The Revolution in Graph Theoretic Optimization Problems

Gary L Miller

Carnegie Mellon University

CSC 2016

October 12, 2016

OUTLINE

- Linear system solvers
- Graph Sparsifiers.
- Regression and Image Denoising.
- Simple formulation and connection with solving linear systems.
- Overview of SDD solvers.
- A better L_1 formulation of denoising.
- Maximum flow using solvers

- er

SPECTRAL GRAPH THEORY LAPLACIAN PARADIGM

Use graph algorithms to solve linear algebra problems.

Use linear algebra to solve graph problems.

Use both to solve optimization problems

ASYMPTOTIC ANALYSIS

- The goal is to find algorithms with provable worst case run times.
- Randomized methods are assumed
- A scalable algorithm is one whose run scales linearly with problems size, ignoring log terms.
- When possible we hope to find a scalable algorithm.

SOLVING AX=B, A SPARSE

1) Direct Methods (Gaussian elimination)+
   1) + Ones gets the exact answer!
   2) + Partial elimination reduces vertex count
   3) - Fill and work may be large!
   4) - Good pivot orders may be expense to find!
   5) - Elimination will not give scalable algorithms on it own.

SOLVING AX=B, A SPARSE

1) Iterative Methods
   1) Each step performs a
      1) Sparse Matrix Vector product
      2) Constant number of dot products and additions
   2) + Each step is linear time and parallel
   - \- Finds at best closets point in the Krylov spaces
   - $< b, Ab, \dots, A^{n-1} b >$

2) Two work arounds
   1) Find a good preconditioner
      1) i.e. Change our Krylov space.
   2) Include matrix-matrix produces and subsample to abtain a approximate product.

GRAPH SPARSIFIERS

Sparse Equivalents of Dense Graphs that preserve some property

- Spanners: distance, diameter.
- [Benczur-Karger '96] Cut sparsifier: weight of all cuts.
- Spectral sparsifiers: eigenstructure

GENERAL GRAPH SAMPLING MECHANISM

- For each edge, flip a coin with probability of ‘keep’ being P(e).
- If coin says ‘keep’, keep the edge, but scale it up by factor of 1/P(e).

Expected value of an edge: same as before

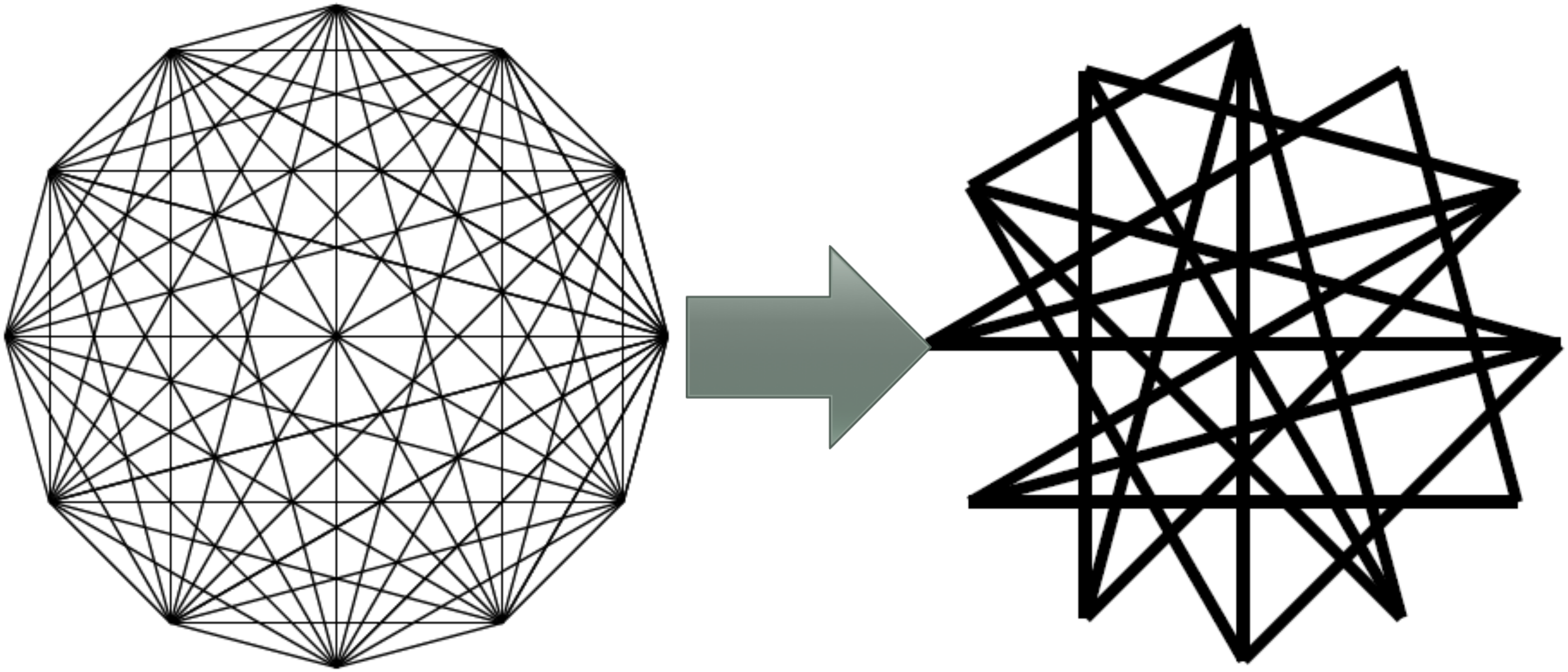
Sampling gives the graph in expectation; we only need to bound concentration.

TWO APPROACHES: GRAPH SAMPLING

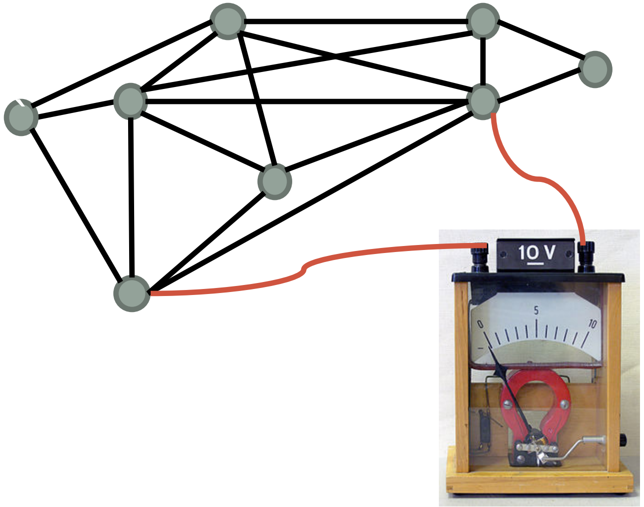
- Sample in parallel: Pick k samples and return the average graph.
- Pick a few samples and based on the sample update the probabilities. Repeat!
-

EXAMPLE: COMPLETE GRAPH

O(n logn) sampling edges uniform suffice!



