ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Alea Meeting, Munich, February 2016

# Lecture 3: Epidemic Tasks and Processes Beyond Rumor Spreading

Benjamin Doerr, LIX, École Polytechnique, Paris-Saclay

Outline:
Computing the minimum, average, and sum of node values
Epidemic discovery (triangulation) or making friends at LinkedIn

# Reminder Last Two Lectures

- Randomized rumor spreading (RRS):
  - round-based process in a graph $G = (V, E)$
  - starts with one node informed (knows the rumor)
  - in each round, each informed node calls a random neighbor and informs it (if it wasn't already)

- Main result first lecture: Works well as expected (now that we understand it ;-) ) – fast and highly robust
  - With probability at least $1 - o(1)$, the rumor spreading time is $\log_2 n + \ln n + o(\log n)$ in complete graphs, $G(n,p)$ random graphs with $p = \omega((\log n)/n)$, random regular graphs $G(n,d)$ with $d = \omega(1)$.
  - If each call fails independently with probability $p$, then the rumor spreading time is $\log_{2-p} n + \ln(n)/(1-p) + o(\log n)$; this is proven for complete graphs, but should hold for all graphs above.

# Reminder Last Two Lectures (2)

- Things also work well in many other graphs, but sometimes the analysis is more tricky.
  - Example: In hypercubes, with probability at least $1 - o(1)$, the rumor spreading time is $\Theta(\log n)$, but so far no-one was able to make the leading constant precise (despite several attempts).

- In real-world network models, rumor spreading is fast, sometimes ultrafast, but often different rumor spreading mechanisms are needed
  - The usual "push" rumor spreading needs time $\Omega(n^{\alpha})$ in preferential attachment (PA) graphs
  - Push-pull does it in $\Theta(\log n)$ time
  - Push-pull without calling someone it two successive rounds works in $\Theta(\log n / \log\log n)$ time, which is the diameter
  - Asynchronous push-pull works in $O\left(\sqrt{\log n}\right)$ time.
  - Similar picture for Chung-Lu graphs

# Plan For Today

- We have need that a single piece of information can be spread quite well in an epidemic (gossip-based) manner. How about other tasks in networks?

- Part 1: Assume that each node has some private date. We want that all nodes learn something about all data, e.g., the sum, average, minimum value…

- Part 2: Gossip-based triangulation
  - epidemic algorithm for resource location
  - random process modelling how people get to know each other via electronic social networks

# Part 1: Information Dissemination Beyond Spreading a Single Rumor

- **Examples:**
  1. All-to-all rumor spreading: each node has a rumor, all rumors shall be disseminated to all nodes
  2. Assume that each node has a local variable $x_v$. Let each node compute some statistic about these values, e.g., minimum, sum, average, quadratic mean, …
  3. Let each node compute the number of nodes of the network

- **Observations:**
  - Task 3 is a special case of Task 2 (summation with all $x_v = 1$)
  - Task 2 can be reduced to Task 1 (send all $x_v$ to everyone and then have everyone compute locally what they want)
    - but this might need more communication than necessary

# All-to-all Rumor Spreading

- **Simple solution:** Run $n$ single-rumor spreading processes in parallel. Each node calls a random neighbor and forwards all rumors it knows.

- **Analysis:**
  - If you look at a single rumor, we have exactly the same process as in single-rumor spreading.
  - Consequence: If single-rumor spreading spreads the rumor from an arbitrary node to all others in time $T$ with probability at least $1 - \varepsilon$, then all-to-all rumor spreading succeeds in time $kT$ with probability at least $1 - n\varepsilon^k$ (for all integers $k$).
    - union bound gives the $n$ in the failure probability
    - re-try argument introduces the $k$
    - E.g., $O(\log n)$ rounds suffice in complete graphs whp

# All-to-all Rumor Spreading (2)

- **Problem with $n$ parallel single-rumor processes: Much traffic in the network!**
  - One call transmits up to $n$ rumors
  - If only one rumor can be sent per round, then the rumor spreading time typically increases by a factor of $n$, because at the end of the process, each node touches each rumor only every $n$-th round

- Better solution via random linear network coding (Haeupler (STOC'11), details omitted here):
  - idea: don't send rumors, but send random linear combinations of rumors. Then you can decode all rumors once you've heard (any) $n$ linearly independent messages.
  - result: very roughly, $O(n + T)$ rounds suffice to spread $n$ rumors, where $T$ is the time you need to spread one rumor with (very) high prob.

