Faster Agreement Via a Spectral Method for Detecting Malicious Behavior

Valerie King and Jared Saia

Byzantine Agreement

Each node starts with a bit

Goal: 1) all good nodes output the same bit; and 2) this bit equals an input bit of a good node

t = # bad nodes controlled by an adversary

Applications

- Bitcoin

  "Bitcoin is based on a novel ***Byzantine agreement*** *protocol in which cryptographic puzzles keep a computationally bounded adversary from gaining too much influence"* [ML '13]

- Game Theory (Mediators)

  "deep connections between implementing mediators and various agreement problems, such as ***Byzantine agreement****"* [ADH '08]

- Peer-to-peer networks

  "These replicas cooperate with one another in a ***Byzantine agreement*** *protocol to choose the final commit order for updates."* [KBCCEGGRWWWZ '00]

Also: Secure Multiparty Computation, Databases, State Machine Replication, Sensor Networks, Cloud Computing, Control systems, etc.

Classic Model

- **Asynchronous:** Adversary schedules message delivery
- **Full Information:** Adversary knows state of all nodes
- **Adaptive Adversary:** Adversary takes over nodes at any time up to t total

Previous Work

- [Ben-Or '83] gave first randomized algorithm to solve BA in this model
- [FLP '85] showed BA impossible for deterministic algorithms even when $t=1$
- Ben-Or's algorithm is exponential expected communication time
- **Communication Time**: maximum length of any chain of messages

Our Result

- Las Vegas algorithm that solves Byzantine agreement in the classic model
- We tolerate $t = \theta(n)$
- Expected communication time is $O(n^3)$
- Computation time and bits sent are also polynomial in expectation

Ben-Or's algorithm

- Consists of iterations
- Uses private random bits to create a fair global coin with probability $1/2^n$ in each iteration
- For each iteration there is a **correct direction**
- **I**f there is a global coin and it is in this direction, agreement is reached

Our goal: Get a fair global coin after polynomial iterations using the private random bits

Key Idea

- With constant probability, sum of coinflips of **good** nodes will be in the correct direction and large enough for Ben-Or to succeed
- Bad nodes need to generate **bad** deviation in the opposite direction of equal magnitude to foil this good event
- If the few bad nodes generate large deviation repeatedly, we can find them

Issues

Ignore in this talk.
See paper for details

No more than 2t coins from good nodes, no more than 2 per node that are not common.

Common coins are known to n-4t good nodes.

Remaining Problem

- Bad nodes create biased coinflips

Deviation

- All coinflips are either +1 or -1
- The **deviation** of p in an iteration is the absolute value of the sum of p's coinflips
- The **direction** of p in an iteration is the sign of the sum of p's coinflips

Iterations and Epochs

- In each iteration, we run modified Ben-Or
- There are $m = \theta(n)$ iterations in an **epoch**
- In each epoch, we expect a constant fraction of iterations to be **good** i.e. deviation of good nodes is $\geq \beta$ in correct direction ($\beta = \theta(n)$)

- In a **good** iteration, bad nodes have deviation $\geq \beta/2$
- (Remaining "good" deviation undone by scheduler)

Bad deviation

In an epoch with no agreement, there is a set of $\theta(n)$ iterations I and a set of at most t nodes B such that:

$$\sum_{i\in I}\sum_{p\in B}(\text{deviation of node p in iteration i}) = \Omega(n^2)$$

Spectral Blacklisting

Matrix

- M is a m by n matrix
- M(i,j) = deviation in iteration i of node j
- M_b is bad columns of M
- M_g is good columns of M
- Assume M = [M_b M_g]

Algorithm Sketch

Repeat until reaching agreement

1. Run an epoch. Let M be the deviation matrix for that epoch
2. If |M| is "sufficiently large" then
   A. Compute the right eigenvector, **r**, of M
   B. Increase **bad** value of each node i by **r**[i]^2
3. Blacklist a node when its **bad** value reaches 1

$|M_b| \geq C\ |M_g|$

- **Lemma 1:** In an epoch with no agreement, whp, for any constant C, for $t=c_1n$ chosen sufficiently small, $|M_b| \geq C\ |M_g|$
  - **Fact 1:** Whp $|M_g| = O(n)$
    "sufficiently large"
  - **Fact 2:** $|M_b| = \Omega(n)$ in such an epoch
  - Lemma 1 then follows by algebra

r_b and r_g

- Let **r** be the top right eigenvector of M
- Let $\mathbf{r}_b$ be the vector such that $\mathbf{r}_b[i] = \mathbf{r}[i]$ for $1 \leq i \leq t$ and all other entries are 0
- Let $\mathbf{r}_g$ be the vector such that $\mathbf{r}_g[i] = \mathbf{r}[i]$ for $t+1 \leq i \leq n$ and all other entries are 0
- Expect $|\mathbf{r}_g|^2$ to be bigger than $|\mathbf{r}_b|^2$

$$\left[\begin{bmatrix} M_b \end{bmatrix} \begin{bmatrix} M_g \end{bmatrix} \right] \begin{bmatrix} r_b \\ \hline r_g \end{bmatrix} = \begin{bmatrix} M_b r_b \\ \hline M_g r_g \end{bmatrix}$$