EFFECTIVE RESISTANCE



- View the graph as a circuit: each edge is a resistor with conductance W(e).
- Effective resistance between u and v, R(u,v) is the resistance of the circuit when passing 1 unit of current from u to v.

MATRIX CHERNOFF BOUNDS

- Def: $A \preccurlyeq B$ if $\forall x \quad x^T A x \leq x^T B x$

- THM: Let $X_1, \dots X_k$ be ind random matries
- $0 \preccurlyeq X_i \preccurlyeq R\,\mathbf{I}$, $\sum E(X_i) \preccurlyeq u\,\mathbf{I}$ then
- Prob[$\lambda_{max}(\sum X_i) > (1+ \delta)\sum E(X_i)$] $\leq$ exp small

This Thm and its extension is central to what follows!

SPECTRAL SPARSIFICATION BY EFFECTIVE RESISTANCE

What probability P(e) should we sample an edge?

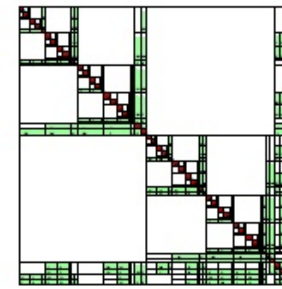
Answer: [Spielman-Srivastava `08]:

- Set P(e) = R(u,v) effective resistance from u to v.
- For each sampled edge set weight to 1/P(e)
- Sample O(n log n) times and return average.

spectral sparsifier with O(nlogn) edges for any graph

Foster: $\sum_e R(e) = n-1$

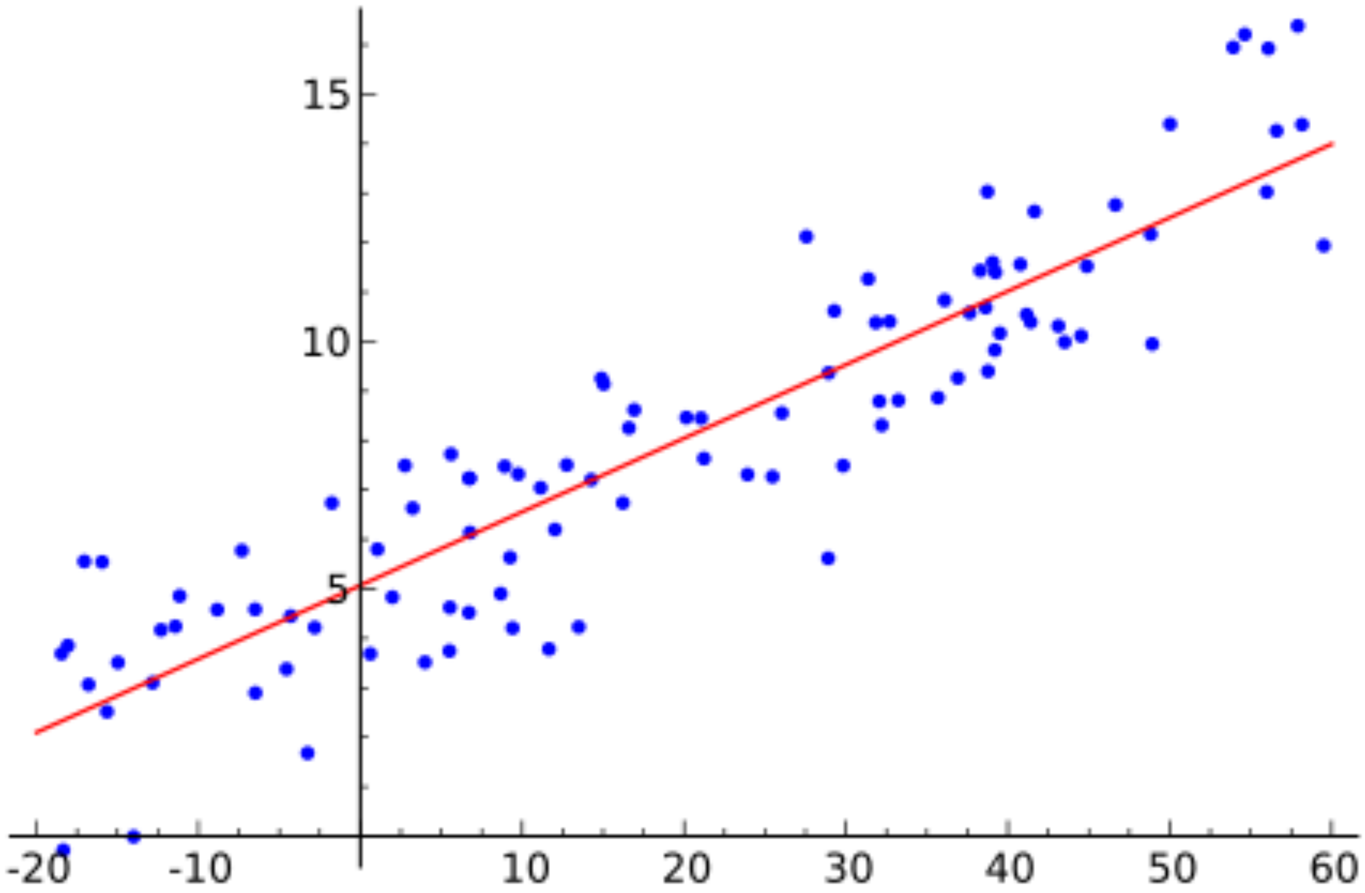
SPARSE CHOLESKY FOR GRAPHS



Kyng-Sachdeva `16:
1. Run Gaussian Elimination in random order
2. After each pivot sample the new fill by effective resistance.

Running time bound: $O(m\log^3 n)$, OPEN: improve this

REGRESSION



OVER CONSTRAINED SYSTEMS

Over Constrained System: $A x = b$.