- Bottom line: All-to-all rumor spreading is costly

# Computing Aggregate Data From Node Values

- Setting:
  - Each node $v$ has a piece of data, a number $x_v$.
  - We want each node to learn some aggregate data (minimum/maximum, sum, average, …) about all these values.

- Plan: We want to be more clever than sending all data to all nodes, as this takes at least $\Omega(n^2)$ rumor transmissions, which hopefully is not necessary.

- Assumptions:
  - We assume that all nodes start the process at the same time (first round). This can, e.g., be initiated by a single-rumor spreading process.
  - We do not care about a termination criterion.

- Let's start with computing the minimum…

# Clever Minimum Computation

- **Epidemic/gossip-based compute-min algorithm:** In each round,
  - each node $v$ calls a random neighbor and forwards to it his $x_v$,
  - then each node $v$ updates its $x_v$ to the minimum of its $x_v$ and all numbers it received.

- **Analysis:** Track how the minimum value $x_{\min} := \min\{x_v | v \in V\}$ spreads!
  - Call a node $v$ "informed" if $x_v = x_{\min}$.
  - Then informed nodes stay informed.
  - If an informed node calls an uninformed node, it becomes informed.
  - → We precisely simulate the classic rumor spreading process ☺

- **Result:** Computing the minimum takes exactly the same time as spreading a single rumor
  - Communication effort: One call per node per round.

# Clever Computation of Sums, Averages,…

**Observation: "Everything" can be reduced to epidemically computing sums.**

- Number of nodes: Compute the sum of the node values $x_v = 1$.
- Average:
  - Compute the sum $X$ of the node values $x_v$ in an epidemic manner
  - compute the number $n$ of nodes (as sum of $y_v = 1$) epidemically
  - compute locally at each node the average $X/n$
- root mean square (quadratic mean) $x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\sum_{v\in \mathrm{V}} x_v^2}$:
  - nodes compute locally $x'_v = x_v^2$
  - compute epidemically $n$ and $X = \sum_{v\in V} x'_v$ as above
  - compute locally $x_{\mathrm{rms}} = \sqrt{X/n}$

# Epidemically Computing Sums

- Difficulties:
  - we don't want to send all $x_v$ to all nodes: quadratic communication effort
  - additional difficulty that nodes may not have unique identifiers: when you hear a number the second time, you do not know it is a copy of the rumor already heard or if two nodes have the same number

Harsh assumption, but useful to avoid algorithms that learn the network structure and then rely on it (not robust)

- Can we do something reasonable under such condition?

- YES ☺ [when we allow approximate solutions]
  - iterated averaging (works for averages, not the rest)
  - reduction to minimum (super-cool trick, needs $x_v \geq 1$)

# Iterated Averaging: Rough Idea

- Basic idea: Nodes call random neighbors and average their values
- Example of an averaging operation between two nodes $u$ and $v$:
  - $u$ and $v$ send their values $x_u$ and $x_v$ to the other node
  - both $u$ and $v$ update their value to $\frac{x_u+x_v}{2}$.
- Observations:
  - the sum of all node values never changes ("conservation of mass")
  - intuitive, but non-trivial: the node values converge to the average
- Difficulties:
  - For pair-wise averaging: How do you ensure that each node is part of one averaging operation only?
  - For other averaging: How do you deal with different node degrees?

# One Iterated Averaging Protocol That Works [Boyd et al. (2006)]

- In each round…
  - each node flips a fair coin to decide whether it is *active* or *passive*
  - initiate communication: each *active* node $v$ …
    - with probability $\deg(v)/2\Delta$ contacts a random neighbor;
    - with probability $1 - \deg(v)/2\Delta$ is does nothing
  - a *passive* nodes that was contacted exactly once accept this contact request (all other request are lost)
  - for each pair $(u, v)$ established in this manner does a pair-wise averaging operation: $u$ and $v$ send their values $x_u$ and $x_v$ to the other node; then both update their values to $\frac{x_u + x_v}{2}$.

