Masters' Thesis Presentation:
A New View on Cycle Toggling Laplacian Solvers

Hui Han Chin

CMU-SCS-CSD

12/13/17

Introduction

Electrical flow on a graph is a special type of flow which corresponds to the physical interpretation of electrical current induced on a network of conductors when an electrical circuit is set up.

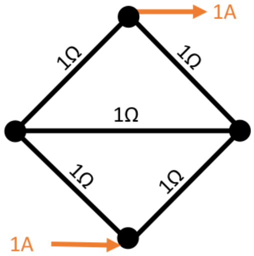


Figure 1: What is the current flow on each edge?

Introduction

Electrical flow has deep connections to spectral graph theory, random walks etc. and is used as an algorithmic primitive in modern algorithm design, resulting in breakthroughs in long-standing problems such as solving the Approximate Maximum Flow problem [CKM$^+$11].

The "Minimum Energy Potential Flow" problem on a weighted undirected graph is to find an electrical flow of minimum energy that satisfies demand on the vertices. This problem can be solved using Laplacian solvers in nearly linear time and there are many vertex potential based solvers that have been developed such as [KMP11, CKM$^+$14].

$$\mathbf{L}v = d$$

Prior Work on Fast Graph Laplacian Solvers

1. Initial breakthrough. [ST06]
   First nearly linear time solver for linear systems of graph Laplacians
2. Potential Based Solver. [KMP11, CKM$^+$14]
   Iterative and recursive scheme that uses preconditioning.
3. Flow Based Solver. [KOSZ13, LS13]
   First flow based solver based on cycle toggling.

Goal

1. Better analysis of the cycle toggling solver and to demystify its practical performance.
2. Better understanding of the primal-dual nature of electrical flows.
3. Establish the connection between the laplacian solvers and stochastic optimization.
4. Incorporate methods from potential solvers into flow solvers.

Outline

Graph Preliminaries I

Definition (Graph)

Let $G = (V, E, W)$ be a weighted, undirected graph with $|V| = n$ vertices, $|E| = m$ edges, and $\forall w \in W, w > 0$ weights. Assumed G is connected

Definition (Conductance, Resistance)

The weights of graph, W, would be termed as conductance. The conductance matrix is a $|E| \times |E|$ diagonal matrix, $\mathbf{C} \in \mathbb{R}^{m \times m}$ where each diagonal entry is the conductance of the edge.
The inverse of the conductance matrix is called the resistance matrix, $\mathbf{R} \in \mathbb{R}^{m \times m}$ and $\mathbf{R} = \mathbf{C}^{-1}$. The conductance matrix can be expressed as $\mathbf{R}^{-1}$

Graph Preliminaries I

Definition (Spanning Tree and Non-Tree edges)

Let the spanning tree of a connected graph G, $T(G) = T$, be a connected subgraph that contains the all the vertices with minimal edges.
Given a spanning tree, the edge set is partitioned into 2 sets: tree edges, denoted as E_t, and non-tree edges, denoted as E_n. There are $m - n + 1$ non-tree edges and for notation convenience, denote this as $m' = m - n + 1$

Graph Preliminaries I

Definition (Stretch and Low Stretch Spanning Tree)

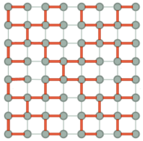
The stretch of an edge $e = (a, b) \in E$ with respect to a tree T is

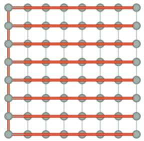
$$st(e) = \frac{\sum_{i \in P_e} r_i}{r_e}$$

where P_e is the unique path of the vertex a to b on the tree T. The induced stretch of a tree on a graph is the sum of the stretch of all the edges,

$$st(T) = \sum_{e \in E} st(e)$$

A Low Stretch Spanning Tree (LSST) is a tree with low total stretch.





Graph Preliminaries I

Definition (Incidence Matrix)

Given an orientation of the edges of G, each edge $(a, b) \in E$ can be expressed as an ordered pair. Orient the edges based on the tree traversal order of a given spanning tree.
Let the incidence matrix $\mathbf{B}$ be the $|V| \times |E|$ matrix whose rows are indexed by vertices and columns are indexed by edges where

$$\mathbf{B}_{v,e} = \begin{cases} 1 & \text{if } e = (u, v)\text{, for some } u \in V, \\ -1 & \text{if } e = (v, u)\text{, for some } u \in V, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Order $\mathbf{B}$ such that the first $n-1$ columns are the edges of the spanning tree. Represent $\mathbf{B}$ as 2 block matrices, $\mathbf{B}_t$, a $n \times (n-1)$ matrix for the tree edges, and $\mathbf{B}_n$, a $n \times (m-n+1)$ matrix for the non-tree edges

$$\mathbf{B} = [\mathbf{B}_t | \mathbf{B}_n]$$

Graph Preliminaries I

Definition (Graph Laplacian)

