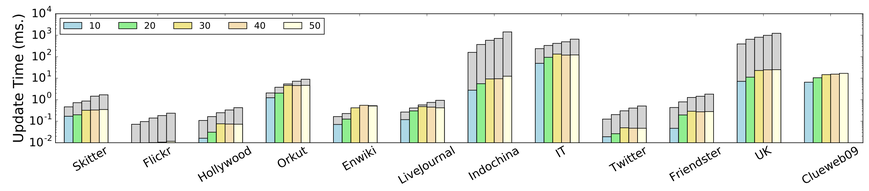
1. Affected vertices *covered* by other landmarks - repaired by removing an entry from their labels (if exists)
2. Affected vertices *not covered* by other landmarks - repaired by accurately calculating distances on a changed graph.

Experimental Results

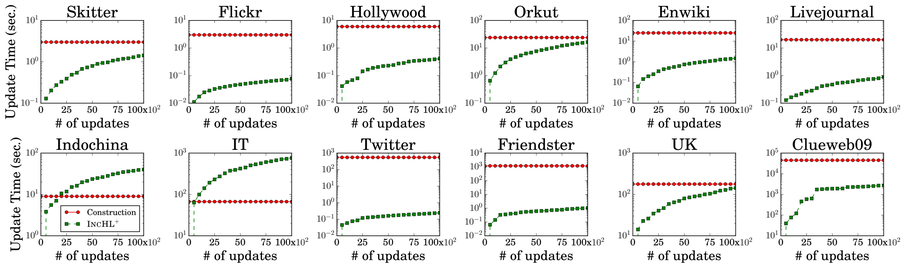
(Q1) How efficiently can our method perform against state-of-the-art methods? The average update time of our method INCHL$^+$ outperforms the state-of-the-art methods INCFD and INCPLL on all the datasets, while remaining significantly smaller labelling sizes and comparable query times.

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	10.53	-	-	16.93	-	-	163 GB	-	-

(Q2) How does the number of landmarks affect the performance of our method? INCHL$^+$ outperforms INCFD under varying landmarks and the performance gap remains stable for most of the datasets when increasing the number of landmarks.



(Q3) How does our method scale to perform updates occurring rapidly in large dynamic networks? The update time of INCHL$^+$ on almost all the datasets is considerably below the construction time of labelling. Note that the update time of our method depends on the fraction of vertices to be affected. Due to a high fraction of affected vertices, Indochina and IT took longer to update the labelling.



Conclusion

- Our method efficiently reflects graph changes in a highly scalable framework of answering distance queries and can scale to billion-scale dynamic graphs.
- Our method guarantees the preservation of minimality of labelling for dynamic graphs.