- Properties: does not need any node identifiers etc.
  - needs that all nodes know the max-degree $\Delta$ (requires one maximum-communication analogous to a minimum communication)

# Performance of Iterated Averaging

- In random geometric graphs $G(n,r)$ with $r$ sufficiently above the connectivity threshold, the $n^{-\varepsilon}$-averaging time (roughly: the time to get the imbalance down to $n^{-\varepsilon}$ times the initial one) is of order $\Theta(r^{-2}\log n)$
  - Recall: The diameter is only $\Theta(r^{-1})$ !
- Roughly the same result holds for grids.
- Polylogarithmic averaging times for expander graphs (and the complete graph)
- **Summary: Iterated averaging can be slow!**

# Intuitive Example: Slow Averaging

- Simplified averaging mechanism in graphs with constant max-degree $\Delta$: *Move a $1/2\Delta$ fraction of your value to each neighbor!*
  - send your value to all neighbors
  - reset your value to $x(v) := \left(\sum_{u \sim v} x(u) + (2\Delta - \deg(v))x(v)\right)/2\Delta$
- Example situation:
  - $n$ odd, $e := (n-1)/2$, $G$ is a path with vertices (in that order) $v_{-e}, v_{-e+1}, \ldots, v_{-1}, v_0, v_1, \ldots, v_e$
  - initial node values: $x(v_i) = i$.
- Analysis: By symmetry, $x(v_0)$ is always zero
  - Induction over time: For positive $i$, $x(v_i)$ never exceeds its initial value.
  - Since always $x(v_0) = 0$ and $x(v_1) \leq 1$, at most a half unit of mass is transported out of $\{v_1, \ldots, v_e\}$.
  - It takes quadratic time to get all $x(v) \leq n/8$.

# Reduction to Minimum: Idea (1)

- Reminder: $X$ has *exponential distribution* with rate $\lambda$ if $\Pr[X > z] = \exp(-\lambda z)$ for all $z \in \mathbb{R}_{\geq 0}$.
  - $E[X] = 1/\lambda$

- **Lemma:** Let $X_1, \ldots, X_n$ be independent exponential random variables with rates $\lambda_1, \ldots, \lambda_n$. Let $X = \min\{X_1, \ldots, X_n\}$. Then $X$ has an exponential distribution with rate $\lambda = \sum_{i=1}^{n} \lambda_i$.

- **Proof:** For all $z \geq 0$, we have
  - $\Pr[X > z] = \Pr[\forall i \in [1..n]: X_i > z]$
  - $= \prod_{i=1}^{n} \Pr[X_i > z]$ [independence of the $X_i$]
  - $= \prod_{i=1}^{n} \exp(-\lambda_i z)$
  - $= \exp(-\lambda z)$.
  - Hence $X$ has an exponential distribution with rate $\lambda$.

# Reduction to Minimum: Idea (2)

- **Plan: Exploiting exponential distributions**
  - Nodes locally sample an exponential random variable $X_v$ with rate $x_v$
  - We use an epidemic minimum-spreading algorithm for the $X_v$
    → all nodes learn $X = \min_{v \in V} X_v$
    - this is an exponential random variable with rate $x := \sum_{v \in V} x_v$.
  - Nodes hope that $X$ is close to $E[X] = 1/x$ and take $1/X$ as approximation for the desired $x$.

- Increase accuracy: Repeat this procedure $r$ times and average $X$ values learned before taking the reciprocal

# Algorithm of Mosk-Aoyama and Shah (2008)

- Each node $v$ locally samples $r$ independent exponential random variables with rate $x_v$, let $y_v^1, \ldots, y_v^r$ be the outcome.
- **Communication phase.** Run the minimum-algorithm for each component $i \in [1..r]$ in parallel: In each round (of sufficiently many),
  - each node $v$ calls a random neighbor and sends to it $(y_v^1, \ldots, y_v^r)$;
  - each node $v$ then resets $(y_v^1, \ldots, y_v^r)$ to the component-wise minimum of $(y_v^1, \ldots, y_v^r)$ and all vectors he received
- Each node $v$ locally computes $\bar{y} = \frac{y_v^1 + \cdots + y_v^r}{r}$ and takes $1/\bar{y}$ as its approximation for the sum $x = \sum_{v \in V} x_v$.
- Two possible problems:
  - rumor spreading could fail (too few rounds, unlucky random choices)
  - the approximation to $x$ could be bad (bad luck when sampling the $y_v^i$)