The graph Laplacian is a $|V| \times |V|$ matrix defined as

$$\mathbf{L}_{i,j} = \begin{cases} \sum_{j \neq i} \mathbf{w}_{ij} & \text{if} \quad i = j, \\ -\mathbf{w}_{ij} & \text{if } i \neq j \text{ and } i \text{ is incident to } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Often, the Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the weighted adjacency matrix. An alternative characterization is by the incidence matrix. It can be verified that

$$\mathbf{L} = \mathbf{BCB}^T$$

When $\mathbf{C}$ has non-negative edge weights, the Laplacian is a positive semi-definite (PSD) matrix. This version of the graph Laplacian is also known as the combinatorial Laplacian or Kirchhoff matrix.

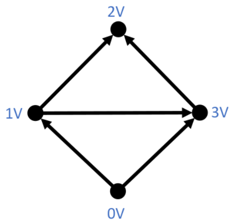
Preliminaries I Summary

Incidence Matrix

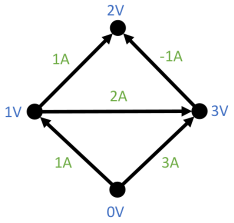
$$\mathbf{B}^T : \text{potential} \to \text{flow}, \quad \mathbf{B} : \text{flow} \to \text{residual}$$

Graph Laplacian

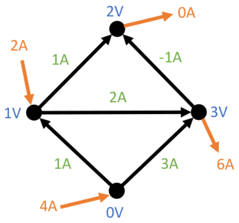
$$\mathbf{L} = \mathbf{BCB}^T$$



v



$\mathbf{CB}^T v$



$\mathbf{BCB}^T v$

Minimal Energy, Demand Satisfying, Potential Flow

Definition

Given a graph $G = (V, E, R)$ and demand d, solve

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & f^T \mathbf{R} f \\ \text{subject to} \quad & \mathbf{B} f = d \end{aligned}$$

The min-energy flow is a potential flow, i.e. $f^* = \mathbf{R}^{-1}\mathbf{B}^T v^*$, so it can be solve using fast laplacian solvers for some potential v.

$$\mathbf{L} v = \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T v = d$$

But fast laplacian solvers have errors in the solution of the potential vector meaning the associated flow would have some error in meeting the demand.

What if we need a flow that meets the demand exactly?

A Cycling Toggling, Flow Based Solver

Theorem

[KOSZ13] A flow based SDD system solver that gives a $1+\epsilon$ approximation in $O(m\log^2 n\log\log n\log(\epsilon^{-1}))$ time.

Key Ideas

1. A good LSST with "tree condition" $\tau = O(m\log n\log\log n)$ can be found using using [AN12].
2. Solving for a flow that meets the demand on a tree is easy and can be done in linear time.
3. However, the flow has too much energy. The optimal potential flow is of minimal energy.
4. Cycle toggle reduces the energy of the flow without changing the demand.
5. Cycle toggle updates can be done efficiently in $O(\log n)$ using a link-cut tree data structure.

KOSZ13 SimpleSolver

KOSZ13 **Algorithm 1:** `SimpleSolver`

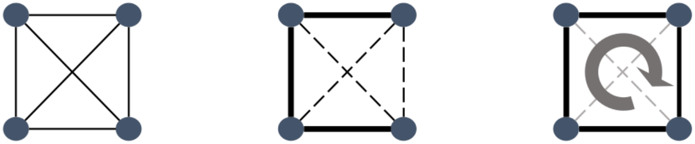
Input: Graph $G = (V, E, R)$ and demand d

Output: Flow that meets demand d, f

1. Find a Low Stretch Spanning Tree, T of graph G.
2. Find a feasible flow that meets the vertices demand on the tree T.
3. Sample each non-tree edge, $e = (a, b)$, with some probability $p_e = \frac{1}{\tau}\frac{R_e}{r_e}$
4. Form cycle, C_e, consisting of e and the path of a to b on T.
5. Minimize the energy of flow on C_e by toggling a circulation of $\frac{\Delta_{C_e}}{R_{C_e}} = \frac{\sum_{i \in C_e} f_i r_i}{\sum_{i \in C_e} r_i}$
6. Repeat for $\tau \log\left(\frac{st(T)\tau}{\epsilon}\right)$ iterations.