Solve system $A^T A x = A^T b$

- Matrix $A^T A$ is Symmetric Positive Semi-Definite (SPSD).
- Open Question:

  Find sub-quadratic time solvers for SPD systems?

We will need problems with an underlying graph.

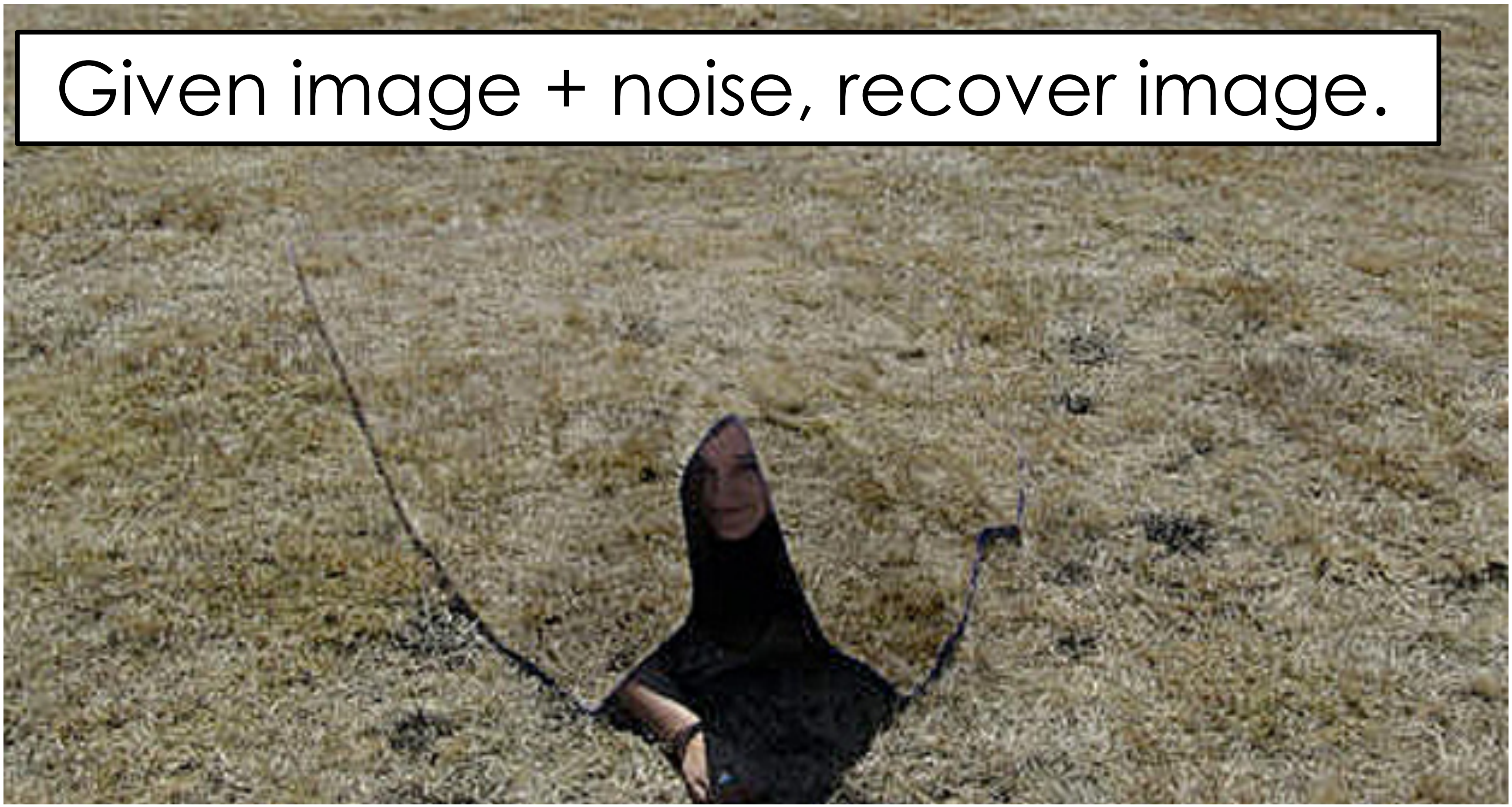
APPROXIMATION ALGORITHMS

- Whole conferences NP-Approximation
- Same ideas and goals can be applied problems in Polytime.
- Our goal is find good approximation but much faster than known exact solutions.
- Maybe even faster exact solutions!

CLASSIC REGRESSION PROBLEM

- Image Denoising
- Critical step in image segmentation and detection
- Good denoising makes the segmentation almost obvious.

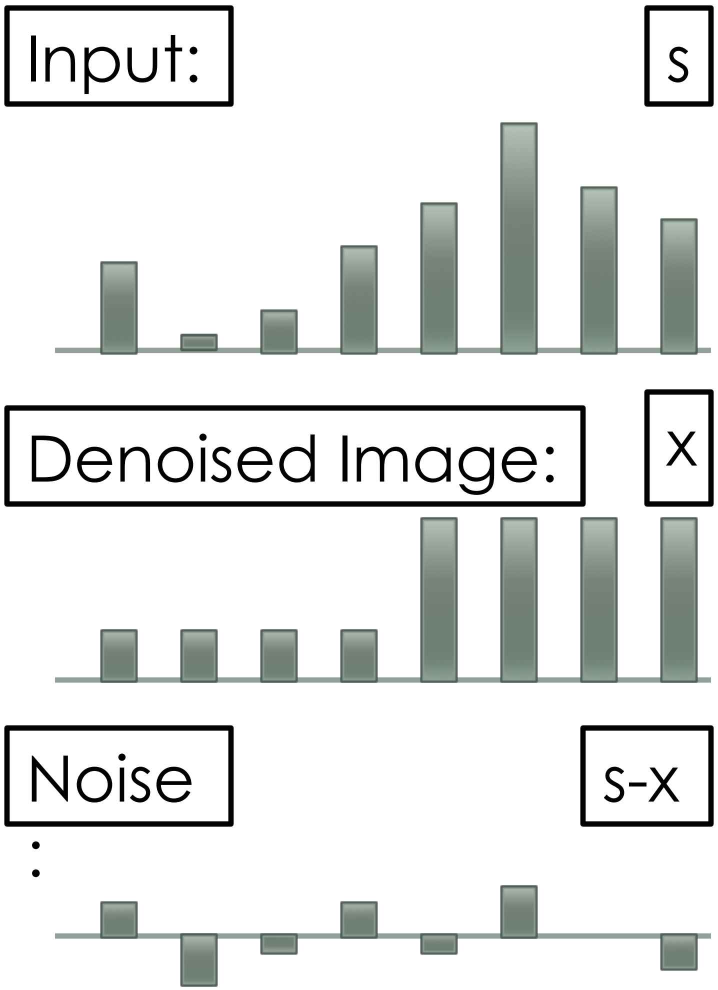
CAMOUFLAGE DETECTION



CAMOUFLAGE DETECTION

Hui-Han Chin

IMAGE DENOISING: THE MODEL



- Assume there exist a 'original' noiseless image.
- Noise generated from some distribution.
- Input: original + noise.
- Goal: approx the original image.

