Age of Information in Random and Bipartite Networks

Thomas Jacob Maranzatto

University of Illinois Chicago (Ph.D.)
University of Maryland College Park

July 2024

Model Description

- Node n_0 sends updates to a network $G = (\mathcal{N}, E)$, $|\mathcal{N}| = n$

Model Description

- Node n_0 sends updates to a network $G = (\mathcal{N}, E)$, $|\mathcal{N}| = n$
- n_0 updates itself via a Poisson process with rate λ_e, and keeps track of how many updates have occurred

Model Description

- Node n_0 sends updates to a network $G = (\mathcal{N}, E)$, $|\mathcal{N}| = n$
- n_0 updates itself via a Poisson process with rate λ_e, and keeps track of how many updates have occurred
- n_0 sends updates to each $i \in \mathcal{N}$ as separate Poisson processes with rates $\lambda_0(i) = \frac{\lambda}{n}$.

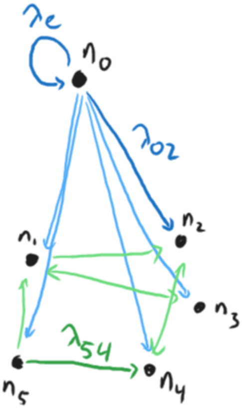
Model Description

- Node n_0 sends updates to a network $G = (\mathcal{N}, E)$, $|\mathcal{N}| = n$
- n_0 updates itself via a Poisson process with rate λ_e, and keeps track of how many updates have occurred
- n_0 sends updates to each $i \in \mathcal{N}$ as separate Poisson processes with rates $\lambda_0(i) = \frac{\lambda}{n}$.
- If $(i, j) \in E$, i communicates to j with rate $\lambda_i(j) = \frac{\lambda}{\deg(i)}$

Model Description

- Node n_0 sends updates to a network $G = (\mathcal{N}, E)$, $|\mathcal{N}| = n$
- n_0 updates itself via a Poisson process with rate λ_e, and keeps track of how many updates have occurred
- n_0 sends updates to each $i \in \mathcal{N}$ as separate Poisson processes with rates $\lambda_0(i) = \frac{\lambda}{n}$.
- If $(i, j) \in E$, i communicates to j with rate $\lambda_i(j) = \frac{\lambda}{\deg(i)}$
- $\lambda_i(S) = \sum_{j \in N(i) \cap S} \lambda_i(j)$ is the rate of node i into subset S

Model Illustration



Version Age

- The source and every node in the network have internal counters

Version Age

- The source and every node in the network have internal counters
- The counter for n_0 increments if and only if the process for n_0 updates.

Version Age

- The source and every node in the network have internal counters
- The counter for n_0 increments if and only if the process for n_0 updates.
- If $j \in \mathcal{N}$ receives a newer version from one of its neighbors, j uses this to update it's current version.

Version Age

- The source and every node in the network have internal counters
- The counter for n_0 increments if and only if the process for n_0 updates.
- If $j \in \mathcal{N}$ receives a newer version from one of its neighbors, j uses this to update it's current version.
- $X_j(t)$ is the number of versions node j is behind n_0 at time t

Version Age

- The source and every node in the network have internal counters
- The counter for n_0 increments if and only if the process for n_0 updates.
- If $j \in \mathcal{N}$ receives a newer version from one of its neighbors, j uses this to update it's current version.
- $X_j(t)$ is the number of versions node j is behind n_0 at time t
- $X_S(t) = \min_{j \in S} X_j(t)$.

Version Age

- The source and every node in the network have internal counters
- The counter for n_0 increments if and only if the process for n_0 updates.
- If $j \in \mathcal{N}$ receives a newer version from one of its neighbors, j uses this to update it's current version.
- $X_j(t)$ is the number of versions node j is behind n_0 at time t
- $X_S(t) = \min_{j \in S} X_j(t)$.
- The limiting average version age of S is $v_G(S) = \lim_{t \to \infty} \mathbb{E} X_S(t)$.

