Distributed Network Tomography: Exact Recovery with Adversarial, Heterogeneous and Sporadic Data

Gugan Thoppe *(IISc, Bengaluru)*

Joint work with

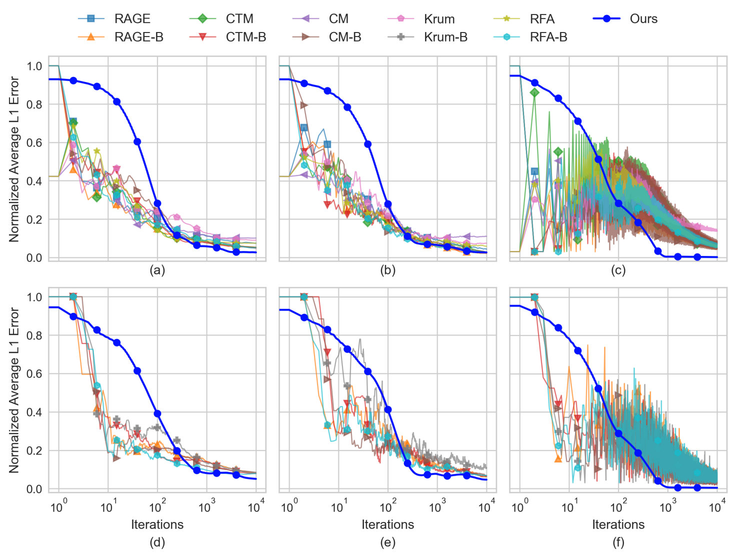
Swetha G., Nibedita R., Mihir D, Naman *(IISc, Bengaluru)*

Vishal H., Alexandre R., Alexandre Azor *(IMT Atlantique, France)*

Talk Highlights

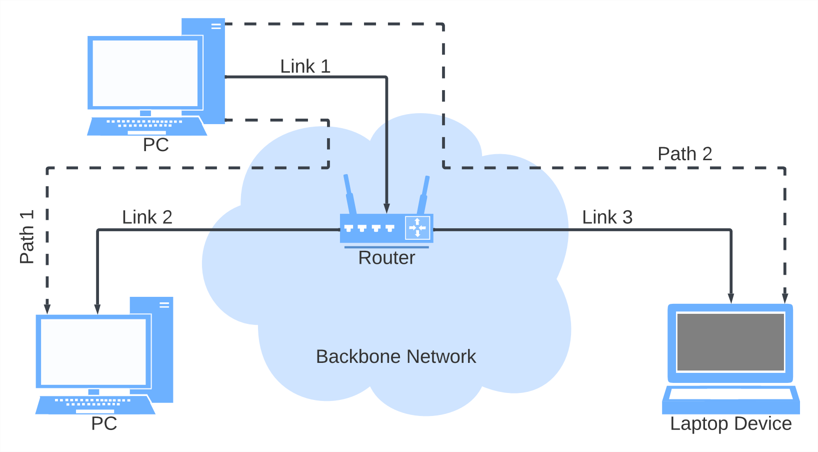
- Network Tomography as Distributed System of Linear Equations - **Adversarial**, **Heterogeneous**, and **Sporadic** Measurements
- Limitations of existing adversary-resilient approaches
- Novel ℓ_1**-minimization**-based algorithm
- $O(1/\sqrt{n})$ convergence rate
- Simulation Results

Preview of Simulation Results



Motivation and Problem Formulation

Network Tomography



Network Tomography

- Network Administrator's Goals: Diagnose and fix Issues
  - Isolate a problem source
  - Allocate resources to address the problem
- Example: Identify links with high latency or packet loss
- Challenge: Link level information cannot be sampled
- Alternative: Use end-to-end path-level measurements

Network Tomography

- Network Administrator's Goals: Diagnose and fix Issues
  - $\boxed{\text{Isolate a problem source}}$
  - Allocate resources to address the problem
- Example: Identify links with **high latency** or **packet loss**
- Challenge: Link level information **cannot** be sampled
- Alternative: Use **end-to-end** path-level measurements