CONDITIONS ON X

- Noise is small: denoised image should still be close to image + noise.
- Real images are 'smooth' except at boundaries.

Function of (x-s)

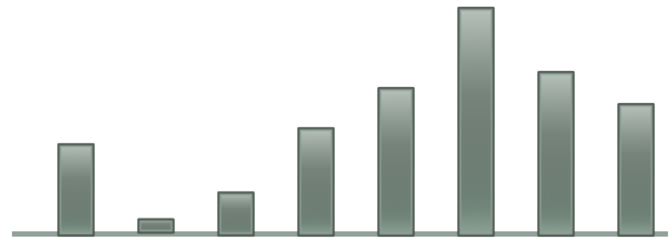
Fidelity(x, s)

Function computed on differences of neighboring pixels of x

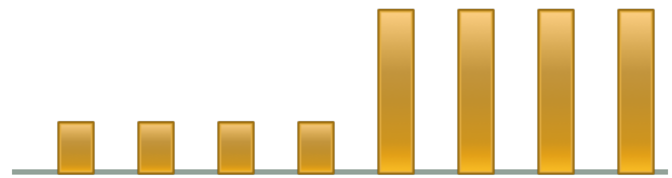
Smoothness(x)

ENERGY FUNCTION

Noisy image: s



Candidate solution: x



Fidelity(x, s) + Smoothness(x)

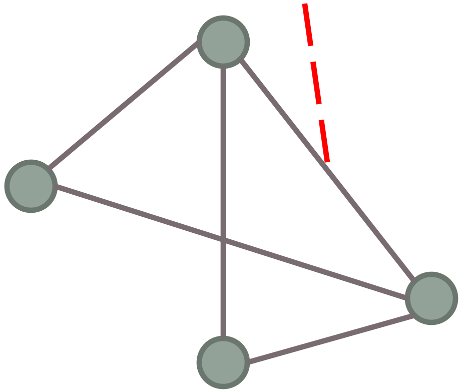
$(x_i - s_i)^2$ summed over all pixels i.

$(x_i - x_j)^2$ summed over all neighbor pixels i, j.

This is a toy example

MATRICES ARISING FROM IMAGE PROBLEM HAVE NICE STRUCTURES

$$\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$



A is Symmetric Diagonally Dominant (SDD)

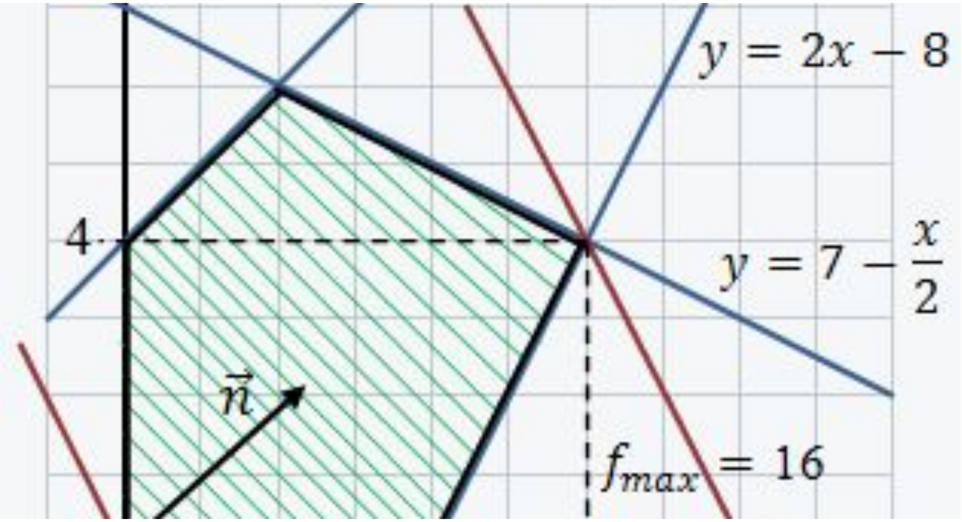
- Symmetric.
- Diagonal entry ≥ sum of absolute values of all off diagonals.

OPTIMIZATION PROBLEMS IN CS

Many algorithm problems in CS are optimization problems with underlying graph.

- Maximum flow in a graph.
- Shortest path in a graph.
- Maximum Matching.
- Scheduling
- Minimum cut.

LINEAR PROGRAMMING



Many optimization problems can be written as an LP

EG: Single source shortest path.

Is this useful?

FASTER OPTIMIZATION

- Undirected Maximum flow in a graph.
  - Peng $O(m\,\varepsilon^{-2})$ time
- Shortest path in a graph (negative weights)
  - Cohen-Madry $O(m^{10/7} \log (1/\varepsilon))$ time
- Maximum Matching.
  - Madry $O(m^{10/7} \log (1/\varepsilon))$ time
- General Linear Programs.
  - Lee-Sidford $O(\sqrt{\mathrm{Rank}(A)}\,(1/\varepsilon))$ iterations
- Total Variation image Denoising
  - Madry-M-Peng $O(m^{4/3}\,\varepsilon^{-2})$ time

THE BOUNDARY MAP B

Let G = (V,E) be n vertex m oriented edges graph.