KOSZ13 SimpleSolver

Pictorial Guide



"Build Tree - Toggle Cycle"

KOSZ13 SimpleSolver Analysis

Theorem 4.1 (Convergence of `SimpleSolver`)
At each iteration i, the distance to optimal

$$E[|f_i|_R^2] - |f_{opt}|_R^2 \leq (1 - \frac{1}{\tau})(|f_0|_R^2 - |f_{opt}|_R^2)$$

Lemma 6.1 (Initial Energy Bound)

$$|f_0|_R^2 \leq O(\tau)(|f_{opt}|_R^2)$$

Combinining both parts gives a $O(\tau \log(n\epsilon^-1))$ iteration count to get a ϵ approximation to the optimal energy.

KOSZ13 FullSolver

KOSZ13 **Algorithm 5:** `ExampleSolver`
Key Ideas: Tree Weight Scaling + Randomized Stopping Time on `SimpleSolve`
Running Time: $O(m \log^2 n \log\log n \log(\epsilon^{-1} \log n))$

KOSZ13 **Algorithm 6:** `FullSolver`
Key Ideas: Applying Tree Scaling in a sequence for $\log^*(n)$ times
Running Time: $O(m \log^2 n \log\log n \log(\epsilon^{-1}))$

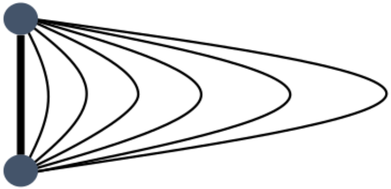
Emperical Studies of KOSZ13

1. Cycle-Toggling Solving is a much simpler algorithm compared to other solvers [KOSZ13].
2. [DGM+16], [BDG16] have shown in practice, despite various implementation optimizations such as parallelization and efficient data structures, KOSZ is not competitive with existing potential based techniques such as Conjugate Gradient or tree-preconditioned Conjugate Gradient.
3. While KOSZ13 has good asympotatical performance, more analysis is needed to understand the effects of the "hidden constants" at various stages of the algorithm.

Improved Cycle Toggling

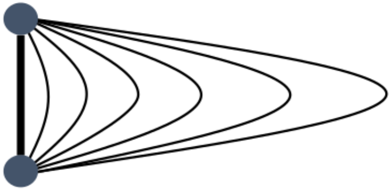
Can we improve the convergence by coming up with a better sequence of cycle toggles?

M-Bond:An Illustrative Example



Consider a simple "m-bond" graph comprising of 2 vertex, 1 tree edge, $m-1$ parallel edges and all edges have weight 1. Let the $d = [-1, 1]$ be the demand vector. It can be seen that the flow, f_0, of 1 on the tree edge and 0 on the non-tree edges is a feasible flow.

M-Bond:An Illustrative Example



KOSZ View

Since every edge has a stretch of 1, the tree condition number is $\tau = m * 1 + m - 4 + 2 = 2m - 2$, thus the convergence rate is $1 - \frac{1}{2m-2}$

M-Bond:An Illustrative Example

Spectral Graph Theory View

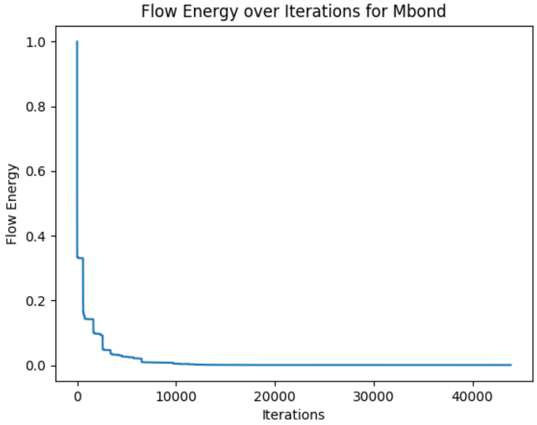
Let A be the random walk matrices for cycle toggling such that $f_{k+1} = A_k f_k$. Let $\bar{A}$ be the expectation of the As. The eigenvalues of $\bar{A}$ are:

1. $\lambda = 1$, with multiplicity 1
2. $\lambda = 1 - \frac{1}{2m-2}$, with multiplicity m-2
3. $\lambda = \frac{m-2}{2m-2}$, with multiplicity 1

This analysis agrees with the KOSZ view that the expected convergence rate should be $1 - \frac{1}{2m-2}$.

The smallest eigenvalue of $\lambda = \frac{m-2}{2m-2}$ suggests that the convergence rate can be much faster than $1 - \frac{1}{2m-2}$ and this agrees with experimental results.