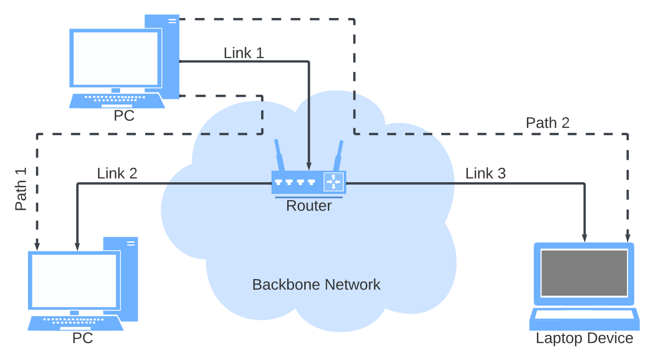
Delay Tomography: Problem Formulation

- $Z(k)$: delay on link k and $Y(j)$: delay on path $\mathcal{P}_j$
- Under the additivity assumption, $Y(j) = \sum_{k \in \mathcal{P}_j} Z(k)$
- Joint relation: $Y = PZ$, where

  $Z \equiv (Z(1), \ldots, Z(d))^\top$ and $Y \equiv (Y(1), \ldots, Y(N))^\top$

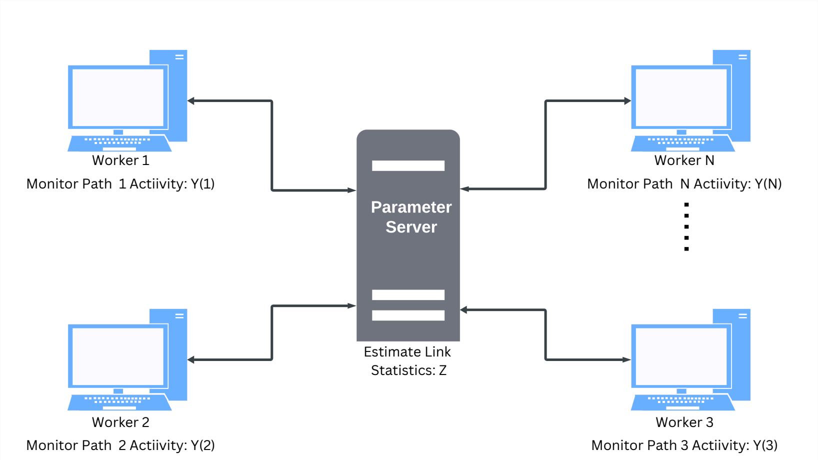
  $P \equiv (a_{jk})$ with $a_{jk} = 1$ if link $k \in \mathcal{P}(j)$
- **Estimate $\mathbb{E}[Z]$ using IID samples of $Y(1), \ldots, Y(N)$**

Network Tomography

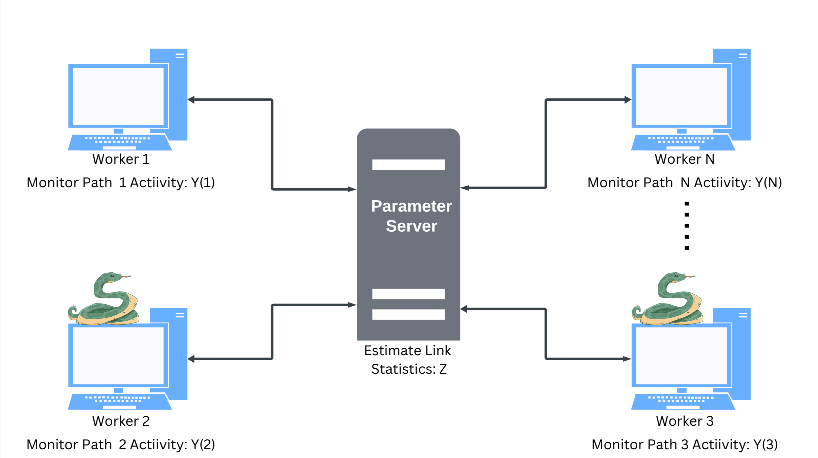


- $Y(1) = Z(1) + Z(2)$ and $Y(2) = Z(1) + Z(3)$
- $Y = PZ$, where $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

