Local Update Algorithms for Random Graphs

Romaric Duvignau

LATIN 2014

March 27, 2014

Joint Work with Philippe Duchon

Introduction

Context

- Peer-to-Peer Networks (modelled as graphs)
- Structure **dependent** on the *update* sequence
- Potential malicious sequence of updates
- Difficulty in designing/analysing update algorithms
- Analysis under nicely behaving update schemes

Introduction

Context

- Peer-to-Peer Networks (modelled as graphs)
- Structure **dependent** on the *update* sequence
- Potential malicious sequence of updates
- Difficulty in designing/analysing update algorithms
- Analysis under nicely behaving update schemes

Proposition: distribution-preserving update algorithms

- Maintain **exactly** a probability distribution of random graphs
- No probabilistic model for the update sequence

Distribution preserving algorithms

Definition

- For each possible vertex set V, G should follow a given target distribution μ_V, which is *preserved* through updates :
  - **Insertion**: If $G \sim \mu_V$ and $u \notin V$ then $\mathcal{I}(G, u) \sim \mu_{V \cup \{u\}}$
  - **Deletion**: If $G \sim \mu_V$ and $u \in V$ then $\mathcal{D}(G, u) \sim \mu_{V \setminus \{u\}}$

Distribution preserving algorithms

Definition

- For each possible vertex set V, G should follow a given target distribution μ_V, which is *preserved* through updates :
  - **Insertion**: If $G \sim \mu_V$ and $u \notin V$ then $\mathcal{I}(G, u) \sim \mu_{V \cup \{u\}}$
  - **Deletion**: If $G \sim \mu_V$ and $u \in V$ then $\mathcal{D}(G, u) \sim \mu_{V \setminus \{u\}}$

Uniform k-out graphs

- Simple digraphs with out-degree k
- Uniform distribution:
  - Each outgoing neighbourhood $N^+(v)$ is uniform among the k-subsets of $V - v$
  - The $N^+(v)$ are mutually independent

A local model

Local update model

- No global knowledge
- Knowledge of the current size
- Ability to pick a uniform random vertex `RandomVertex()`
- Ability to examine neighbours of a given node

A local model

Local update model

- No global knowledge
- Knowledge of the current size
- Ability to pick a uniform random vertex `RandomVertex()`
- Ability to examine neighbours of a given node

`RandomVertex()`

- Substitute for "contact a *friend* node"
- Uniformity : strong assumption
- Similar *external* mechanisms in the literature
- Very **costly**

A local model

Local update model

- No global knowledge
- Knowledge of the current size
- Ability to pick a uniform random vertex `RandomVertex()`
- Ability to examine neighbours of a given node

`RandomVertex()`

- Substitute for "contact a *friend* node"
- Uniformity : strong assumption
- Similar *external* mechanisms in the literature
- Very **costly**

We measure the cost of our algorithms essentially as the expected number of calls to `RandomVertex()`.

Our results

Preservation of uniform k-out graphs

- Several insertion and deletion algorithms
- Our best algorithms:
  - **Deletion**: calls $o(1)$ times `RandomVertex()`
  - **Insertion**: calls asymptotically k times `RandomVertex()`

Our results

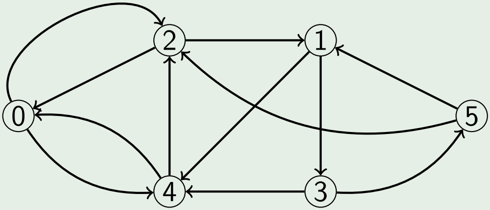
Preservation of uniform k-out graphs

- Several insertion and deletion algorithms
- Our best algorithms:
  - **Deletion**: calls $o(1)$ times `RandomVertex()`
  - **Insertion**: calls asymptotically k times `RandomVertex()`

These asymptotic bounds are optimal.