# Analysis: Communication Phase

- Communication phase: In each round,
  - each node $v$ calls a random neighbor and sends to it $(y_v^1, \ldots, y_v^r)$;
  - each node $v$ resets $(y_v^1, \ldots, y_v^r)$ to the component-wise minimum of $(y_v^1, \ldots, y_v^r)$ and all vectors he received

- Lemma:
  - Let $\varepsilon > 0$. Let $T$ be large enough so that classic rumor spreading spreads a rumor with probability $1 - \varepsilon$ in $T$ rounds
  - Then after $T$ rounds, all nodes know the true minima with prob. $1 - r\varepsilon$.

- Proof: For each $i \in [1..r]$ in parallel we run a minimum-spreading process with the node values $\left(y_v^i\right)_{v \in V}$.
  - Each of these processes spreads its minimum in time $T$ w. prob. $1 - \varepsilon$
  - union bound

# Analysis: Approximation Error

- Assume that the communication phase was successful:
  - all nodes know the minima $m^i := \min\{y_v^i | v \in V\}, i \in [1..r]$.
  - Consequently, all compute the same $\hat{x} = r/(m^1 + \cdots + m^r)$ as approximation to $x$

- Lemma: For all $\delta < 1$, we have $(1-\delta)x \leq \hat{x} \leq (1+\delta)x$ with probability $1 - 2\exp(-\delta^2 r/12)$.

- Proof: $M = m^1 + \cdots + m^r$ is the sum of $r$ independent exponential random variables with rate $x$.
  - $\Pr[|M - E[M]| \geq \delta E[M]] \leq 2\exp(-\delta^2 r/3)$. [Cramér's theorem, $\delta < 1/2$]
  - Since $\hat{x} = r/M$ and $x = r/E[M]$, we obtain $\Pr[|\hat{x} - x| \geq \delta x] \leq 2\exp(-\delta^2 r/12)$ for all $\delta < 1$.

# Summary MAS Algorithm

- Let $G = (V, E)$ be any network. Assume that each node has a node value $x_v \geq 1$.

- Let $T$ be such that $\Pr[T_G > T] \leq \varepsilon$.
- Run the MAS-algorithm with parameter $r$ for $T$ rounds.
  - recall that in each round each node sends out $r$ numbers!

- Then with probability at least $1 - 2\exp(-\delta^2 r/12) - r\varepsilon$ all nodes learn an approximation $\hat{x}$ of the sum $x = \sum_{v \in V} x_v$ of node values such that
$$|\hat{x} - x| \leq \delta x.$$

# Part 2: Gossip-based Discovery

- Bib.-note: All of the following is from Haeupler, Pandurangan, Peleg, Rajaraman, Sun (SPAA'12, CPC to appear)

- Discovery process (gossip-based triangulation):
  - Start: a connected network $G_0 = (V, E_0)$
  - Round $t$:
    - Each node $u$ picks two neighbors $x_u$ and $y_u$ in $G_{t-1}$
    - $E_t := E \cup \{\{x_u, y_u\} | u \in V\}$, $G_t := (V, E_t)$
  - until $G_t = K_V$ (the complete graph on $V$)

- Main structural difference to what we did so far: *Dynamic network*
  - the network structure changes (massively) over time

- Question: How long does this process need to terminate?