M-Bond:An Illustrative Example



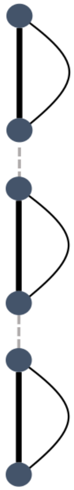
Idea List

1. Solver chain: A sequence of subgraphs $G_0 \subset G_1 \subset \cdots \subset G$ that has the same vertex set but are subsets of the edge set. An idea from the potential solver and the inituition is that it improves sampling at the start.
   Solver chain subsampling methods: [Uniform, Stretch, Reverse Stretch].
2. Change cycle sampling metric. [Uniform, Stretch raised to pth power].
3. Sampling without replacement. Commonly known as "Shuffle". Mixing between sampling with and without replacement.
4. Greedy. Get the edge with the highest toggle value. Assumes that is given by some oracle or can be estimated.
5. Toggle multiple cycles?

Experiment Setup

1. Implemented [KOSZ13] `SimpleSolver` without data structures as goal was to observe the number of cycle toggle iterations and not true timings.
2. Implemented in Python 3.5 which is not the fastest but has reasonably good libraries.
3. Tested on graphs which the Low Stretch Tree is known.
4. Run till error within $1e - 8$ of true solution.

MChain

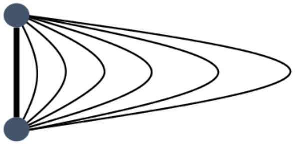


"Series Graph"

Number of cycle toggle iterations

m=	**KOSZ**	**SChain+mix**	**SChain+shuffle**
128	420	226	115
512	1329	778	327
2048	9449	3086	1528
4096	13586	6500	3222
8192	33838	11516	5529

Mbond

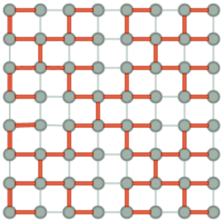


"Parallel Graph"

Number of cycle toggle iterations

m=	**KOSZ**	**SChain+mix**
64	1555	1240
256	5525	4719
1024	25809	19865
2048	53693	37688
4096	110486	76018
8192	192,948	115,050

2D-Grid



"Recursive C Low Stretch Tree of a 2D Grid"

Number of cycle toggle iterations

(n,m)=	**KOSZ**	**SChain+mix**
64,112	7155	7331
256,480	80145	73076
1024,1984	706,072	203,438
4096,8064	5,939,078	1,251,767

2D-Grid

Mix of other methods on 2D grid

(n,m)=	256, 480
KOSZ	80145
SChain+Mix	73076
SChain+Mix+Pow	36171
Greedy	30031

Experiements Remarks

1. **Solver Chain + Mixing** reduces the iteration count compared to **KOSZ**, showing that having a better toggle sequence yields better convergence.
2. However, it is still slow at the "end game" when all edges have some flow.
3. **Greedy Toggle** shows that there is a lower bound on cycle toggling. Even knowing the best move still requires greater than $O(m)$ iterations.

Randomized Kaczmarz Algorithm

The Kaczmarz's algorithm is an iterative algorithm for solving linear equation systems $Ax = b$ [Kac37]. In each iteration k, the ith row of A is pick and the following update is made.

$$x^{k+1} = x^k - \frac{(A_i^T x^k - b)}{|A_i^T|_2^2} A_i^T$$

The Randomized Kaczmarz's algorithm is where the rows of A are picked at random. [SV07] showed that pick the rows based on the row norm give better convergence than picking at uniform.

[KOSZ13] in section 9.3 commented on a possible link of their algorithm to the Randomized Kaczmarz Algorithm.

Stochastic Dual Ascent

[GR15] developed and analyzed a versatile randomized iterative method for solving a consistent system of linear equations which they term as **Stochastic Dual Ascent**. In [GR15], they show reductions for 6 algorithms to their framework:

1. Sketching viewpoint: Sketch and Project
2. Optimization viewpoint: Constrain and Approximate
3. Geometric viewpoint: Random Intersect
4. Algebraic viewpoint 1: Random Linear Solve
5. Algebraic viewpoint 2: Random Update
6. Analytic viewpoint: Random Fixed Point

Primal-Dual Problems

Given $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m,\ B \in \mathbb{R}^{n\times n} and sym - pos - definite$

Primal	**Dual**
$\underset{x}{\text{minimize}} \quad \frac{1}{2}\lvert x\rvert_B^2$	$\underset{y}{\text{maximize}} \quad b^T y - \frac{1}{2}\lvert A^T y\rvert_{B^{-1}}^2$
subject to $\quad Ax = b$	subject to $\quad y \in \mathbb{R}^m$
$x \in \mathbb{R}^n$	

Assumes system is consistent and strong duality holds. Optimal solutions x^*, y^* exists.

