"51th International Conference on Very Large Data Bases (VLDB'25)"

Approximate 2-hop neighborhoods on incremental graphs: An efficient lazy approach

Andrea Clementi* Luca Becchetti+ Luciano Gualà* Luca Pepè Sciarria*
Alessandro Straziota* Matteo Stromieri*

(+) SAPIENZA UNIVERSITÀ DI ROMA
(*) TOR VERGATA UNIVERSITÀ DEGLI STUDI DI ROMA

Abstract

In this work, we propose, analyze and empirically validate a lazy-update approach to maintain accurate approximations of the 2-hop neighborhoods of dynamic graphs resulting from sequences of edge insertions.
We first show that under random input sequences, our algorithm exhibits an optimal trade-off between accuracy and insertion cost: it only performs $O(\frac{1}{\varepsilon})$ (amortized) updates per edge insertion, while the estimated size of any vertex's 2-hop neighborhood is at most a factor ε away from its true value in most cases, *regardless* of the underlying graph topology and for any $\varepsilon > 0$.
As a further theoretical contribution, we explore adversarial scenarios that can force our approach into a worst-case behavior at any given time t of interest. We show that while worst-case input sequences do exist, a necessary condition for them to occur is that the *girth* of the graph released up to time t be at most 4.
Finally, we conduct extensive experiments on a collection of real, incremental social networks of different sizes, which typically have low girth. Empirical results are consistent with and typically better than our theoretical analysis anticipates. This further supports the robustness of our theoretical findings: forcing our algorithm into a worst-case behavior not only requires topologies characterized by a low girth, but also carefully crafted input sequences that are unlikely to occur in practice.
Combined with standard sketching techniques, our lazy approach proves an effective and efficient tool to support key neighborhood queries on large, incremental graphs, including neighborhood size, Jaccard similarity between neighborhoods and, in general, functions of the union and/or intersection of 2-hop neighborhoods.

General Framework

Layer 2: Downstream tasks. Using the output of the basic queries to solve more complex *downstream tasks.*

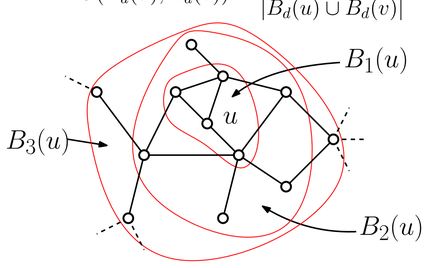
Layer 1: Basic Queries. Answer *basic queries* on d-hop neighborhoods (or d-radius balls)

$$B_d(u) := \{v \in V \mid d(u,v) \le d\}.$$

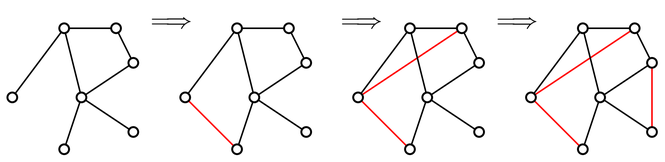
Basic queries:

- *size estimation* of $B_d(u)$
- *jaccard similarity* between $B_d(u), B_d(v)$

$$J(B_d(u), B_d(v)) = \frac{|B_d(u) \cap B_d(v)|}{|B_d(u) \cup B_d(v)|}$$

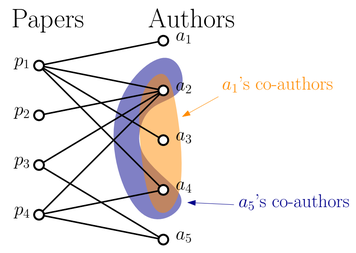


Layer 0: The Input. Underlying *dynamic* network.