# Applications

- Algorithm: Resource discovery in peer-to-peer (P2P) networks
  - nodes can only communicate with nodes of which they know the IP address
  - resource discovery problem: let nodes learn the IP addresses of all other nodes in the network

- Process: Making “friends” in social networks like LinkedIn, …

# Preliminary Observations

- General lower bound: any process where each node can add at most one edge per round needs $\Omega(n)$ rounds to obtain a complete graph if the starting graph is missing at least a constant fraction of all edges

- Easy to find starting graphs where gossip-based triangulation needs $\Omega(n \log n)$ rounds
  - complete graph plus one out-going edge: each $\Theta(n)$ rounds the degree of the outside vertex roughly doubles
  - complete graph minus $\Theta(n^{\alpha})$ disjoint edges, $0 < \alpha < 1$: each missing edge in each round has a $\Theta(1/n)$ probability of being added.
    - roughly resembles a coupon collector process

# More Observations

- Non-monotone process: More edges can increase the termination time!
- Example: A path on 4 vertices vs. a triangle with one additional outgoing edge.
- See the original paper at http://arxiv.org/abs/1202.2092 for a graphic representation which I can't give in a web version

# Result

- Theorem:
  - when started with any connected graph, the gossip-based triangulation process ends after $O(n\ (\log n)^2)$ rounds whp.
  - when started with any connected graph having $k$ edges less than the complete graph, the gossip-based triangulation process needs $\Omega(n \log k)$ rounds whp.

- Gap between upper and lower bound: $\Theta(\log n)$

- Key challenges in the analysis:
  - non-monotonicity
  - dynamic networks

  → any “new” edge can change the situation

# An Example: Star Graphs

- Star graph on $n$ vertices: One central node connected to $n-1$ nodes of degree one.

- Gossip-based triangulation on a star graph with $n$ vertices:
  - First $T = o(n^{0.5})$ rounds: $T$ disjoint edges between leaves are added
  - At time $T = \Theta(n^{0.5})$: The first intersection of new edges and $O(1)$ rounds later, this path of length 2 is completed to a triangle
  - Up to time $T = (0.5 - \varepsilon)n$: The new edges (viewed alone) form small components of size at most $O(\log n)$. These are almost immediately completed to a clique.
  - Phase transition: In the (short) time up to $T = (0.5 + \varepsilon)n$, a giant component in the graph of new edges appears.

# Star Graphs (2)

- Observation: The components formed by the new edges alone are the same in the true process and in the process where only the central vertex is active.

- For a component $C$ of size $s \leq (1-\varepsilon)n$,
  - the probability that the central vertex adds an edge from $C$ to the outside, is $\Theta(s/n)$, hence the expected waiting time for this is $\Theta(n/s)$
  - in time $O(s\,(\log s)^2)$, the component completes itself to a clique if nothing else happens to it (Haeupler et al. result)

- Components with less than $\tilde{O}(n^{0.5})$ vertices are cliques whp, because the clique formation is much faster than the addition of edges to the outside
- Larger components: the work of the central vertex becomes faster and new vertices are added quicker than the self-triangulation.

# Summary Star Graphs

- Already when starting with a simple star graph, the gossip-based triangulation process is highly non-trivial

- It combines two subprocesses
  - action of the central node (independent of all the rest): throw in one random edge between other vertices per round
    - “evolution of random graphs”
  - “self-triangulation” process of the components formed by new edges
    - small components very quickly become cliques
    - larger components: self-triangulation slower than the addition of new edges

# Proof Ideas: Upper Bound

- Key lemma: If you start the process with a graph having minimum degree $\delta$, then after $T = O(n \log n)$ rounds, the minimum degree is $\min\{(9/8)\delta, n-1\}$ whp.
  - a logarithmic number of such phases brings the min-deg. to $n-1$

- How does the degree of a node $u$ grow?
- A neighbor $w \in N(u)$ adds an edge between $u$ and a node $v \in N(w) \cap N^2(u)$
  - $N^i(u)$: vertices in distance *exactly* $i$ from $u$

# Proof Ideas: Upper Bound (2)

- Lemma 1: Let $\delta := \delta(G)$, $\deg(u) = \Theta(\delta)$, $w \in N(u)$ such that $|N(w) \cap N^2(u)| = \Omega(\delta)$, then in one round with probability $\Omega(1/n)$ an edge between $u$ and $N^2(u)$ is added by $w$.