Why Stochastic?

Let S be a $m \times k$ random matrix drawn independently from some distribution $\mathcal{D}$.
$\mathcal{D}$ can be view as a family of algorithms that solves the given optimization problem.
To goal of S is to make $(S^T A B^{-1} A^T S)^+$ easy to solve.

Definition

Primal Iterates
$x^{k+1} = x^k - B^{-1} A^T S((S^T A B^{-1} A^T S)^+) S^T (Ax^k - b)$

Definition

Dual Iterates
$y^{k+1} = y^k + S((S^T A B^{-1} A^T S)^+) S^T (b - A B^{-1} A^T y^k)$

Convergence Analysis

[GR15] Theorem 1.1 (Convergence of primal iterates and residuals).

1. Weak Assumption on $\mathcal{D}$. Assume H is well defined and non-singular

$$H := E_{S\sim\mathcal{D}}\left[S(S^TAB^{-1}A^TS)^{+}S^T\right]$$

2. Convergence Factor

$$\rho := 1 - \lambda_{\min}^{+}(B^{-\frac{1}{2}}A^THAB^{-\frac{1}{2}})$$

$$\text{Primal: } E[|x^k - x^*|_B^2] \leq \rho^k|x^0 - x^*|_B^2$$

$$\text{Residual: } E[|Ax^k - b|_B] \leq \rho^{k/2}|A|_B|x^0 - x^*|_B$$

3. Convergence Rate Bounds

$$1 - \frac{E[Rank(S^TA)]}{Rank(A)} \leq \rho < 1$$

Graph Preliminaries II

Definition (Cycle Matrix)

Given a spanning tree, since there is a unique tree path for every pair of vertices, every non-tree edge forms a unique cycle with the tree.
Given a non-tree e, call this unique cycle, C_e, the fundamental cycle of e. It is crucial to note that the edges on the cycle is oriented and the orientation is given by $\mathbf{B}$
Let the cycle matrix $\mathbf{K}$ be the $|E| \times |E_n|$ matrix whose rows are indexed by edges and columns are indexed by non-tree edges where

$$\mathbf{K}_{a,b} = \begin{cases} 1 & \text{if } a \in C_b \text{ and } a, b \text{ have the same orientation in } C_b, \\ -1 & \text{if } a \in C_b \text{ and } a, b \text{ have opposite orientation in } C_b, \\ 0 & \text{if } a \notin C_b. \end{cases} \tag{3}$$

Graph Preliminaries II

Checked that

$$\mathbf{BK} = 0$$

The cycle matrix is in the nullspace of the incidence matrix. Similar to the incidence matrix, by ordering the first $n-1$ rows of $\mathbf{K}$ to be the edges of the tree, we can represent $\mathbf{K}$ as 2 block matrices, $\mathbf{K}_t$, a $(n-1) \times (m-n+1)$ matrix for the tree edges, and $\mathbf{K}_n$, a $(m-n+1) \times (m-n+1)$ matrix for the non-tree edges

$$\mathbf{K} = \left[\begin{array}{c} \mathbf{K}_t \\ \hline \mathbf{K}_n \end{array}\right]$$

Note that $\mathbf{K}_n = I$ since each fundamental cycle would contain only 1 non-tree edge.

Graph Preliminaries II

We can express **K** in terms of **B**

$$\begin{aligned}\mathbf{B}\mathbf{K} &= 0\\ [\mathbf{B}_t|\mathbf{B}_n]\left[\frac{\mathbf{K}_t}{\mathbf{K}_n}\right] &= 0\\ \mathbf{B}_t\mathbf{K}_t + \mathbf{B}_n I &= 0\\ \mathbf{K}_t &= -\mathbf{B}_t^{+}\mathbf{B}_n\end{aligned}$$

where $\mathbf{B}_t^{+}$ is the pseudo inverse of $\mathbf{B}_t$

Flow Preliminaries

Definition (Potential, Potential Flow)

Let the vertex potential (potential), v, be a $\mathbb{R}^{|V|}$ vector. Similar to a electrical circuit, a potential flow is an induced flow on the edges where the flow amount is proportional to the conductance and the difference between the potentials of the start and end vertices of the edge.

$$f_{potential} = \mathbf{R}^{-1}\mathbf{B}^T v$$

Definition (Generator, Circulation Flow)

Let a potential difference generator (generator),u, be a $\mathbb{R}^{|E_n|}$ vector. This is setting of flows on the non-tree edges. A circulation flow is where the in flow and out flow on every vertex is 0.