Examples of Downstream tasks for 2-hop neighborhoods

Task 1: Predict possible future interactions in social netowork [1,2,3], e.g., between paper's authors.



high similarity between co-authors $\Longrightarrow$ possibily future collaboration

Task 2: Approximate top-k central nodes in a (dynamic) network according to some distance-based indices [5,6], e.g.,

Harmonic Centrality
$$HC(v) = \sum_{u \neq v} \frac{1}{d(u,v)}$$

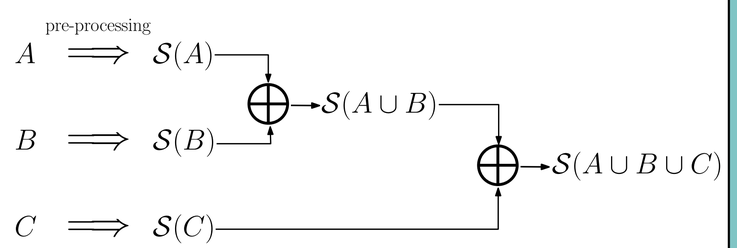
Closeness Centrality
$$CC(v) = \frac{1}{\sum_u d(u,v)}$$

Typical Solution: Data Sketches

Efficiency problems are typically solved using **data sketches** [4]: compact and efficient representations that accurately approximate basic queries.

More formally, a *data sketch* $\mathcal{S}(A)$ for a (large) set A, is a probabilistic representation, s.t.

1. has **compact size**, usually almost constant
2. **efficiently approximate** summaries of the sets
3. (optional) can be **composable**, i.e., given only $\mathcal{S}(A), \mathcal{S}(B)$ we can compute $\mathcal{S}(A \cup B)$



Efficiency Issues

To answer basic queries (even on *static graphs*) could be **inefficient**. For example, computing $J(B_d(u), B_d(v))$ requires $\Omega(|B_d(u)| + |B_d(v)|)$ time. This is unfortunately not feasible for many applications on large real-world networks.

Example of Composable Sketches

Basic Query	Composable Sketches
Size Estimation	Morris' counter, LogLog counter, KMV counter
Jaccard Similarity	MinHash, Bottom-k, one-permutation hashing

Our Contribution

We propose the Lazy-Updates algorithm, a **framework** that keeps data sketches updated on the 2-hop neighborhoods of incremental graphs, resulting from sequences of edge insertions.

The framework is **general**: any type of sketch can be plugged in, as long as it is composable. This makes our tool independent of the basic query, and therefore of the downstream task.

The Lazy-Update algorithm has two parameters that determine the trade-offs between precision and level of *"laziness"*, namely $\varphi \in (0,1)$ and $k \in \mathbb{N}$.

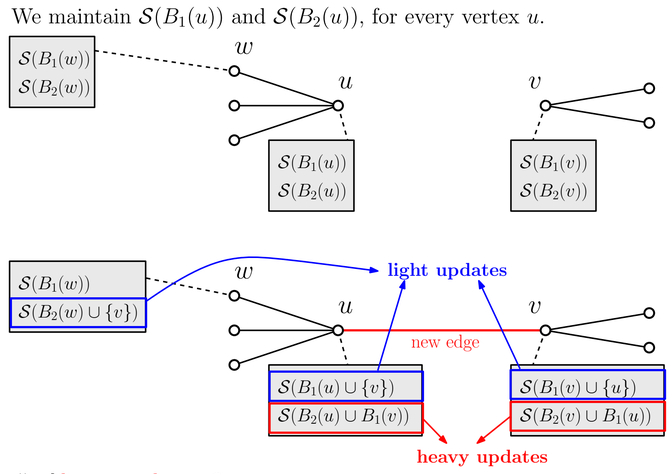
Performance: $O(\frac{1}{\varphi} + k)$ amortized number of sketch compositions.

Accuracy (under random insertions): By choosing $\varphi = \frac{\varepsilon}{1-\varepsilon}$, the sketches will contain **at least** a $(1-\varepsilon)$-fraction of the actual $B_2(u)$, for every vertex u, *with high probability.*

Accuracy (under adversarial insertions): We characterize a broad family of (dense) graphs for which, by choosing the parameters φ, k appropriately, the Lazy-Update algorithm always guarantees a $(1-\varepsilon)$-fraction with high probability, even under **adversarial** edge insertion streams.

Insert a new edge

We maintain $\mathcal{S}(B_1(u))$ and $\mathcal{S}(B_2(u))$, for every vertex u.



