Online Appendix for the paper "Efficient Closeness Centrality Computation in Time-Evolving Graphs"

Peng Ni, Masatoshi Hanai, Wen Jun Tan, Wentong Cai

Appendix A Theorems and Proofs

Lemma A.1 *Given a source vertex v_s and a target vertex v_t, if the shortest distance between them $d(v_s, v_t)$ is k at time instance t, then $\exists$ a shortest path $p^* = \langle v_s = v_1, v_2, ..., v_k, v_{k+1} = v_t \rangle$, s.t. every prefix-subpath $p_m = \langle v_s = v_1, v_2, ..., v_m \rangle$, $m \leq k$ is a shortest path from v_s to v_m at time instance t.*

Proof A.1 *We prove Lemma A.1 by contradiction. Assume for the given shortest path p^*, some prefix-subpath $p_m = \langle v_s = v_1, v_2, ..., v_m \rangle$ is not a shortest path at time instance t. Let the shortest path from v_s to v_m be p'_m. Since it is shorter than p_m, if we take path p^* and replace subpath p_m with path p'_m, we get a shorter path from v_s to v_t compared to the original path p^*. This is contradictory to our assumption that p^* is a shortest path.*

Theorem A.1 *Algorithm 1 **ECC** correctly computes the distance labels $L[v]$ for each vertex $v \in V$ for each graph snapshot $G_s, s \in T$.*

Proof A.2 *Given $\forall$ vertex $v \in V$, if its distance to v_s is k at time instance t, there must exist a shortest path $p = \langle v_s = v_1, v_2, ..., v_k, v_{k+1} = v \rangle$, where $(v_i, v_{i+1}, \Lambda_i) \in E$ is the i-th edge on p for $1 \leq i \leq k$. Based on Lemma A.1, we can let p be the shortest path that each prefix subpath is also a shortest path.*

We prove that Algorithm 1 visits vertex v_i ($1 \leq i \leq k+1$) on the path and update its distance label $(i, [t, t])$ correctly at i-th BFS level. We prove it by induction. When $i = 1$, the source vertex is visited since it is the only vertex in the current queue. The distance label for the source vertex v_s is initialized to be 0 for all graph snapshots including G_t, which is correct. Assume when $i = m$, vertex v_m is visited and its label is updated to be $(m, [t, t])$. Based on line 13 of Algorithm 1, when its label is updated, vertex v_i will be inserted to the next queue. In the next iteration, which is $m+1$-th BFS traversal, when v_m is visited, all associated edges including $(v_m, v_{m+1}, \Lambda_m)$ will be examined. Since path $\langle v_1, v_2, ..., v_m, v_{m+1} \rangle$ is also a shortest path, visiting edge $(v_m, v_{m+1}, \Lambda_m)$ will update the label of v_{m+1} to be $(m+1, [t, t])$, since $t \in \Lambda_m$. Thus for $i = m+1$, the induction hypothesis still holds. Algorithm 1 finds the distance labels $L[v]$ for each vertex $v \in V$ for each graph snapshot $G_s, s \in T$ correctly.

Appendix B Examples and Illustrations

Example B.1 *Table 1 shows an example of merge labels process with input Ł[v]={(3,[0,1]), (1,[4,5]), (2,[7,9])} and* L_v *= {(3,[3,7]), (3,[9,9])}, as illustrated in Figure 1.*

In the first iteration, Ł[v] = (3,[0,1]) and l_v *= (3,[3,7]) are compared and merged. Since there is no overlapping, (3,[0,1]) is added to* L_m *and Ł[v] points to the next label. In the second iteration, Ł[v] = (1,[4,5]) and* l_v *= (3,[3,7]) are merged. Since there are overlapping, the first step is to remove the head (3,[3,3]) from Ł[v] and add it to* L_m *and* L_Δ*. Then the body is merged with a smaller distance coming from Ł[v], so (1,[4,5]) is added to* L_m*. Finally* l_v *is shortened to be (3,[6,7]) and Ł[v] points to the next label. The merging process continues until there is no labels left. The final output of the algorithm is* L_m *= {(3,[0,1]), (3,[3,3]), (1,[4,5]), (3,[6,6]), (2,[7,9])}, and* L_Δ *= {(3,[3,3]), (3,[6,6])}.*