- Def: B is a Vertex by Edge matrix

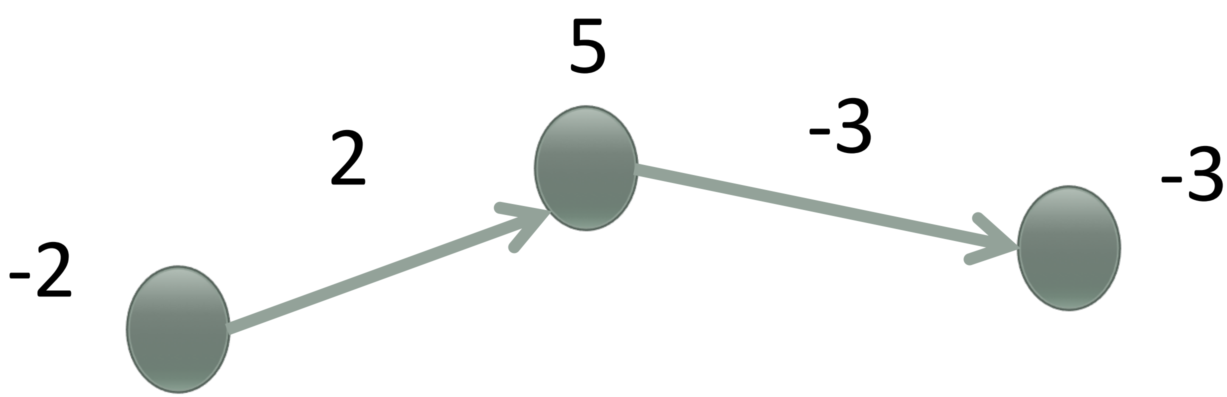
  where B_{ij} = +1 if v_i is head of e_j

  -1 if v_i is tail of e_j

  0 otherwise

- Note: If f is a flow then Bf is residual vertex flow.

BOUNDARY MATRIX



$$\begin{array}{c} \\ {\color{blue}v_1} \\ {\color{blue}v_2} \\ {\color{blue}v_3} \end{array} \begin{array}{c} \begin{array}{cc} {\color{blue}e_1} & {\color{blue}e_2} \end{array} \\ \begin{pmatrix} -1 & 0 \\ +1 & -1 \\ 0 & +1 \end{pmatrix} \end{array} \begin{pmatrix} 2 \\ -3 \end{pmatrix} = \begin{pmatrix} -2 \\ 5 \\ -3 \end{pmatrix}$$

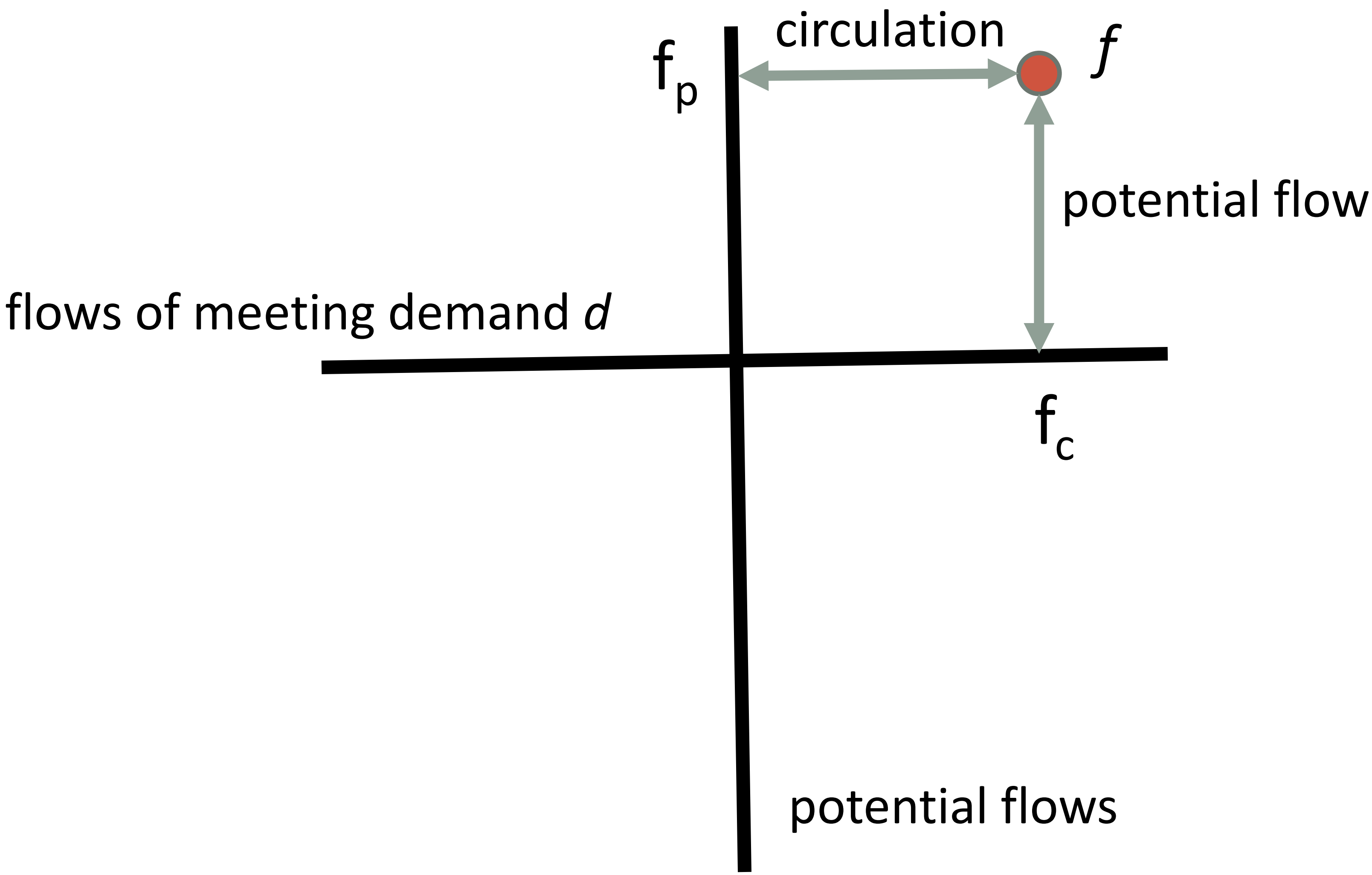
B^T AND POTENTIAL DROPS

- Let v be a n-vector of potentials
- $B^T v$ = vector of potential drops.
- $R^{-1}BTv$ = vector of edge flows.
  - R a diagonal matrix of resistive values
  - Ohms law: Rule to go from potentials to flows.
- Today we set resistors all to one.
- Thus $B^T v$ = vector of flows.