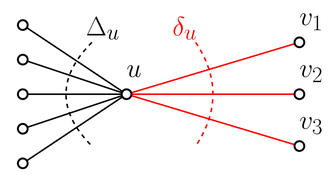
# of heavy updates: 2.
# of light updates: $2 + \Delta_u + \Delta_v$.

The Lazy-Updates Algorithm

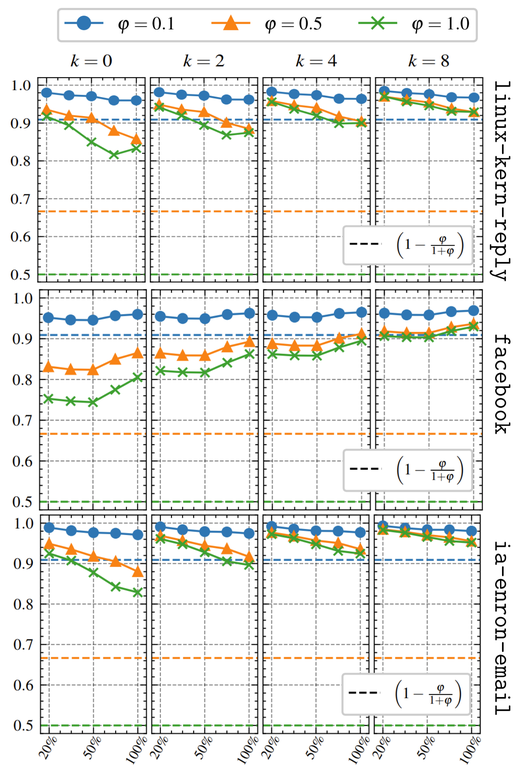
Idea: Since light-updates change the sketches by at most one element, we can be *lazy* and postpone the $O(\Delta_u + \Delta_v)$ updates, and run them later as *single heavy update.*

The algorithm.

1. fix the parameter $\varphi \in (0,1)$, and an integer $k \ge 0$.
2. when the edge (u,v) is inserted, execute heavy and light updates on u and v, and on k random neighbors of u and v.
3. when $\delta_u \ge \varphi\Delta_u$, execute a batch of light updates as a single heavy update, i.e., $\mathcal{S}(B_2(w)) \leftarrow \mathcal{S}(B_2(w) \cup B_1(u))$, for every neighbor w of u.
4. set $\delta_u = 0$, and continue.



Quality of Lazy-Updates Scheme



Average fraction of the 2-hop neighborhoods covered by sketches, for different values of φ, k, over real sequences of edge insertions, from 20% to 100% of the edges. Dashed lines show the theoretical expected fraction for random sequences.

Answering Basic Queries

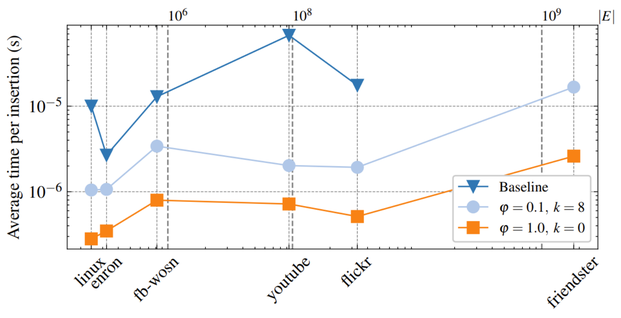
Basic Query: estimate the size of the 2-hop neighborhoods.
Sketch: KMV probabilistic counter.
Baseline: Keep everything always updated.

	k	φ 0.1	0.25	0.5	0.75	1
linux	0	14.39x	22.73x	30.27x	32.36x	35.65x
	2	12.00x	18.22x	22.03x	21.49x	23.18x
	4	11.52x	15.34x	17.63x	16.93x	17.54x
	8	9.57x	11.86x	12.80x	12.03x	12.22x
fb-wosn	0	5.48x	9.40x	11.15x	15.23x	16.22x
	2	5.22x	7.35x	7.16x	9.21x	9.67x
	4	4.32x	6.14x	6.05x	7.18x	7.23x
	8	3.79x	4.38x	4.91x	5.31x	5.22x
enron	0	4.17x	5.38x	6.64x	7.12x	7.70x
	2	3.61x	4.11x	4.68x	4.82x	5.27x
	4	3.05x	3.42x	3.78x	3.96x	4.21x
	8	2.51x	2.73x	2.85x	3.01x	3.12x