Distributed Learning Formulation



Learning Amidst Frenemies



The Rise of Adversaries

Adversaries could arise when a subset of workers wish to

1. Disrupt services
2. Hide illicit activities
3. Mislead traffic management
4. Sabotage competitors

Existing Adversary-resilient Approaches: A Survey

Problem Formulation

- Setup: Parameter-server and (possibly) adversarial workers
- Joint goal: **min.** $f(x)$, where

$$f(x) = \frac{1}{N} \sum_{j=1}^{N} f_j(x)$$

- Workers can obtain (noisy) estimates of $\nabla f_j(x)$
- Within network tomography, e.g., $f_j(x) = (p_j^\top x - \mathbb{E}Y(j))^2$, where $p_j^\top$ is the j-th row of P and $\mathbb{E}Y(j)$ is the j-th coordinate of $\mathbb{E}Y(j)$

Naive Approach (without Adversaries): Aggregation

- Each worker j shares an estimate $g_n^j \equiv \nabla f_j(x_n, \xi_{n+1})$ of $\nabla f_j(x_n)$
- Server computes $g_n = \sum_{j=1}^N g_n^j / N$ and then updates x_n using

$$x_{n+1} = x_n - \alpha_n g_n$$

Convergence Rate: Naive Approach [Wang et al., 2023]

- Suppose the following assumptions hold:
  - f is strongly convex
  - ∇f_j is Lipschitz continuous
  - $\mathbb{E}\|g^j(x) - \nabla f_j(x)\|^2 \leq \sigma^2(1 + \|x - x_*\|^2)$
  - Stepsize $\alpha_n = c/n$
- Then,

$$\mathbb{E}\|x_n - x_*\|^2 = O\left(\frac{1}{\mathbf{N}n}\right)$$

Classification of Existing Adversary-resilient Approaches

1. Data encoding
2. Filtering
3. Homogenization

Data Encoding

1. [Chen et al., 18], [Data et al., 2019, 2020]
2. Each worker j estimates some **function of $\nabla f_1(x_n), \ldots, \nabla f_N(x_n)$**
3. These functions **incorporate redundancy** to enable the parameter server to reliably **reconstruct $\nabla f(x_n)$**
4. Within network tomography, this approach would force each worker to process **samples of multiple Y-coordinates**
5. All workers would need to share their estimates **synchronously**

Filtering

- **Synchronous**: Robust Aggregator [Data21, Pillutla22]
- **Asynchronous**:
  - Private Data [Xie20, Fang22]
  - Lipshitz filter [Damaskinos18]
  - Asynchronous worker, Synchronous server updates [Yang21]
- Within network tomography, private data approach is infeasible since the server would need **true path measurements**
- Other approaches: Convergence to $O(\zeta^2)$, where

$$\mathbb{E}\|\nabla f_j(x) - \nabla f(x)\|^2 \leq \zeta^2$$

Filtering: Robust aggregation

- Each worker only shares an estimate g_n^j of $\nabla f_j(x_n)$
- Server computes a robust aggregate $g = \mathcal{F}(g_n^1, \ldots, g_n^N)$, where $\mathcal{F}$ could be

  - coordinate-wise median,

  - coordinate-wise trimmed mean,

  - geometric median, etc.