Lemma 2

Lemma 2: Whp, $|\mathbf{r_g}|^2 < |\mathbf{r_b}|^2/2$

Proof: Assume not. Then $|\mathbf{r_b}|^2 \leq 2/3$

$$
\begin{aligned}
|M_B| &\leq |M| \\
&= \boldsymbol{\ell}^T(M\boldsymbol{r}) \\
&\leq |\boldsymbol{\ell}||M\boldsymbol{r}| \\
&\leq |M_B||\boldsymbol{r_b}| + |M_G||\boldsymbol{r_g}| \\
&\leq |M_B|(|\boldsymbol{r_b}| + (1/C)|\boldsymbol{r_g}|) \\
&\leq |M_B|(\sqrt{2/3} + 1/C) \\
&< |M_B|
\end{aligned}
$$

where the last line holds if $C \geq 5.45$ (i.e. $t \leq .004n$)

Implications

Lemma 2: Whp, $|\mathbf{r}_g|^2 < |\mathbf{r}_b|^2 /2$

So, whp, **bad** values for bad nodes increase at twice the rate as **bad** values for good nodes

Thus "most" good nodes:

1) Blacklist no more than t good nodes

2) Blacklist all bad nodes within n epochs

Conclusion

- First expected fully polynomial time algorithm for classic Byzantine agreement
- Previous best algorithm (Ben-or's) was expected exponential time
- New technique: design algorithms that force attackers into **statistically** deviant behavior that is detectable

Open Problems

- Can we use spectral blacklisting in problems where an adversary is trying to attack reputations or page rank?
- Can we learn bad nodes faster via different scoring e.g. weighted majority?
- Connections to planted clique type problems?
- Improve latency, resilience, and bandwidth

Questions?

(D)etector/(N)eutralizer Game

1. N claims columns, provided total claimed over game $\leq$ t
2. Entries in unclaimed columns set to sum of n indep coinflips
3. Each row selected indep. with prob. 1/2
4. N sets all entries in its columns
5. D sees matrix & may remove columns provided total removed over game $\leq$ 2t

N's goal: Deviation of all "selected" rows $\leq$ 2n
D wins if N fails in its goal

Our result: Win for D in expected O(n) iterations

(D)etector/(N)eutralizer Game

1. N claims columns, provided total claimed over game $\leq$ t
2. Entries in unclaimed columns set to sum of n indep coinflips
3. Each row selected indep. with prob. 1/2
4. N sets all entries in its columns
5. D sees matrix & may remove columns provided total removed over game $\leq$ 2t

Related Work (Spectral)

- Page Rank
- Eigentrust
- Hidden Clique

Page Rank [PBMW '99]

- Google's $300 billion "secret sauce"
- M is a stochastic matrix, representing a random walk over the web link graph
- **r** is top right eigenvector of M (and stationary distribution of M's walk)
- For a web page, i, **r**[i] = "authority" of i

Eigentrust [KSG '03]

- M is a matrix s.t. M(i,j) represents amount which party i trusts party j
- **r** is top right eigenvector of M
- For a party, i, **r**[i] = "trustworthiness" of i
- Party i is trustworthy if it is trusted by parties that are themselves trustworthy

Differences

- Eigentrust and PageRank: Want to identify good players based on **feedback from other players**
- D/N Game: Want to identify bad players based on **deviation from random coinflips**

Hidden Clique

- The problem
  - A random G(n,1/2) graph is chosen
  - A k-clique is randomly placed in G
- [AKS '98] give an algorithm for $k = \sqrt{n}$
  1. **v** is second eigenvector of adj. matrix of G
  2. W is top k vertices sorted by abs. value in **v**
  3. Returns all nodes with 3k/4 neighbors in W

Differences

- Hidden Clique: Matrix entries are 0 and 1; Want to find submatrix that is **all** 1's
- D/N Game: Matrix entries in [-n,+n]. Want to find submatrix where **sum** of each row has high absolute value