Combinatorial Formulation

- Yates (2021) showed the stochastic quantity $v_G(S)$ exists and can be computed combinatorially

Combinatorial Formulation

- Yates (2021) showed the stochastic quantity $v_G(S)$ exists and can be computed combinatorially
- We are mostly concerned with worst-case version age of a single vertex, $v_G(\{i\})$ or the average version age over the network

Combinatorial Formulation

- Yates (2021) showed the stochastic quantity $v_G(S)$ exists and can be computed combinatorially
- We are mostly concerned with worst-case version age of a single vertex, $v_G(\{i\})$ or the average version age over the network
- $$v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)}$$

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)
- k-neighbor cycles (Srivastava and Ulukus 2023)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)
- k-neighbor cycles (Srivastava and Ulukus 2023)
- $v_{\mathbb{Z}^2}(\{i\}) = O(n^{1/3})$ (Srivastava and Ulukus 2023)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)
- k-neighbor cycles (Srivastava and Ulukus 2023)
- $v_{\mathbb{Z}^2}(\{i\}) = O(n^{1/3})$ (Srivastava and Ulukus 2023)
- Higher Moments (Abd-Elmagid, Dhilon 2023)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)
- k-neighbor cycles (Srivastava and Ulukus 2023)
- $v_{\mathbb{Z}^2}(\{i\}) = O(n^{1/3})$ (Srivastava and Ulukus 2023)
- Higher Moments (Abd-Elmagid, Dhilon 2023)
- Non-Poisson updates (Kaswan, Ulukus 2023)

Previous Work

- $v_{K_n}(\{i\}) = \Theta(\log n)$ (Yates 2021)
- $v_{\overline{K}_n}(\{i\}) = \Theta(n)$ (Yates 2021)
- $v_{C_n}(\{i\}) = O(n^{1/2})$ (Buyukates, Bastopcu, Ulukus 2021)
- k-neighbor cycles (Srivastava and Ulukus 2023)
- $v_{\mathbb{Z}^2}(\{i\}) = O(n^{1/3})$ (Srivastava and Ulukus 2023)
- Higher Moments (Abd-Elmagid, Dhilon 2023)
- Non-Poisson updates (Kaswan, Ulukus 2023)
- Recent Survey (Kaswan, Mitra, Srivastava, Ulukus 2023)

Main Question

How does version age evolve as the communication network interpolates between K_n and $\overline{K_n}$?

Graphs We Study

- Uniform random d-regular graph $G(n, d)$
- Complete Bipartite Graph $K_{L,R}$
- Erdős-Reyni Random Graph $G(n, p)$

Summary of Results

Theorem

1. *For any fixed $d \geq 3$ and n growing, the worst case version age of $G(n, d)$ is $\Theta(\log n)$**

* *Holds with probability $1 - o(1)$ as $n \to \infty$*

Summary of Results

Theorem

1. *For any fixed* $d \geq 3$ *and* n *growing, the worst case version age of* $G(n, d)$ *is* $\Theta(\log n)$*
2. *If* $p = \frac{(1-\epsilon)\log n}{n}$*, then the average version age in* $G(n, p)$ *is* $\Omega(n^{\epsilon})$.* *If* $p = \frac{(100+\epsilon)\log n}{n}$ *then the average version age is* $\Theta(\log n)$*

* *Holds with probability* $1 - o(1)$ *as* $n \to \infty$

Summary of Results

Theorem

1. *For any fixed* $d \geq 3$ *and* n *growing, the worst case version age of* $G(n, d)$ *is* $\Theta(\log n)$*
2. *If* $p = \frac{(1-\epsilon)\log n}{n}$, *then the average version age in* $G(n, p)$ *is* $\Omega(n^{\epsilon})$.* *If* $p = \frac{(100+\epsilon)\log n}{n}$ *then the average version age is* $\Theta(\log n)$*
3. *If* $L(n) < R(n)$ *and both are nondecreasing with* n, *then the worst-case version age of* $K_{L,R}$ *scales as:*

* *Holds with probability* $1 - o(1)$ *as* $n \to \infty$

Summary of Results

Theorem

1. *For any fixed $d \geq 3$ and n growing, the worst case version age of $G(n, d)$ is $\Theta(\log n)$**
2. *If $p = \frac{(1-\epsilon)\log n}{n}$, then the average version age in $G(n, p)$ is $\Omega(n^{\epsilon})$.* If $p = \frac{(100+\epsilon)\log n}{n}$ then the average version age is $\Theta(\log n)$**
3. *If $L(n) < R(n)$ and both are nondecreasing with n, then the worst-case version age of $K_{L,R}$ scales as:*
   1. $L = \Theta(1) \implies v(K_{L,R}) = \Theta(n)$

* *Holds with probability $1 - o(1)$ as $n \to \infty$*

Summary of Results

Theorem

1. *For any fixed* $d \geq 3$ *and* n *growing, the worst case version age of* $G(n, d)$ *is* $\Theta(\log n)$ *
2. *If* $p = \frac{(1-\epsilon)\log n}{n}$, *then the average version age in* $G(n, p)$ *is* $\Omega(n^{\epsilon})$.* *If* $p = \frac{(100+\epsilon)\log n}{n}$ *then the average version age is* $\Theta(\log n)$ *
3. *If* $L(n) < R(n)$ *and both are nondecreasing with* n, *then the worst-case version age of* $K_{L,R}$ *scales as:*
   1. $L = \Theta(1) \implies v(K_{L,R}) = \Theta(n)$
   2. $L = n^{\alpha} \implies v(K_{L,R}) = \tilde{\Theta}(n^{1-\alpha})$