$$f_{circulation} = \mathbf{K}u$$

Preliminaries II Summary

Incidence and Cycle Matrices

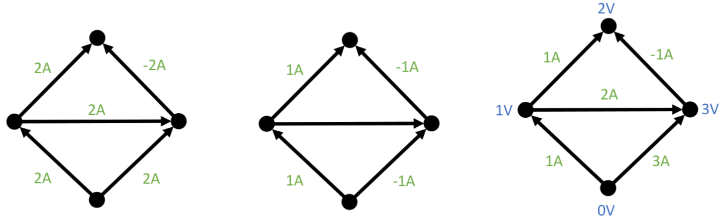
$$\mathbf{B} = [\mathbf{B}_t | \mathbf{B}_n],\ \mathbf{K} = \left[\frac{\mathbf{K}_t}{\mathbf{K}_n}\right],\ \mathbf{BK} = 0 \rightarrow \mathbf{K}_t = -\mathbf{B}_t^+ \mathbf{B}_n$$

Potential Flow, Pure Circulations

$$f_{potential} = \mathbf{R}^{-1}\mathbf{B}^T v,\ f_{circulation} = \mathbf{K}u$$

Flow Decomposition

$$f = \mathbf{R}^{-1}\mathbf{B}^T v + \mathbf{K}u = \mathbf{R}^{-1}(\mathbf{B}^T v + \mathbf{RK}u)$$



Flow = Pure Circulation + Potential Flow

KOSZ13 SimpleSolver in GR15

SDA Primal	**KOSZ13**
$\underset{x}{\text{minimize}} \quad \frac{1}{2}\lvert x\rvert_B^2$	$\underset{f}{\text{minimize}} \quad \frac{1}{2} f^T R f$
subject to $\quad Ax = b$	subject to $\quad \mathbf{K}^T \mathbf{R} f = 0$
$x \in \mathbb{R}^n$	$f \in \mathbb{R}^m$
	given $\quad \mathbf{B} f_0 = d$

Symbol Mapping of SDA to KOSZ

$$B \to \mathbf{R},\ A \to \mathbf{K}^T\mathbf{R},\ x \to f,\ b \to 0$$

Since KOSZ13 starts of with a demand statisfying flow, updating by some row of $\mathbf{K}^T\mathbf{R}$ will keep the flow at each iteration within the space of demand-statisfying flow. Recall $\mathbf{BK} = 0$.

KOSZ13 SimpleSolver in GR15

KOSZ13 Primal Iterates

$$\begin{aligned}
x^{k+1} &= x^k - B^{-1}A^T S((S^T A B^{-1} A^T S)^+) S^T (Ax^k - b) \\
\to f^{k+1} &= f^k - \mathbf{R}^{-1}\mathbf{RK}S((S^T\mathbf{K}^T\mathbf{RR}^{-1}\mathbf{RK}S)^+)S^T(\mathbf{K}^T\mathbf{R}f^k) \\
&= f^k - (\mathbf{K}S)((S^T\mathbf{K}^T\mathbf{RK}S)^+)(S^T\mathbf{K}^T\mathbf{R}f^k) \\
\to &= f^k - \alpha\mathbf{K}_i
\end{aligned}$$

Using `Randomized Coordinate Descent`, at each round, pick some $S_i = e_i$ where e_i is ith standard basis vector in $\mathbb{R}^{m'}$ with probability $p_i = \frac{1}{\tau}\log\left(\frac{R_e}{r_e}\right)$.

KOSZ13 SimpleSolver in GR15

[GR15] gives the convergence to be

$$\rho := 1 - \lambda_{\min}^{+}(B^{-\frac{1}{2}} A^T H A B^{-\frac{1}{2}})$$

Goal: Calculate H, ρ

$$\begin{aligned}
H &= E_{S\sim\mathcal{D}}\left[S(S^T A B^{-1} A^T S)^{+} S^T\right] \\
\rightarrow &= E_{S\sim D}\left[S(S^T \mathbf{K}^T \mathbf{R}\mathbf{R}^{-1}\mathbf{R}^T \mathbf{K} S)^{+} S^T\right] \\
&= E_{S\sim D}\left[S(S^T \mathbf{K}^T \mathbf{R}^T \mathbf{K} S)^{+} S^T\right] \\
&= \sum_{i\in\text{non-tree}} p(S_i)\left(S_i (R_{c_i})^{-1} S_i^T\right) \\
&= \sum_i \left(\frac{R_{c_i}}{\tau r_i}\right)\left(S_i (R_{c_i})^{-1} S_i^T\right) \\
&= \frac{1}{\tau} R_n^{-1}
\end{aligned}$$