Asynchronous Worker, Synchronous Server-side Updates

- Form B buckets of workers
- Wait until ≥ 1 worker in each bucket provides an estimate
- Take average of received estimates in Bucket j to output h_n^j
- Server computes $h_n = \mathcal{F}(h_n^1, \ldots, h_n^B)$ and then updates using

$$x_{n+1} = x_n - \alpha_n h_n$$

Homogenization

- Presumes synchronous workers
- Randomly permute workers and then form B buckets of workers
- Take average of received estimates in Bucket j to output h_n^j
- Server computes $h_n = \mathcal{F}(h_n^1, \ldots, h_n^B)$ and then updates using

$$x_{n+1} = x_n - \alpha_n h_n$$

- Promises exact recovery if $K^2 = O(1/\delta)$ and

$$\mathbb{E}_{j \sim \mathcal{G}} \|\nabla f_j(x) - \nabla f(x)\|^2 \leq K^2 \|\nabla f(x)\|^2$$

Proposed ℓ_1-based Algorithm

Initial Thoughts

- Suppose $b = Ax_*$
- **Question**: How to recover x_*?
- **Case I**: A and b known
  - Multiple algorithms
  - **Exact recovery**: A has full column rank

Intermediate Thoughts

- **Case II**: A and $b' = b + e$ known, where e is m-**sparse**
  - Smart idea: Solve $\min \|Ax - b'\|_1$
  - Exact Recovery [FTD11]: A is **robust**, i.e.,

    for each $x \in \mathbb{R}^d \setminus 0$ and each $S \subseteq \{1, \ldots, N\}$ with $|S| \leq m$

$$\sum_{i \in S^c} |a_i^\top x| > \sum_{i \in S} |a_i^\top x|,$$

    where $a_i^\top$ is the i-th row of A.

[FTD11]: Fawzi, Tabuada, and Diggavi., Secure state-estimation for dynamical systems under active adversaries, Allerton '11

Examples of Robust Matrices

- $A = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

- $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 1 & 2 \\ -2 & 1 \end{bmatrix}$

Extension to Network Tomography

- Recall that Z is the vector of link-level measurements
- Identify matrices A and B such that

$$P\mathbb{E}Z = AB\mathbb{E}Z$$

 and A **is robust**

- Solve $\min \|Ax - \mathbb{E}Y\|_1$ to recover $B\mathbb{E}Z$, presuming access only to IID samples of Y-coordinates in an asynchronous fashion.

Proposed Algorithm to Estimate $\mathbb{E}X$: Pseudocode

Initialize $x_0 \in \mathbb{R}^d$ at server and $y_0(i)$ at worker i

for $n \geq 0$ **do**

Server

Sample index $i_{n+1} \in \{1, \ldots, N\}$ uniformly randomly

Send x_n to agent i_{n+1}

Worker i_{n+1} (if honest)

Send $\text{sign}(y_n(i_{n+1}) - a_{i_{n+1}}^\top x_n)$ to server

$y_{n+1}(i_{n+1}) = y_n(i_{n+1}) + \beta_n \left[Y_{n+1}(i_{n+1}) - y_n(i_{n+1})\right]$

\\\\ $i_{n+1} = i$ *implies* $Y_{n+1}(i_{n+1}) \sim Y(i)$

Server

$x_{n+1} = \Pi_{\mathcal{X}} \left(x_n + \alpha_n \, \text{sign}(y_n(i_{n+1}) - a_{i_{n+1}}^\top x_n) \, a_{i_{n+1}}\right)$

end for

Convergence Rates

Our Main Result

Assumptions

1. **Target Vector**: Z has finite mean and finite covariance entries
2. **Observation Matrix**: A is robust
3. **Stepsizes**: $\alpha_n = 1/\sqrt{n+1}$ and $\beta_n = 1/(n+1)$.

