Solving SDD Linear Systems in Nearly-Linear Time

M. Trefois   J.-C. Delvenne   M. Schaub*   P. Van Dooren

Université catholique de Louvain

*and Université de Namur

Benelux Meeting on Systems and Control
March 24, 2015

Goal:

Solve

$$Av = b$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric and diagonally-dominant (SDD):

$$a_{ii} \geq \sum_{j \neq i} |a_{ij}|.$$

Several classical methods:

- Gauss elimination: time $\mathcal{O}(n^3)$
- Fast matrix inversion (Strassen 1969, Coppersmith-Winograd 1987, ...): time $\mathcal{O}(n^{2.37})$
- ...

$\Rightarrow$ Too slow in case of huge matrix A

Revised goal I:

Solve any SDD system

$$Av = b$$

in nearly-linear time, i.e. in

$$\mathcal{O}(m \log^c n) \text{ time,}$$

where

- n is the size of the system
- m is the number of nonzero entries in A

Particular case of interest: when A is a Laplacian matrix.

Laplacian systems for...

- solving any SDD linear system

but also...

- computing effective resistances in a network
- computing dominant eigenvectors of graphs (by inverse power method)
- ...

Outline

Laplacian systems: definition

Overview of Kelner, Orecchia, Sidford and Allen Zhu's method

Kelner et al's method: Step 1

Running time

Conclusion

Laplacian systems: definition

Overview of Kelner, Orecchia, Sidford and Allen Zhu's method

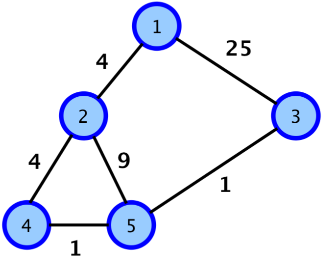
Kelner et al's method: Step 1

Running time

Conclusion

Laplacian systems: definition

G : undirected graph: n nodes, m edges
positive weights along the edges



Laplacian matrix:

$$L = \begin{vmatrix} 29 & -4 & -25 & 0 & 0 \\ -4 & 17 & 0 & -4 & -9 \\ -25 & 0 & 26 & 0 & -1 \\ 0 & -4 & 0 & 5 & -1 \\ 0 & -9 & -1 & -1 & 11 \end{vmatrix}$$

The Laplacian matrix is SDD .

Laplacian systems: definition

Revised Goal II:

Solve the Laplacian system

$$Lv = b$$

in nearly-linear time, namely in time $\mathcal{O}(m \log^c n)$.

More precisely, given $\epsilon > 0$, find $v_K \in \mathbb{R}^n$ such that

$$||v_K - v_{opt}||_L \leq \epsilon \cdot ||v_{opt}||_L,$$

where $Lv_{opt} = b$.

Laplacian systems: definition

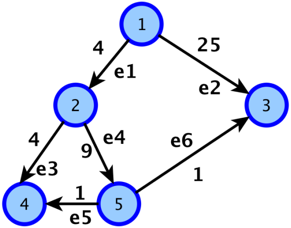
Overview of Kelner, Orecchia, Sidford and Allen Zhu's method

Kelner et al's method: Step 1

Running time

Conclusion

Kelner et al's method: overview



$$B = \begin{array}{c|ccccc|} & \color{red}{1} & \color{red}{2} & \color{red}{3} & \color{red}{4} & \color{red}{5} \\ \color{red}{e_1} & 1 & -1 & 0 & 0 & 0 \\ \color{red}{e_2} & 1 & 0 & -1 & 0 & 0 \\ \color{red}{e_3} & 0 & 1 & 0 & -1 & 0 \\ \color{red}{e_4} & 0 & 1 & 0 & 0 & -1 \\ \color{red}{e_5} & 0 & 0 & 0 & -1 & 1 \\ \color{red}{e_6} & 0 & 0 & -1 & 0 & 1 \end{array} \quad R = \begin{vmatrix} 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Factorize L as:

$$L = B^T R^{-1} B$$

Kelner et al's method: overview

Laplacian System:

$$B^T R^{-1} B v = b$$

Set $f = R^{-1} B v$.

- Step 1: Find f_K an approximation of the unique solution $f_{opt} \in \mathbb{R}^m$ of minimal R-norm to

$$B^T f = b.$$

- Step 2: given f_K and $\epsilon > 0$, compute $v_K \in \mathbb{R}^n$ such that

$$||v_K - v_{opt}||_L \leq \epsilon \cdot ||v_{opt}||_L,$$

where $L v_{opt} = b$.

Kelner et al's method: overview

Laplacian System:

$$B^T R^{-1} B v = b$$

Set $f = R^{-1} B v$.

- Step 1: Find f_K an approximation of the unique solution $f_{opt} \in \mathbb{R}^m$ of minimal R-norm to

$$B^T f = b.$$

- Step 2: given f_K and $\epsilon > 0$, compute $v_K \in \mathbb{R}^n$ such that

$$||v_K - v_{opt}||_L \leq \epsilon \cdot ||v_{opt}||_L,$$

where $L v_{opt} = b$.