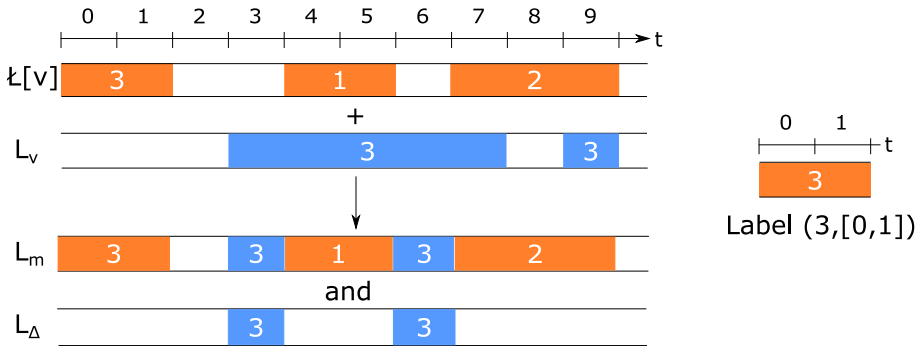

Figure 1: Overview of *mergeLabels*: (Ł[v], L_v) $\mapsto$ (L_m, L_Δ)

Table 1: Merge labels illustration: Ł[v]={(3,[0,1]), (1,[4,5]), (2,[7,9])} and L_v = {(3,[3,7]), (3,[9,9])}

Iteration	Ł[v]	l_v	result
1	(3,[0,1])	(3,[3,7])	(3,[0,1]) added to L_m
2	(1,[4,5])	(3,[3,7])	(3,[3,3]) added to L_m and L_Δ (1,[4,5]) added to L_m
3	(2,[7,9])	(3,[6,7])	(3,[6,6]) added to L_m and L_Δ (2,[7,9]) added to L_m

Example B.2 *Table 2 shows the execution of Algorithm* ***ECCI*** *with source vertex v_a in the graph shown in Figure 2. The elements in the current and next queue, updates of vertex earliest reachable time $t_m[v]$, vertex counts $V_m[t], V[t]$ and $R[t], D[t]$ are shown for each BFS level.*

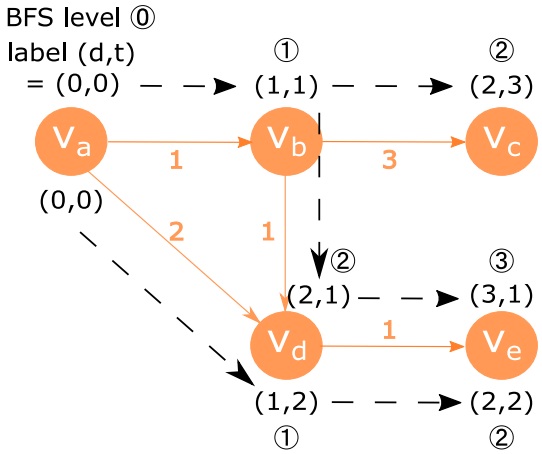

Figure 2: Insertion-only graph

Table 2: Algorithm 2 ECCI illustration

BFS		time instance t				*current*	*next*	$t_m[v]$
		0	1	2	3			
Level 0	$V_m[t]$	1	0	0	0	$(v_a,0)$	$(v_b,1)$ $(v_d,2)$	v_a: $\infty \mapsto 0$
	$V[t]$	1	1	1	1			
	$R[t]$	1	1	1	1			
	$D[t]$	0	0	0	0			
Level 1	$V_m[t]$	0	1	1	0	$(v_b,1)$ $(v_d,2)$	$(v_c,3)$ $(v_d,1)$ $(v_e,2)$	v_b: $\infty \mapsto 1$ v_d: $\infty \mapsto 2$
	$V[t]$	0	1	2	2			
	$R[t]$	1	2	3	3			
	$D[t]$	0	1	2	2			
Level 2	$V_m[t]$	0	1	0	1	$(v_c,3)$ $(v_d,1)$ $(v_e,2)$	$(v_e,1)$	v_c: $\infty \mapsto 3$ v_d: $2 \mapsto 1$ v_e: $\infty \mapsto 2$
	$V[t]$	0	1	1	2			
	$R[t]$	1	3	4	5			
	$D[t]$	0	3	4	6			
Level 3	$V_m[t]$	0	1	-1	0	$(v_e,1)$		v_e: $2 \mapsto 1$
	$V[t]$	0	1	0	0			
	$R[t]$	1	4	4	5			
	$D[t]$	0	6	4	6			