KOSZ13 SimpleSolver in GR15

Goal to find $\rho = 1 - \lambda^{+}_{\min}(B^{-\frac{1}{2}} A^T H A B^{-\frac{1}{2}})$

Bound $\lambda^{+}_{\min}(B^{-\frac{1}{2}} A^T H A B^{-\frac{1}{2}})$

$$(B^{-\frac{1}{2}} A^T H A B^{-\frac{1}{2}}) = (\mathbf{R}^{-\frac{1}{2}} \mathbf{R}^T \mathbf{K} (\frac{1}{\tau} R_n^{-1}) \mathbf{K}^T \mathbf{R} \mathbf{R}^{-\frac{1}{2}})$$
$$= \frac{1}{\tau} \mathbf{R}^{\frac{1}{2}} \mathbf{K} \mathbf{R}_n^{-1} \mathbf{K}^T \mathbf{R}^{\frac{1}{2}}$$

Find the number of 0 eigenvalues, first show that
$Rank(\mathbf{R}^{\frac{1}{2}} \mathbf{K} \mathbf{R}_n^{-1} \mathbf{K}^T \mathbf{R}^{\frac{1}{2}}) = m'$.
But we know from $Rank(A) = Rank(A^T A)$, sufficies to show
$Rank(\mathbf{R}_n^{-\frac{1}{2}} \mathbf{K}^T \mathbf{R}^{\frac{1}{2}}) = m'$

$$\mathbf{R}_n^{-\frac{1}{2}} \mathbf{K}^T \mathbf{R}^{\frac{1}{2}} = \mathbf{R}_n^{-\frac{1}{2}} \left[\frac{\mathbf{K}_t}{\mathbf{I}} \right] \mathbf{R}^{\frac{1}{2}}$$
$$= \left[\frac{\mathbf{R}_n^{-\frac{1}{2}} \mathbf{K}_t \mathbf{R}_t^{\frac{1}{2}}}{\mathbf{I}} \right]$$

KOSZ13 SimpleSolver in GR15

$\mathbf{R}^{\frac{1}{2}}\mathbf{K}\mathbf{R}_n^{-1}\mathbf{K}^T\mathbf{R}^{\frac{1}{2}}$ is a $m \times m$ matrix, meaning it has $m - m' = n - 1$, 0's as eigenvalues.

Observe that

$$\mathbf{R}^{\frac{1}{2}}\mathbf{K}\mathbf{R}_n^{-1}\mathbf{K}^T\mathbf{R}^{\frac{1}{2}} = \left[\begin{array}{c|c} \mathbf{R}_t^{\frac{1}{2}}\mathbf{K}_t\mathbf{R}_n^{-1}\mathbf{K}_t^T\mathbf{R}_t^{\frac{1}{2}} & \mathbf{R}_t^{\frac{1}{2}}\mathbf{K}_t\mathbf{R}_n^{-\frac{1}{2}} \\ \hline \mathbf{R}_n^{-\frac{1}{2}}\mathbf{K}_t^T\mathbf{R}_t^{\frac{1}{2}} & \mathbf{I} \end{array}\right]$$

Remove the first $n - 1$ row & columns and apply **Cauchy Interlace Theorem** $n - 1$ times.

Since $\lambda^+_{\min}(\mathbf{I}) = 1$, we get $\lambda^+_{\min}(\mathbf{R}^{\frac{1}{2}}\mathbf{K}\mathbf{R}_n^{-1}\mathbf{K}^T\mathbf{R}^{\frac{1}{2}}) \geq 1$

KOSZ13 SimpleSolver in GR15

Upper and lower bounds for ρ

$$\rho = 1 - \frac{1}{\tau}\left(\lambda^{+}_{\min}(\mathbf{R}^{\frac{1}{2}}\mathbf{K}\mathbf{R}_n^{-1}\mathbf{K}^T\mathbf{R}^{\frac{1}{2}})\right)$$

$$\rho \leq 1 - \frac{1}{\tau}$$

$$1 - \frac{E[Rank(S^T A)]}{Rank(A)} \leq \rho$$

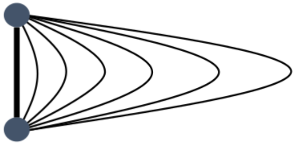
$$1 - \frac{E[Rank(S^T \mathbf{K}^T \mathbf{R})]}{Rank(\mathbf{K}^T \mathbf{R})} \leq \rho$$

$$1 - \frac{1}{m'} \leq \rho \leq 1 - \frac{1}{\tau}$$

$$1 - \frac{1}{m - n + 1} \leq \rho \leq 1 - \frac{1}{\tau}$$

Matches the upper bound in KOSZ13.
But we get a lower bound too!