- Proof: The probability is

$$2 \cdot \frac{|N(w) \cap N^2(u)|}{\deg(w)} \cdot \frac{1}{\deg(w)} \geq 2 \cdot \frac{|N(w) \cap N^2(u)|}{\deg(u) + |N(w) \cap N^2(u)|} \cdot \frac{1}{n} = \Theta\left(\frac{1}{n}\right)$$

# Proof Ideas: Upper Bound (3)

- Lemma 1: Let $\delta := \delta(G)$, $\deg(u) = \Theta(\delta)$, $w \in N(u)$ such that $|N(w) \cap N^2(u)| = \Omega(\delta)$, then in one round with probability $\Omega(1/n)$ an edge between $u$ and $N^2(u)$ is added by $w$.

- Lemma 2: Let $\delta := \delta(G)$, $\deg(u) = \Theta(\delta)$, and there are $\Omega(\delta)$ nodes $w \in N(u)$ such that $|N(w) \cap N^2(u)| = \Omega(\delta)$ ["good nodes"]. Then after $T = \Theta(n \log n)$ rounds, the degree of $u$ has increased by $\Theta(\delta)$ with prob. $1 - n^{-2}$.

- Proof: The only way how a good node becomes bad is that $\Theta(\delta)$ nodes in $N(w) \cap N^2(u)$ move to $N(u)$. If this happens, we are done ☺
  - Can assume that assumptions of lemma hold for $T$ rounds (else done)
- Now $\Theta(\delta)$ good nodes in parallel try to use Lemma 1 → after $O(T/\delta)$ rounds, wp. $1 - n^{-3}$, one is lucky and adds a neighbor to $u$
- Repeat this $\delta$ times and $u$ got $\Theta(\delta)$ new neighbors

# Proof Ideas: Upper Bound (4)

- Lemma 2: Let $\delta := \delta(G)$, $\deg(u) = \Theta(\delta)$, and there are $\Omega(\delta)$ nodes $w \in N(u)$ such that $N(w) \cap N^2(u) = \Omega(\delta)$ ["good nodes"]. Then after $T = \Theta(n \log n)$ rounds, the degree of $u$ has increased by $\Theta(\delta)$ with prob. $1 - n^{-2}$.

- Lemma 2 was the easiest among three cases. The other two:
  - at least one good $w$, but less than the $\Omega(\delta)$ of Lemma 2
  - no good $w$ at all
- We have to omit both cases here ☹

- Summary: Relatively efficient gossip-based resource discovery
  - proofs much harder due to the dynamic behavior of the network and the non-monotonicity
  - Open problem: logarithmic gap between upper and lower bound

# Summary: Epidemics Beyond Rumor Spreading

- Many tasks other than spreading a single rumor can be done by gossip-based algorithms
  - minimum is easy (essentially same difficulty as rumor spreading)
  - averages: iterated averaging
    - relatively slow on several graphs with large diameter
  - MAS algorithm (for sums, averages, …) computes a $(1 \pm \delta)$ factor approximation in time roughly the rumor spreading time (parallel spreading of $r$ rumors)
  - gossip-based triangulation

# Course Summary

Epidemic/gossip-based algorithms&processes

- Models for processes in the real world
  - epidemics (including computer viruses and malware)
  - rumors
  - making acquaintances
  - [viral marketing, influence processes, adoption of new technologies]
- Lightweight and robust distributed algorithms
  - information dissemination
  - computing averages etc.
  - resource location

# Course Summary: Algorithms

- Epidemic/gossip-based algorithmics
  - simple generic algorithm design paradigm
  - performance often close to the best that can be achieved
  - works well in networks without central organization, stable structure, or reliable communication

- Analysis techniques: Similar to other randomized/distributed algorithms
  - Expectations, birthday paradox, coupon collector
  - Markov chain arguments (adding waiting times)
  - Strong concentration: Chernoff bounds, method of bounded differences

# Many Open Problems

- This is a young area of research, so many things are not well understood
  - many particular research problems (some mentioned in this course)
  - designing “more clever” gossip-based algorithms (add the right dependencies to the random choice)
  - epidemic processes in real-world networks
  - gossip-based algorithms in dynamic networks

- Lots of work on STOC, FOCS, SODA, PODC, … in the last 5 years that is not yet fully digested
  - → good topic for an internship, a Master thesis, a PhD thesis, …

- This is the end of this short course. I hope you enjoyed it. Don’t be shy to contact me if you have questions or comments.