Appendix C Experiment Results

C.1 Synthetic Graph With Edge Insertions Only

Table 3: Experiment on synthetic data set of various topological sizes with only edge insertions

Data set	|T|	ECCI	Dynamic	HPLL-EXC	HPLL-PP	HPLL-SIZE	Avg # of Labels
Scale 17	2K	1,195	267,727 (224.04x)	24,325 (20.36x)	280,894	803MB	1.43
Scale 18	2K	2,559	749,433 (292.86x)	56,006 (21.89x)	850,250	2.09GB	1.46
Scale 19	2K	3,920	1,319,749 (336.67x)	254,046 (64.81x)	2,772,362	5.59GB	1.38
Scale 20	2K	16,219	5,812,834 (358.40x)	801,504 (49.42x)	9,214,751	15.0GB	1.52
Scale 21	2K	39,674	13,496,088 (340.17x)	1,926,235 (48.55x)	31,819,966	40.01GB	1.52

Table 4: Experiment on synthetic data set of various temporal sizes with only edge insertions

Data set	|T|	ECCI	Dynamic	HPLL-EXC	HPLL-PP	HPLL-SIZE	Avg # of Labels
Scale 21	500	27,196	6,122,163 (225.11x)	1,868,899 (68.72x)	32,473,270	39.6GB	1.49
Scale 21	1000	24,347	6,542,932 (268.74x)	1,368,979 (56.23x)	31,654,829	40.0GB	1.48
Scale 21	2000	39,674	13,496,088 (340.17x)	1,926,235 (48.55x)	31,819,966	40.0GB	1.52
Scale 21	4000	31,702	14,211,722 (448.29x)	1,410,922 (44.51x)	33,625,101	40.1GB	1.47
Scale 21	8000	31,219	16,695,736 (534.79x)	1,345,148 (43.09x)	33,696,332	40.2GB	1.47

C.2 Synthetic Graph With Edge Deletions

Table 5: Experiment on synthetic data set with edge deletions

Data set	|T|	% of deletions	ECC	Dynamic	Speedup	Avg # of Labels
Scale 17	2,000	0.05	73,807	410,031	5.56	2.20
Scale 18	2,000	0.05	113,871	1,104,722	9.70	2.30
Scale 19	2,000	0.05	255,783	2,845,908	11.13	2.12
Scale 20	2,000	0.05	993,091	7,548,091	7.60	2.64
Scale 21	2,000	0.05	1,969,356	14,278,437	7.25	2.55
Scale 21	500	0.05	1,370,419	5,920,827	4.32	2.51
Scale 21	1,000	0.05	1,220,337	7,170,815	5.88	2.42
Scale 21	4,000	0.05	1,706,108	14,185,630	8.31	2.47
Scale 21	8,000	0.05	1,762,195	21,995,172	12.48	2.46
Scale 17	2,000	0.1	66,314	451,041	6.80	2.27
Scale 18	2,000	0.1	124,970	1,036,607	8.29	2.36
Scale 19	2,000	0.1	360,746	3,087,018	. 8.56	2.22
Scale 20	2,000	0.1	1,247,703	8,095,228	6.49	2.65
Scale 21	2,000	0.1	1,930,947	14,614,718	7.57	2.66
Scale 21	500	0.1	1,295,215	6,489,788	5.01	2.62
Scale 21	1,000	0.1	1,284,292	8,708,594	. 6.78	2.52
Scale 21	4,000	0.1	1,761,623	15,868,067	9.01	2.57
Scale 21	8,000	0.1	1,821,417	23,189,266	12.73	2.53
Scale 17	2,000	0.15	64,422	380,381	5.90	2.36
Scale 18	2,000	0.15	102,759	954,735	. 9.29	2.48
Scale 19	2,000	0.15	257,449	2,928,169	11.37	2.24
Scale 20	2,000	0.15	738,462	5,376,536	7.28	2.73
Scale 21	2,000	0.15	1,699,324	12,923,220	7.60	2.76
Scale 21	500	0.15	1,254,446	6,496,121	. 5.18	2.60
Scale 21	1,000	0.15	1,213,926	7,757,869	6.39	2.68
Scale 21	4,000	0.15	1,383,928	12,164,652	8.79	2.68
Scale 21	8,000	0.15	1,726,279	21,096,900	12.22	2.67