Kelner et al's method: Step 1

$f_{opt} \in \mathbb{R}^m$ is the solution to $B^T f = b$ of minimal R-norm, namely

- $B^T f_{opt} = b$
- f_{opt} is orthogonal to the kernel of B^T

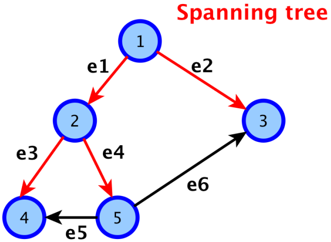
Revised Goal III:

> given $\epsilon > 0$, find $f_K \in \mathbb{R}^m$ such that $B^T f_K = b$ and
>
> $$||f_K - f_{opt}||_R \leq \epsilon \cdot ||f_{opt}||_R$$
>
> in nearly-linear time.

Needed:

- A basis of the kernel of B^T $\Rightarrow$ spanning tree
- An iterative algorithm $\Rightarrow$ a coordinate descent method

A basis of the kernel of B^T

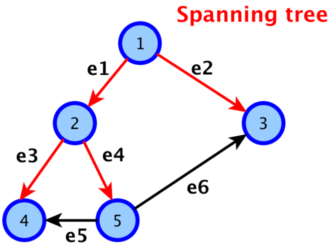


Any off-tree edge $e = \{a, b\}$ defines a unique cycle of the form:

$$e + \text{ path between } a \text{ and } b \text{ in } T$$

For edge e_5: cycle $= \{e_3, e_4, e_5\} \Rightarrow$ define cycle vector $Q_{e_5} = \begin{vmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 1 \\ 0 \end{vmatrix}$

A basis of the kernel of B^T



The cycle vectors Q_e's where e is an off-tree edge form a basis of the kernel of B^T

In our example, a basis is:

$$Q_{e_5} = \begin{vmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 1 \\ 0 \end{vmatrix} \qquad Q_{e_6} = \begin{vmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 1 \end{vmatrix}$$

An iterative algorithm: coordinate descent

We want to minimize

$$\mathbb{R}^m \to \mathbb{R} : f \mapsto ||f||_R$$

under the constraint $B^T f = b$.

Coordinate Descent Method:

- start with $f_0 \in \mathbb{R}^m$ such that $B^T f_0 = b$
- search directions: the cycle vectors Q_e's
- at each iteration:
  - ⋆ pick randomly a cycle vector Q_e
  - ⋆ find

$$\alpha^\star := \arg\min_{\alpha} ||f_k + \alpha Q_e||_R = -\frac{f_k^T R Q_e}{Q_e^T R Q_e}$$

  So, the iterations become:

$$f_{k+1} = f_k - \frac{f_k^T R Q_e}{Q_e^T R Q_e} \cdot Q_e$$

An iterative algorithm: coordinate descent

- Start with f_0 such that $B^T f_0 = b$
- Recursive step:

$$f_{k+1} = f_k - \frac{f_k^T R Q_e}{Q_e^T R Q_e} \cdot Q_e,$$

with Q_e in the kernel of B^T.

$$\Rightarrow \text{For any } k, \quad B^T f_k = b.$$

Convergence rate: depends on the spanning tree
Number of iterations: always at least in $\mathcal{O}(m)$

Wanted:
Running time in $\mathcal{O}(m \log^c n) \Rightarrow$ each iteration in logarithmic time

An iterative algorithm: coordinate descent

Recursive step:

$$f_{k+1} = f_k - \frac{f_k^T R Q_e}{Q_e^T R Q_e} \cdot Q_e$$

Wanted: each iteration in logarithmic time

Problem: support of Q_e in $\mathcal{O}(n)$

Kelner *et al.* solve this as a data-structure problem.

Question: can we understand this purely from a Linear Algebra perspective ?

An iterative algorithm: coordinate descent

Recursive step:

$$f_{k+1} = f_k - \frac{f_k^T R Q_e}{Q_e^T R Q_e} \cdot Q_e$$

Wanted: each iteration in logarithmic time

Our contribution:

work with two different bases G_1, G_2 such that

- in G_1 and G_2, any Q_e is $\mathcal{O}(\log n)$-sparse
- given f_k and Q_e in bases G_1 and G_2, compute $f_k^T R Q_e$ in $\mathcal{O}(\log n)$ time.

An iterative algorithm: coordinate descent

Any iteration: in $\mathcal{O}(\log n)$ time

Convergence rate: depends on the spanning tree

Number of iterations: always at least in $\mathcal{O}(m)$

But, is the running time nearly-linear ???

Running time

Theorem (Abraham-Neiman, 2012):

One can find in $\mathcal{O}(m \log n)$ time a spanning tree such that the number of iterations is $\mathcal{O}(m \log n \log(n/\epsilon))$.

The running time of Kelner et al's method is:

$\mathcal{O}(m \log^2 n \log(n/\epsilon)) = \sharp$ of iterations $\times\, \mathcal{O}(\log n)$ time per iteration

Conclusion

- Goal: solving any Laplacian system in time which is nearly linear in the number of nonzero entries
- Motivation: solving any SDD system in nearly-linear time
- Kelner et al's method (2013): running time: $\mathcal{O}(m \log^2 n)$
- Our contribution I: understand Kelner et al's method from a Linear Algebra perspective
- Our contribution II: extend the trick to other linear systems

J.A. Kelner, L. Orecchia, A. Sidford, Z. Allen Zhu, A simple, combinatorial algorithm for solving SDD systems in nearly-linear time, in Proceedings of the 45th annual ACM Symposium on Theory Of Computing (STOC), pp. 911-920, New York, NY, USA, 2013.