* *Holds with probability* $1 - o(1)$ *as* $n \to \infty$

Summary of Results

Theorem

1. *For any fixed* $d \geq 3$ *and* n *growing, the worst case version age of* $G(n, d)$ *is* $\Theta(\log n)$*
2. *If* $p = \frac{(1-\epsilon)\log n}{n}$, *then the average version age in* $G(n, p)$ *is* $\Omega(n^{\epsilon})$.* *If* $p = \frac{(100+\epsilon)\log n}{n}$ *then the average version age is* $\Theta(\log n)$*
3. *If* $L(n) < R(n)$ *and both are nondecreasing with* n, *then the worst-case version age of* $K_{L,R}$ *scales as:*
   1. $L = \Theta(1) \implies v(K_{L,R}) = \Theta(n)$
   2. $L = n^{\alpha} \implies v(K_{L,R}) = \tilde{\Theta}(n^{1-\alpha})$
   3. $L = \Theta(n) \implies v(K_{L,R}) = \Theta(\log n)$

* *Holds with probability* $1 - o(1)$ *as* $n \to \infty$

Proof Idea for the Theorem

- Use the identity $$v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \qquad (1)$$

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \qquad (1)$
- Leverage combinatorial properties of the graph classes to upper bound (1)

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)}$ (1)
- Leverage combinatorial properties of the graph classes to upper bound (1)
  - 'Binary Tree' Property of $K_{L,R}$

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \qquad (1)$
- Leverage combinatorial properties of the graph classes to upper bound (1)
  - 'Binary Tree' Property of $K_{L,R}$
  - Isoperimetric property for $G(n, d)$ (Bollobás '88)

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)}$ (1)
- Leverage combinatorial properties of the graph classes to upper bound (1)
  - 'Binary Tree' Property of $K_{L,R}$
  - Isoperimetric property for $G(n, d)$ (Bollobás '88)
  - For $G(n, p)$, count isolated vertices when p is small, and use lower bound on degree when p is large.

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \quad (1)$
- Leverage combinatorial properties of the graph classes to upper bound (1)
  - 'Binary Tree' Property of $K_{L,R}$
  - Isoperimetric property for $G(n, d)$ (Bollobás '88)
  - For $G(n, p)$, count isolated vertices when p is small, and use lower bound on degree when p is large.
- Do some elementary algebra.

Proof Idea for the Theorem

- Use the identity $v_G(S) = \frac{\lambda_e + \sum_{i \notin S} \lambda_i(S) v_G(S \cup \{i\})}{\lambda_0(S) + \sum_{i \notin S} \lambda_i(S)} \quad (1)$
- Leverage combinatorial properties of the graph classes to upper bound (1)
    - 'Binary Tree' Property of $K_{L,R}$
    - Isoperimetric property for $G(n, d)$ (Bollobás '88)
    - For $G(n, p)$, count isolated vertices when p is small, and use lower bound on degree when p is large.
- Do some elementary algebra.
- Prove a lemma showing that $v_G(S)$ is non-decreasing when adding edges.

Random Regular Graphs

Definition

Let ∂S be the set of edges in the cut spanning S and S^c. For any graph G, the edge expansion number $h(G)$ is given by

$$h(G) := \min_{|S| \leq n/2} \frac{|\partial S|}{|S|}$$

Theorem

(Bollobás 1988) For every fixed $d \geq 3$. Then there is a constant $c_d < \frac{1}{2}$ such that for the random d-regular graph $G(n, d)$,

$$\mathbb{P}[h(G(n,d)) \geq dc_d] \to 1 \textit{ as } n \to \infty$$

Proof of Logarithmic Version Age in $G(n, d)$

Proof

Rearranging Yates' Identity,

$$\lambda_e = \lambda_0(S)v(S) + \sum_{i \notin S} \lambda_i(S)(v(S) - v(S \cup \{i\}))$$

Since $G(n, d)$ is regular, we can partition ∂S into sets $A_1, ..., A_d$ where $A_j = \{v \notin S : |N(v) \cap S| = j\}$. Then,

$$\begin{aligned} \lambda_e &= \lambda_0(S)v(S) + \sum_{i=1}^{d} \sum_{j \in A_i} \lambda_j(S)(v(S) - v(S \cup j)) \\ &\geq \lambda_0(S)v(S) + \left(\frac{\lambda}{d} \sum_{i=1}^{d} i|A_i| \right) (v(S) - \max_{i \in N(S)} (v(S \cup i))) \end{aligned}$$

$G(n, d)$ (Cont.)

