On Solving Linear Systems in Sublinear Time

Alexandr Andoni, Columbia University
Robert Krauthgamer, Weizmann Institute
Yosef Pogrow, Weizmann Institute → Google

WOLA 2019

Solving Linear Systems

- Input: $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$
- Output: vector x that solves $Ax = b$

- **Many algorithms, different variants:**
  - Matrix A is sparse, Laplacian, PSD etc.
  - Bounded precision (solution x is approximate) vs. exact arithmetic

- **Significant progress:** Linear system in **Laplacian** matrix L_G can be solved approximately in near-linear time $\tilde{O}(\mathrm{nnz}(L_G) \cdot \log \frac{1}{\epsilon})$ [Spielman-Teng'04, …, Cohen-Kyng-Miller-Pachocky-Peng-Rao-Xu'14]

Our focus: <u>Sublinear</u> running time

Sublinear-Time Solver

- Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ (also $\epsilon > 0$) and $i \in [n]$
- Output: approximate coordinate $\hat{x}_i$ from (any) solution x^* to $Ax = b$
  - Accuracy bound $\|\hat{x} - x^*\|_\infty \le \epsilon\|x^*\|_\infty$

- Formal requirement: There is a solution x^* to the system, such that

$$\forall i \in [n], \qquad \Pr\left[\left|\hat{x}_i - x_i^*\right| \le \epsilon\|x^*\|_\infty\right] \ge \frac{3}{4}$$

- Follows framework of Local Computation Algorithms (LCA), previously used for graph problems [Rubinfeld-Tamir-Vardi-Xie'10]

Motivation

- Fast quantum algorithms for solving linear systems and for machine learning problems [Harrow-Hassidim-Lloyd'09, …]
  - Can we match their performance classically?
  - Recent success story: quantum $\rightarrow$ classical algorithm [Tang'18]

- New direction in sublinear-time algorithms
  - "Local" computation in numerical problems
  - Compare computational models (representation, preprocessing), accuracy guarantees, input families (e.g., Laplacian vs. PSD)
  - Known quantum algorithms have modeling requirements (e.g., quantum encoding of b)

Algorithm for Laplacians

- Informally: Can solve Laplacian systems of bounded-degree expander in polylog(n) time
  - Key limitations: sparsity and condition number

- Notation:
  - $L_G = D - A$ is the Laplacian matrix of graph G
  - L_G^+ is its Moore-Penrose pseudo-inverse

- Theorem 1: Suppose the input is a d-regular n-vertex graph G, together with its condition number $\kappa > 0$, $b \in \mathbb{R}^n$, $u \in [n]$ and $\epsilon > 0$.
  Our algorithm computes $\hat{x}_u \in \mathbb{R}$ such that for $x^* = L_G^+ b$,
  $$\forall u \in [n], \qquad \Pr[|\hat{x}_u - x_u^*| \le \epsilon \|x^*\|_\infty] \ge \frac{3}{4},$$
  and runs in time $\tilde{O}(d\epsilon^{-2}s^3)$ for $s = \tilde{O}(\kappa \log n)$.

More inputs? Faster?

Some Extensions

- Can replace n with $\|b\|_0$
  - Example: Effective resistance can be approximate (in expanders) in constant running time!

$$R_{\text{eff}}(u,v) = (e_u - e_v)^T L_G^+ (e_u - e_v)$$

- Improved running time if
  - Graph G is preprocessed
  - One can sample a neighbor in G, or

- Extends to Symmetric Diagonally Dominant (SDD) matrix S
  - κ is condition number of $D^{-1/2} S D^{-1/2}$

Lower Bound for PSD Systems

- Informally: Solving "similar" PSD systems requires polynomial time
  - Similar = bounded condition number and sparsity
  - Even if the matrix can be preprocessed

- Theorem 2: For certain invertible PSD matrices S, with bounded sparsity d and condition number κ, every randomized algorithm must query $n^{\Omega(1/d^2)}$ coordinates of the input b.

- Here, the output is $\hat{x}_u \in \mathbb{R}$ for a fixed $u \in [n]$, required to satisfy

$$\forall u \in [n], \qquad \Pr\left[|\hat{x}_u - x_u^*| \le \frac{1}{5}\|x^*\|_\infty\right] \ge \frac{3}{4},$$

for $x^* = S^{-1}b$.
- In particular, S may be preprocessed

Dependence on Condition Number

- Informally: Quadratic dependence on κ is necessary
  - Our algorithmic bound $\tilde{O}(\kappa^3)$ is near-optimal, esp. when matrix S can be preprocessed

- Theorem 3: For certain graphs G of maximum degree 4 and <u>any</u> condition number $\kappa > 0$, <u>every randomized algorithm</u> (for L_G) with accuracy $\epsilon = \frac{1}{\log n}$ must probe $\tilde{\Omega}(\kappa^2)$ coordinates of the input b.