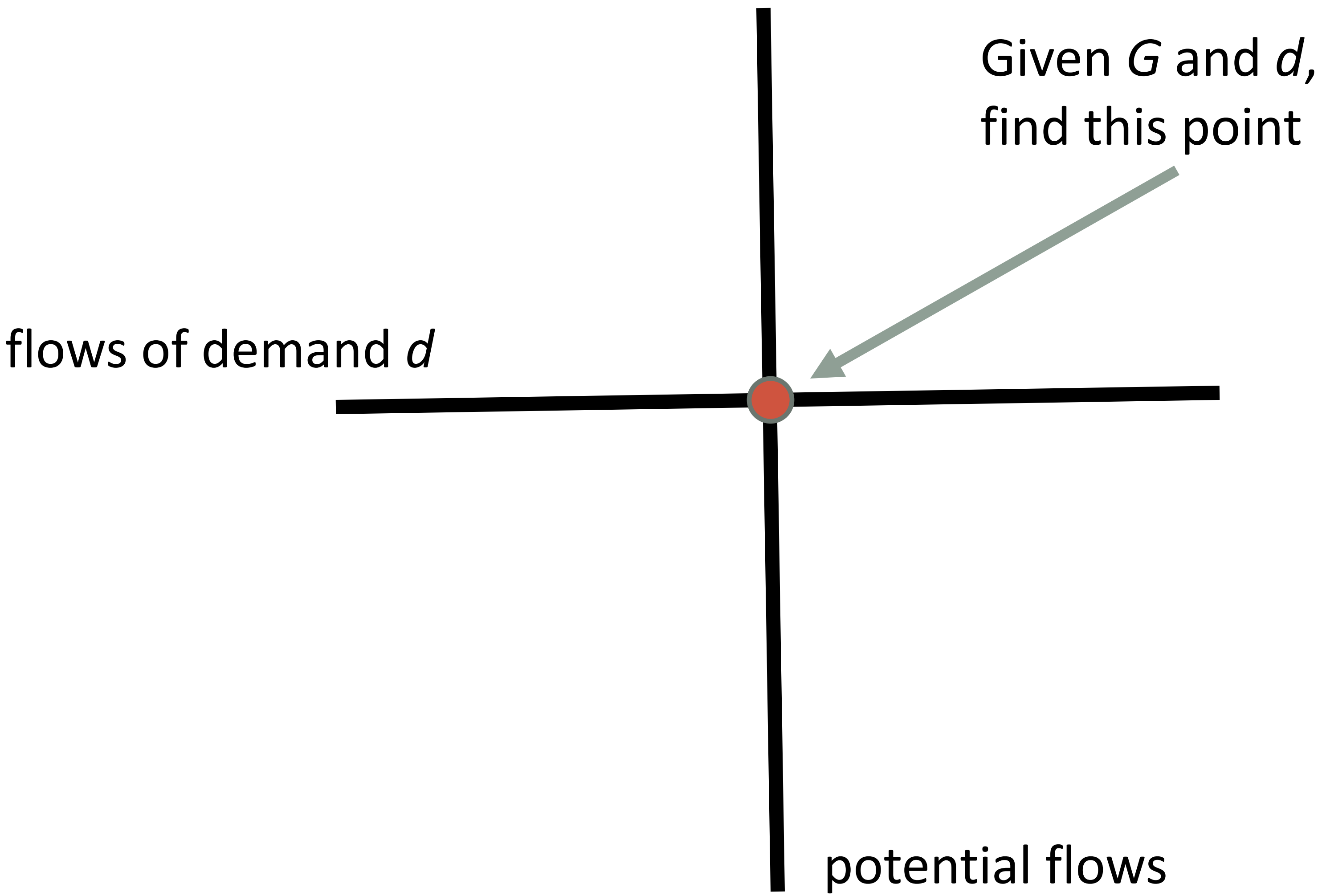
GRAPH LAPLACIAN SOLVERS

- Def: $L := BB^T$, Laplacian of G.
- Two dual approaches to approximately solving $Lv = b$
- 1) Find a potential that minimizes $Lv - b$
- 2) find a minimum energy flow f s.t. $Bf = b$