Conclusion: Let $g(x) = \frac{1}{N}\|Ax - \mathbb{E}Y\|_1$. Then, for $r \in (0,1)$ and $i = \lceil rn \rceil$,

$$\mathbb{E}g(\bar{x}_i^n) = O\left(\frac{1}{\sqrt{n}}\right),$$

where

$$\bar{x}_i^n = \sum_{j=i}^{n} \bar{\alpha}_k x_j \qquad \text{and} \qquad \bar{\alpha}_j = \frac{\alpha_j}{\sum_{k=i}^{n} \alpha_k}$$

Proof Sketch - I

- For $E_n := \mathbb{E}\|x_n - B\mathbb{E}Z\|_2^2$

$$E_{n+1} \leq E_n + 2\alpha_n \mathbb{E}[(x_n - \mathbb{E})^\top (g_n + \epsilon_n)] + \alpha_n^2 \bar{A},$$

where

$$g_n = \frac{1}{N}\left[\sum_{i \in \mathcal{H}} \mathrm{sign}(\mathbb{E}Y(i) - a_i^\top x_n) a_i + \sum_{i \in \mathcal{A}} \mathrm{sign}(y_n(i) - a_i^\top x_n) a_i\right]$$

$$\epsilon_n = \frac{1}{N} \sum_{i \in \mathcal{H}} \left[\mathrm{sign}(y_n(i) - a_i^\top x_n) - \mathrm{sign}(\mathbb{E}Y(i) - a_i^\top x_n) a_i\right]$$

Proof Sketch - II

- Robustness of A implies

$$\mathbb{E}[(x_n - \mathbb{E})^\top g_n] \leq \frac{1}{K}\mathbb{E}(x_n - \mathbb{E}X)^\top g'_n,$$

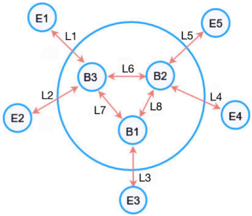
  where $g'_n = \frac{1}{N}\sum_{i=1}^{N} \text{sign}(\mathbb{E}Y(i) - a_i^\top x_n)a_i$ is the true sub-gradient

- Since $y_n(i) \to \mathbb{E}Y(i)$ for all $i \in \mathcal{H}$,

$$\mathbb{E}[(x_n - \mathbb{E})^\top \epsilon_n] = O\left(\frac{1}{\sqrt{n}}\right)$$

Empirical Simulations

Network Setup



(a) A simple network example

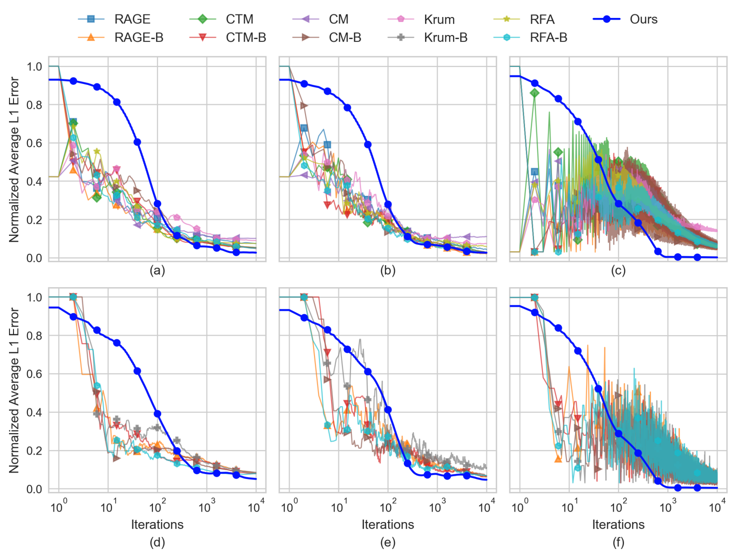
$$P := \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(b) Matrix P

$$A := \begin{bmatrix} 2 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{bmatrix}$$

(c) Matrix A

Simulation Results



Conclusions

- Novel ℓ_1-**minimization**-based approach for exact recovery with adversarial, asynchronous, and heterogeneous data
- Convergence rate: $O(1/\sqrt{n})$
- Empirically demonstrated higher accuracy

Future Directions

- Automate A-matrix design
- Extend to tracking
- Extend to general optimization