Proof (Cont.)

Since $\sum_{i=1}^{d} i|A_i| = \partial S$, by the result of Bollobás when $|S| < n/2$ a.a.s. $G(n, d)$ satisfies:

$$\lambda_e \geq \frac{\lambda |S|}{n} v(S) + \frac{\lambda}{d} c_d d |S| (v(S) - \max_{i \in N(S)} v(S \cup i))$$

$$\implies v(S) \leq \left(\frac{\lambda_e}{\lambda} + c_d |S| \max_{i \in N(S)} v(S \cup i) \right) / \left(\frac{|S|}{n} + c_d |S| \right) \quad (1)$$

By an analogous argument for when $|S| > n/2$:

$$v(S) \leq \left(\frac{\lambda_e}{\lambda} + c_d (n - |S|) \max_{i \in N(S)} v(S \cup i) \right) / \left(\frac{|S|}{n} + c_d (n - |S|) \right) \quad (2)$$

$G(n, d)$ (Cont.)

Proof (Cont.)

Therefore when unrolling the recursion for $v(\{i\})$, if $S < n/2$ we use inequality (1), otherwise we use (2). To that end let X be the sum corresponding to small subset size and letting $j := |S|$,

$$\begin{aligned} X &\leq \frac{\lambda_e}{\lambda}\left(\frac{1}{c_d + \frac{1}{n}}\right)\left(1 + \sum_{i=1}^{n/2}\prod_{j=1}^{i}\frac{c_d j}{\frac{j+1}{n} + c_d(j+1)}\right) \\ &\leq \frac{\lambda_e}{c_d \lambda}\left(1 + \sum_{i=1}^{n/2}\prod_{j=1}^{i}\frac{j}{j+1}\right) \quad \text{(Telescoping product)} \\ &= O(\log n) \end{aligned}$$

$G(n, d)$ (Cont.)

Proof (Cont.)

Letting Y be the terms corresponding to $|S| > n/2$, it can be shown that these also bounded as $O(\log n)$ by a similar argument. By our monotonicity lemma, no graph can have version age less than $O(\log n)$, which finishes the proof.

Complete Bipartite Graph Lemma

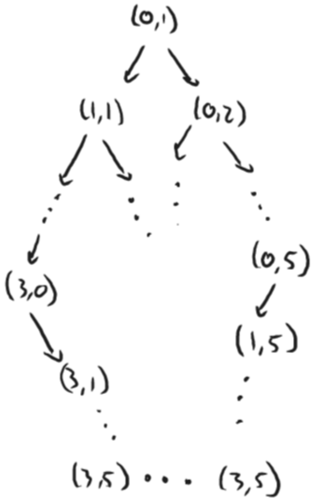
- Define $v(i,j)$ to be the version age of a subset with i elements on the left, j elements on the right.
- Define $u(i,j) = \frac{\lambda}{\lambda_e} v(i,j)$.

Lemma (1)

Let $K_{L,R}$ be a complete bipartite graph on n vertices. Then for any $S \subset V$ with $S \cap L = i$, $S \cap R = j$,

$$u(i,j) = \frac{1 + \frac{(|L|-i)j}{|R|} u(i+1,j) + \frac{(|R|-j)i}{|L|} u(i,j+1)}{\frac{i+j}{n} + \frac{(|L|-i)j}{|R|} + \frac{(|R|-j)i}{|L|}}.$$

Binary Tree Representation

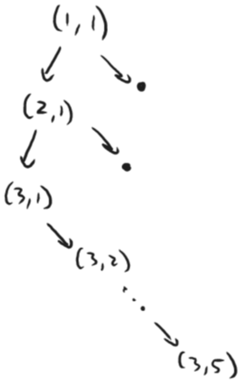


Bipartite Recursive Lemma

Lemma (2)

For any complete bipartite graph $K_{L,R}$,

$$u_{K_{L,R}}(1,1) \leq \min\{|R|(\log(|L|)+1), |L|(\log(|R|)+1)\}.$$



Future Work

- Developing new methodology to analyze non-Poisson gossip networks via combinatorial invariants (in preparation with Marcus Michelen)
- How to analyze time varying mobile networks
- Gossip protocols on spatial graphs with interference