THE SPACE OF FLOWS



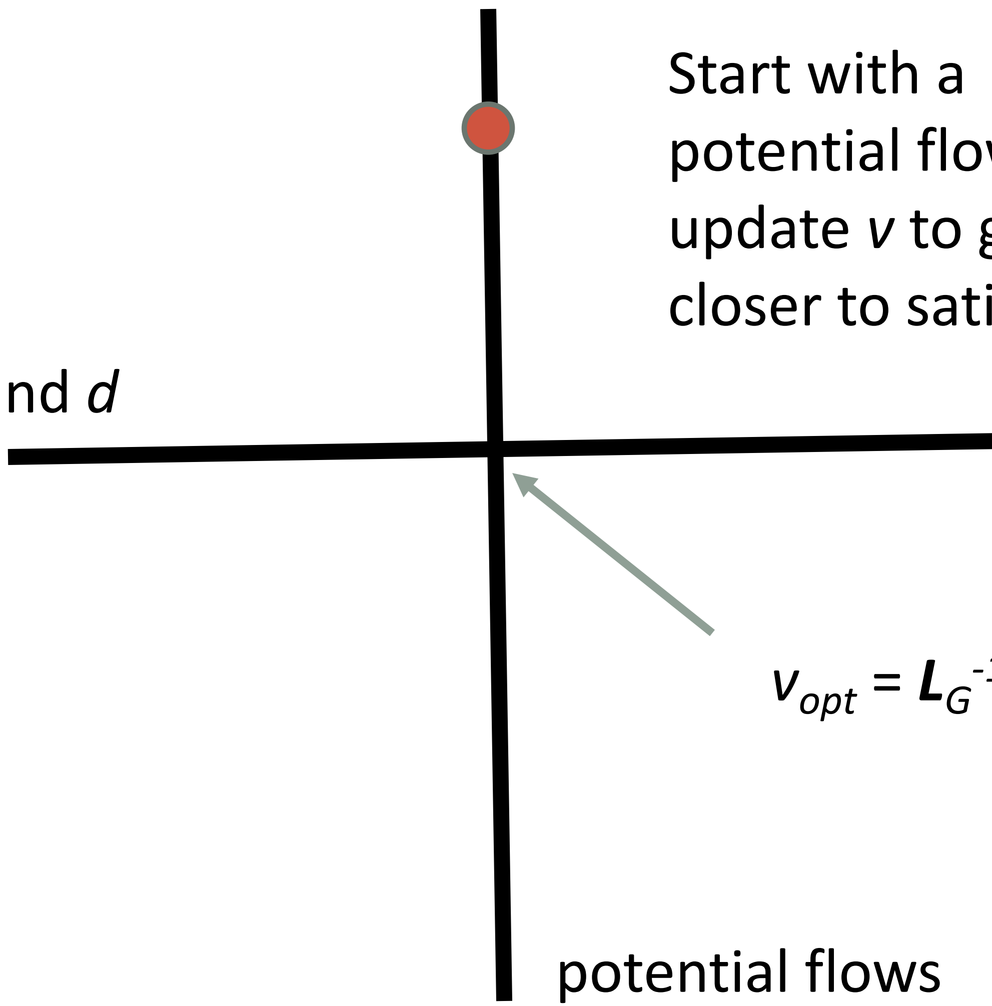
SOLVING LAPLACIANS



DUAL APPROACH: SOLVING A LINEAR SYSTEM

Start with a potential flow $\boldsymbol{B}v$, update v to get closer to satisfying d