Deletion Algorithms

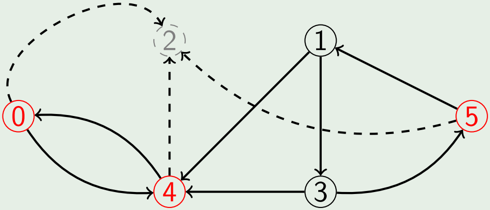
Deletion of vertex 2



Vertex 2 wants to leave the network.

Deletion Algorithms

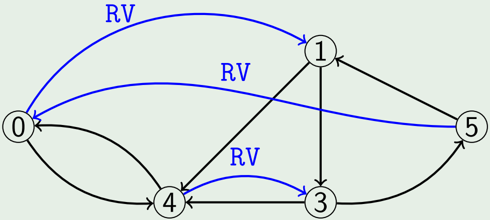
Deletion of vertex 2



Vertex 2 leaves the network, and there are 3 loose edges.

Deletion Algorithms

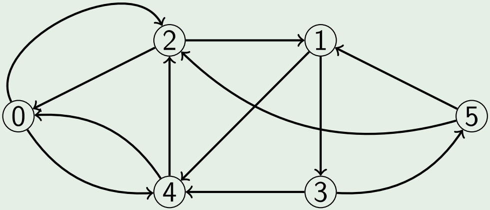
Deletion of vertex 2



Vertices 0, 4 and 5 replace 2 using `RandomVertex()`.
We need k calls on average.

Deletion Algorithms

Deletion of vertex 2

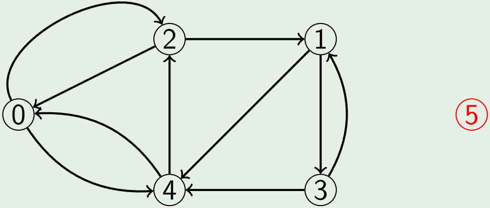


Deletion of vertex u

- Simple algorithm needs k calls to `RandomVertex()`
- Better algorithm: re-using u's successors
- $o(1)$-algorithm: re-using u's predecessors

Insertion Algorithms

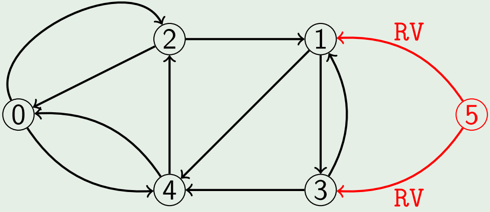
Insertion of vertex 5



Vertex 5 wants to join the network.

Insertion Algorithms

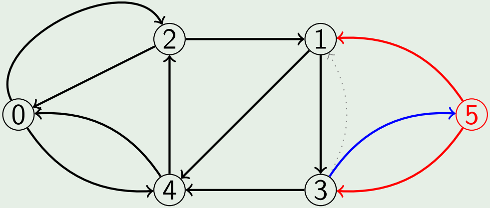
Insertion of vertex 5



Vertex 5 chooses 2 distinct vertices as successors, using `RandomVertex()`.

Insertion Algorithms

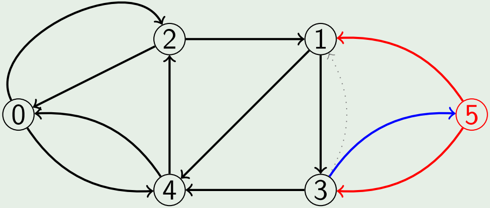
Insertion of vertex 5



Vertex 5 chooses $X \sim \text{Binomial}(n, k/n)$ distinct random vertices as predecessors, and *steals* one edge from each of them.
We need k calls in expectation to chose the predecessors.

Insertion Algorithms

Insertion of vertex 5



Insertion of vertex *u*

- Simple insertion needs, on average, $2k$ calls to `RandomVertex()`
- k-insertion: re-use the deleted edges

Conclusion

Results

- Precise definition of distribution-preserving algorithms
- Uniform k-out graphs: asymptotically optimal algorithms

Further research

- Uniform undirected k-regular graphs
- Distribution depending on the "identities" of the nodes: e.g. geometric graphs

Thank you

Thank you for your attention.