Speed-up with respect the baseline

	k	$\varphi = 0.1$	$\varphi = 0.5$	$\varphi = 1$	baseline
linux	0	0.14 ± 0.12	0.19 ± 0.14	0.17 ± 0.11	0.12 ± 0.10
	2	0.13 ± 0.11	0.14 ± 0.10	0.16 ± 0.11	
	4	0.14 ± 0.09	0.17 ± 0.12	0.14 ± 0.10	
	8	0.14 ± 0.11	0.14 ± 0.09	0.13 ± 0.10	
fb-wosn	0	0.16 ± 0.12	0.15 ± 0.11	0.21 ± 0.12	0.14 ± 0.11
	2	0.13 ± 0.09	0.15 ± 0.10	0.17 ± 0.11	
	4	0.14 ± 0.11	0.15 ± 0.10	0.16 ± 0.11	
	8	0.16 ± 0.13	0.14 ± 0.10	0.15 ± 0.11	
enron	0	0.13 ± 0.10	0.16 ± 0.11	0.20 ± 0.14	0.13 ± 0.11
	2	0.14 ± 0.11	0.16 ± 0.12	0.16 ± 0.12	
	4	0.13 ± 0.12	0.15 ± 0.12	0.15 ± 0.12	
	8	0.13 ± 0.11	0.14 ± 0.10	0.16 ± 0.13	

Average Absolute Percentage Error

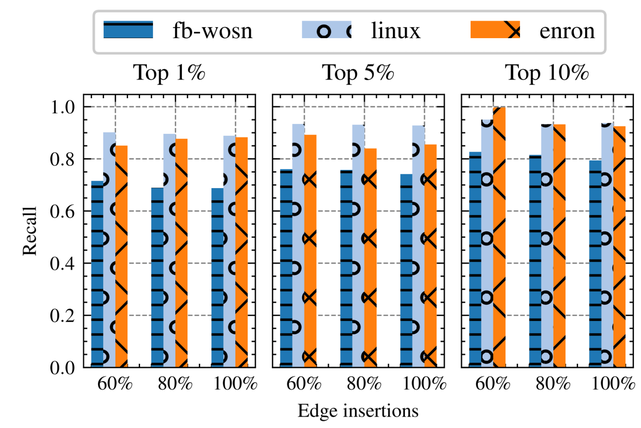


Average Update Time

Average time per insertion operation (in seconds) in log-scale. The datasets are arranged on the x-axis in increasing order of number of operations, again in log-scale. The baseline time for `soc-friendster` dataset is not reported, since it exceeded a time limit of 36 hrs.

Effectiveness of Lazy-Updates in Practice

Downstream task: maintain top-α% vertices with respect *harmonic centrality* on incremental networks.



Recall of the top 1%, 5%, and 10% vertices using the approximate harmonic centrality. The x-axis indicate different time snapshot along the edge insertion sequence.

Full Version

Our `C++` implementation

[1] Sachin U. Balvir, Mukesh M. Raghuwanshi, and Purushottam D. Shobhane. *An overview of similarity-based methods in predicting social network links: A comparative analysis.* IEEE Access

[2] Ahmad Zareie and Rizos Sakellariou. *Similarity-based link prediction in social networks using latent relationships between the users. Scientific reports.*

[3] Tao Zhou, Yan-Li Lee, and Guannan Wang. *Experimental analyses on 2-hopbased and 3-hop-based link prediction algorithms.* Physica A: Statistical Mechanics and its Applications.

[4] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. *Mergeable summaries.* ACM Transactions on Database Systems

[5] Paolo Boldi and Sebastiano Vigna. *Axioms for centrality.* Internet Mathematics.

[6] Yannick Rochat. *Closeness centrality extended to unconnected graphs: the harmonic centrality index.* In ASNA.