M-Bond: A Review



Recap KOSZ View

Since every edge has a stretch of 1, the tree condition number is $\tau = m * 1 + m - 4 + 2 = 2m - 2$, thus the convergence rate is $1 - \frac{1}{2m-2}$

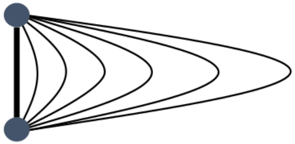
New KOSZ View

The convergence factor is $1 - \frac{1}{m-1} \leq \rho \leq 1 - \frac{1}{2(m-1)}$. The total iteration count is $O(m \log m)$ for a constant approximation.

New Solver Chain View

The convergence factor at the k stage with 2^k non-tree edges is $1 - \frac{1}{2^k} \leq \rho \leq 1 - \frac{1}{2^k+1}$. The total iteration count is $O(m)$ for a constant approximation.

Mbond: A Review



"Parallel Graph"

Number of cycle toggle iterations for 1+1/2 approx

m=	**KOSZ**	**SChain**
64	190	251
256	1669	1014
1024	4888	3963
2048	11882	8698
4096	36365	16999
8192	71132	32488
8192	203027	67672

KOSZ13 SimpleSolver in GR15 Remarks

1. The lower bound of $1 - \frac{1}{m-n+1} \leq \rho$ shows that single cycle updates cannot reach $O(m)$ iterations even though there might be good eigenvalues.
2. Solver Chain methods helped (initially) as the number of non-tree edges are reduced making $1 - \frac{1}{m'}$ smaller.
3. Future improvements would need to "toggle more than 1" cycle per iteration.

Min-Energy, Demand Statisfying, Potential Flow in GR15

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & f^T \mathbf{R} f \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{K}^T\mathbf{R} \\ \mathbf{B} \end{bmatrix} f = \begin{bmatrix} 0 \\ d \end{bmatrix} \end{aligned}$$

Work In Progress

Applied [GR15] directly to the general verion of the electrical flow problem.
Show to be faster in practice as multiple cycles can be toggled in each iteration. However, currently missing fast way to "fix" the flow such that it always meet the demand as per [KOSZ13] requirements.

Conclusion

1. Show the connection of [KOSZ13] with stochastic ascent methods.
2. Gave an analysis of the upper and lower bound of the convergence of [KOSZ13].
3. Explained cycle toggling's practical performance.
4. Show experimentally on the behavior of different sampling schemes.

Thank You

Q&A

Reference I

Ittai Abraham and Ofer Neiman.
Using petal decompositions to build a low stretch spanning tree.
In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 395–406, New York, NY, USA, 2012. ACM.

Erik G. Boman, Kevin Deweese, and John R. Gilbert.
An empirical comparison of graph Laplacian solvers.
In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pages 174–188, 2016.

Reference II

Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng.
Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs.
In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282, 2011.
Available at http://arxiv.org/abs/1010.2921.

Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup Rao, and Shen Chen Xu.
Solving SDD linear systems in nearly $m\log^{1/2}n$ time.
In *STOC*, pages 343–352, 2014.

Reference III

Kevin Deweese, John R. Gilbert, Gary L. Miller, Richard Peng, Hao Ran Xu, and Shen Chen Xu.
An empirical study of cycle toggling based laplacian solvers.
CoRR, abs/1609.02957, 2016.

Robert M Gower and Peter Richtarik.
Stochastic Dual Ascent for Solving Linear Systems.
ArXiv e-prints, December 2015.

S. Kaczmarz.
Angenäherte Auflösung von Systemen linearer Gleichungen.
Bulletin International de l'Académie Polonaise des Sciences et des Lettres, 35:355–357, 1937.

Reference IV

Ioannis Koutis, Gary L. Miller, and Richard Peng.
A nearly-m log n time solver for SDD linear systems.
In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 590–598, Washington, DC, USA, 2011. IEEE Computer Society.
Available at http://arxiv.org/abs/1102.4842.

Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu.
A simple, combinatorial algorithm for solving SDD systems in nearly-linear time.
In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 911–920, New York, NY, USA, 2013. ACM.
Available at http://arxiv.org/abs/1301.6628.

Reference V

Yin Tat Lee and Aaron Sidford.
Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems.
In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 147–156, Washington, DC, USA, 2013. IEEE Computer Society.

Daniel A. Spielman and Shang-Hua Teng.
Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems.
CoRR, abs/cs/0607105, 2006.

T. Strohmer and R. Vershynin.
A randomized Kaczmarz algorithm with exponential convergence.
ArXiv Mathematics e-prints, February 2007.