- Again, the output is $\hat{x}_u \in \mathbb{R}$ for a fixed $u \in [n]$, required to satisfy

$$\forall u \in [n], \qquad \Pr\left[|\hat{x}_u - x_u^*| \le \frac{1}{\log n}\|x^*\|_\infty\right] \ge \frac{3}{4},$$

  for $x^* = L_G^+ b$.
- In particular, G may be preprocessed

Algorithmic Techniques

- Famous Monte-Carlo method of von Neumann and Ulam:

  Write matrix inverse by power series

  $$\forall \|X\| < 1, \qquad (I - X)^{-1} = \sum_{t \geq 0} X^t$$

  then estimate it by random walks (in X) with unbiased expectation

- Inverting a Laplacian $L_G = dI - A$ corresponds to summing walks in G
  - For us: view $e_u^T \sum_{t \geq 0} A^t b$ as sum over all walks, estimate it by sampling (random walks)

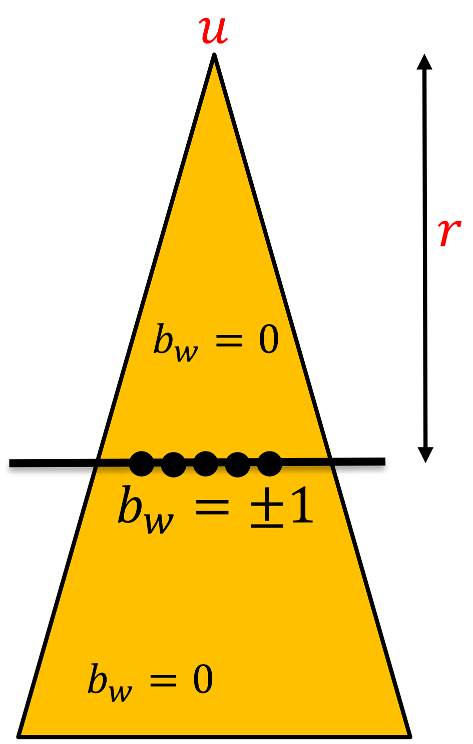
- Need to control: number of walks and their length
  - Large powers $t > t^*$ contribute relatively little (by condition number)
  - Estimate truncated series ($t \leq t^*$) by short random walks (by Chebyshev's inequality)

Related Work – All Algorithmic

- Similar techniques were used before in related contexts but under different assumptions, models and analyses:
  - Probabilistic log-space algorithms for approximating L_G^+ [Doron-Le Gall-Ta-Shma'17]
    - Asks for entire matrix, uses many long random walks (independent of κ)
  - Local solver for Laplacian systems with boundary conditions [Chung-Simpson'15]
    - Solver relies on a different power series and random walks
  - Local solver for PSD systems [Shyamkumar-Banerjee-Lofgren'16]
    - Polynomial time $\mathrm{nnz}(S)^{2/3}$ under assumptions like bounded matrix norm and random $u \in [n]$
  - Local solver for Pagerank [Bressan-Peserico-Pretto'18, Borgs-Brautbar-Chayes-Teng'14]
    - Polynomial time $\mathrm{O}(n^{2/3})$ and $\mathrm{O}((\mathrm{nd})^{1/2})$ for certain matrices (non-symmetric but by definition are diagonally-dominant)

Lower Bound Techniques

- PSD lower bound: Take Laplacian of $2d$-regular expander but with:
  - high girth,
  - edges signed ± 1 at random, and
  - $O(\sqrt{d})$ on the diagonal (PSD but not Laplacian)
- The graph looks like a tree locally
  - Up to radius $\Theta(\log n)$ around u



- Set $b_w = \pm 1$ for w at distance r, and 0 otherwise
  - Signs have small bias $\delta \approx d^{-r/2}$
  - Recovering it requires reading $\Omega(\delta^{-2})$ entries
- Using inversion formula, $x_u \approx$ average of b_w's
- Condition number lower bound: Take two 3-regular expanders connected by a matching of size n/κ
  - Let $b_w = \pm 1$ with slight bias inside each expander

Further Questions

- Accuracy guarantees
  - Different norms?
  - Condition number of S instead of $D^{-1/2}SD^{-1/2}$?
- Other representations (input/output models)?
  - Access the input b via random sampling?
  - Sample from the output x?
- Other numerical problems?

Thank You!