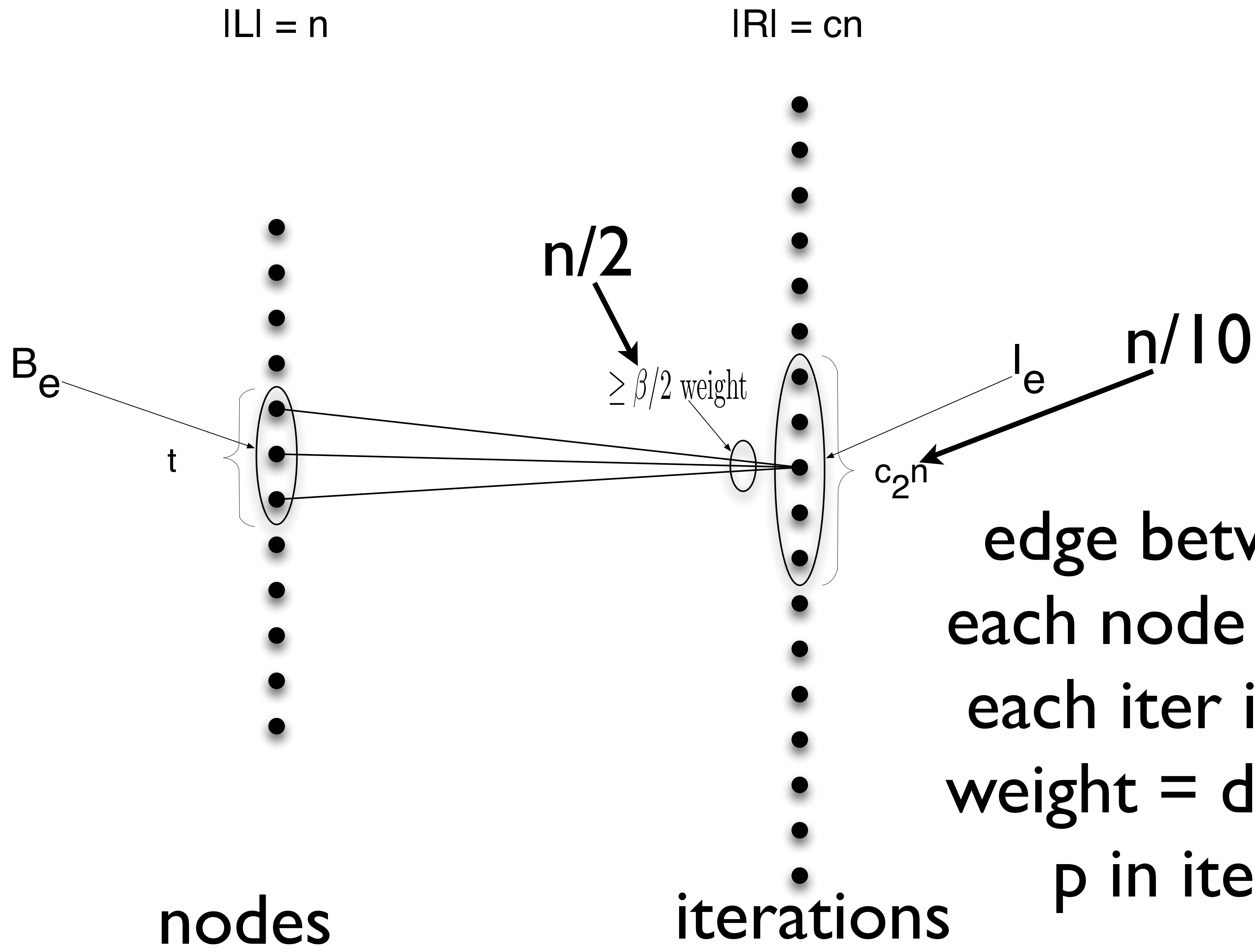
Reliable Broadcast (Bracha)

- All coinflip values sent using reliable broadcast
- Ensures if a message is "received" by a good node, same message is eventually "received" by all nodes
- Prevents equivocation
- Doesn't solve BA
  - If a bad player reliably broadcasts, may be case that **no** good player "receives" the message

Common Coins

- There are at least n(n-2t) **common coins** and no more than 2t coins from good nodes, no more than 2 per node that are not common
- The common coins are known to n-4t good nodes

Bipartite Graph



edge between each node p and each iter i with weight = dvtn of p in iter i

$|M_g|$

Fact 1: Whp, $|M_g| \leq 5(n(m+n))^{1/2}$

- M_g is a random matrix
- Each entry is an independent r.v. with expectation 0; s.d. = $\sqrt{n}$; and range [-k,k] where $k \sim n^{1/2} \log n$
- Fact 1 follows from Theorem 3 in [AS '07]

$|M_b|$

Fact 2: $|M_b| \geq (mn)^{1/2}/(2c_1)$ (where $t = c_1 n$)

- **x** is a unit vector with all values $1/t^{1/2}$
- **y** is a unit vector with entries $\pm 1/(m/10)^{1/2}$ for the m/10 good iterations and 0 everywhere else (sign of non-zero entries is direction of bad deviation)
- Then $\mathbf{y}^t M_b x \geq (mn/20)/(mt/10)^{1/2} \geq (mn)^{1/2}/(2c_1)$

When to update bad values

- Some good nodes may not receive the coinflips of the bad nodes in a given epoch
- If $|M| \leq (mn)^{1/2} /(2c_1)$ then don't do **bad** updates (recall $t = c_1 n$)
- If there is no agreement, a linear number of good nodes will perform updates

Deviation Probabilities