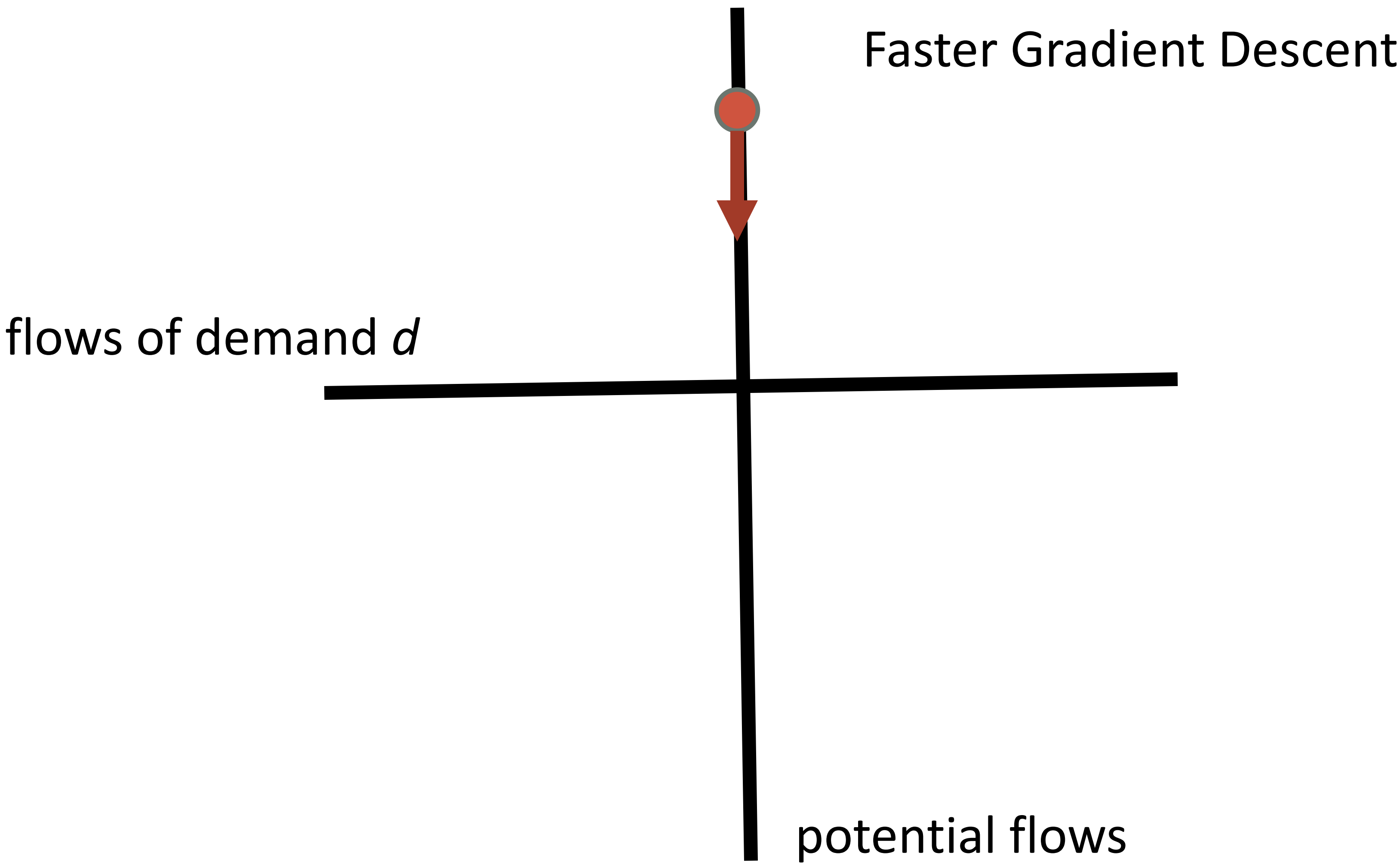
flows of demand d



$v_{opt} = \boldsymbol{L}_G^{-1} d$

potential flows

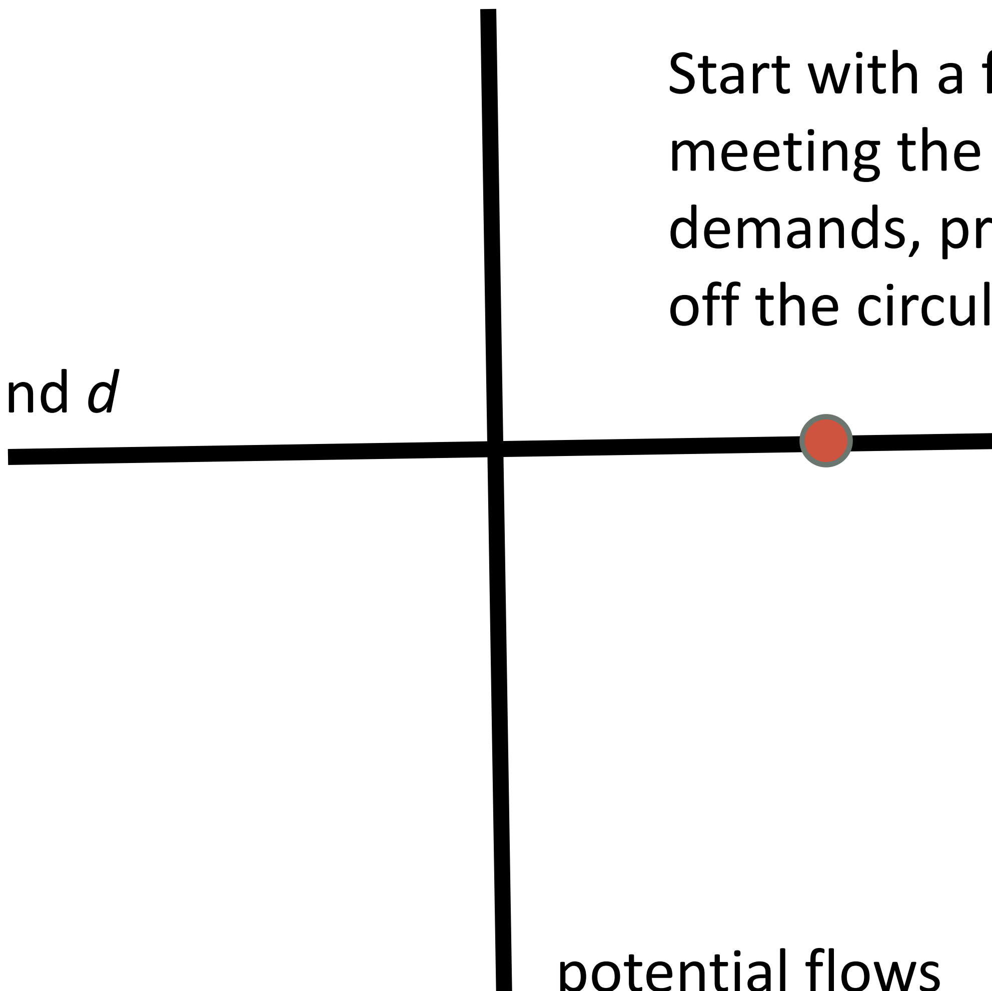
DUAL APPROACH: SINGLE STEP (ST'04,KMP '10, '11)



PRIMAL APPROACH: SOLVING A FLOW PROBLEM

Start with a flow meeting the demands, project off the circulation

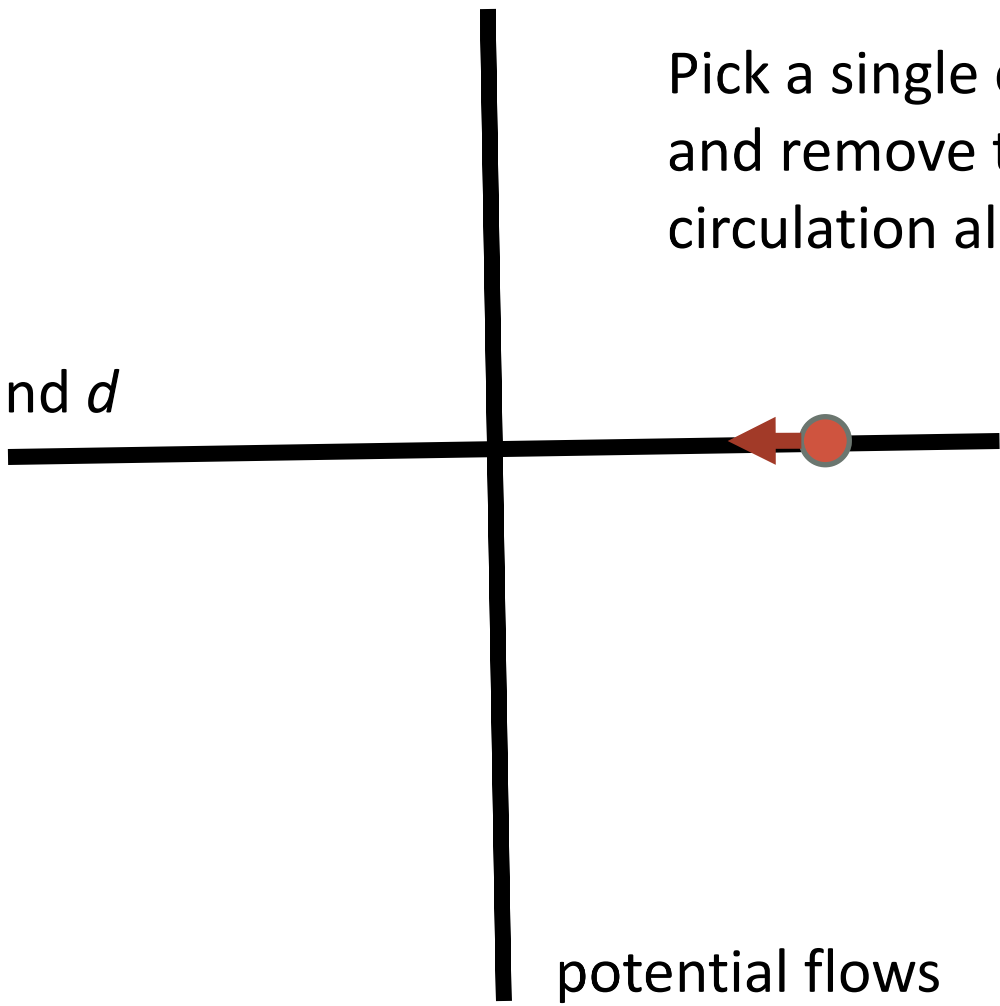
flows of demand *d*



potential flows

PRIMAL APPROACH: SINGLE STEP (KOSZ '13)

Pick a single cycle and remove the circulation along it



POTENTIAL BASED SOLVERS [SPIELMAN-TENG`04] [KOUTIS-M-PENG`10, `11]

Input: n by n SDD matrix A with m non-zeros
vector b

Output: Approximate solution Ax = b

Runtime: O(m log n)

[Blelloch-Gupta-Koutis-M-Peng-Tangwongsan. `11]:
Parallel solver, $O(m^{1/3})$ depth and nearly-linear work

FLOW BASED SOLVER [KELNER-ORECCHIA-SIDFORD-ZHU `13] [LEE-SIDFORD `13]

Input: n by n SDD matrix A with m non-zeros, demand b
Output: Approximate minimum energy electrical flow
Runtime: $O(m \log^{1.5} n)$

POTENTIAL BASED SOLVER AND ENERGY MINIMIZATION

- Suppose that A is SPD:

  Claim: minimizing $\frac{1}{2} x^T A x - x^T b$

  gives solution to $Ax = b$.

  Note: Gradient = $Ax-b$

Thus solving these systems are quadratic minimization problems!

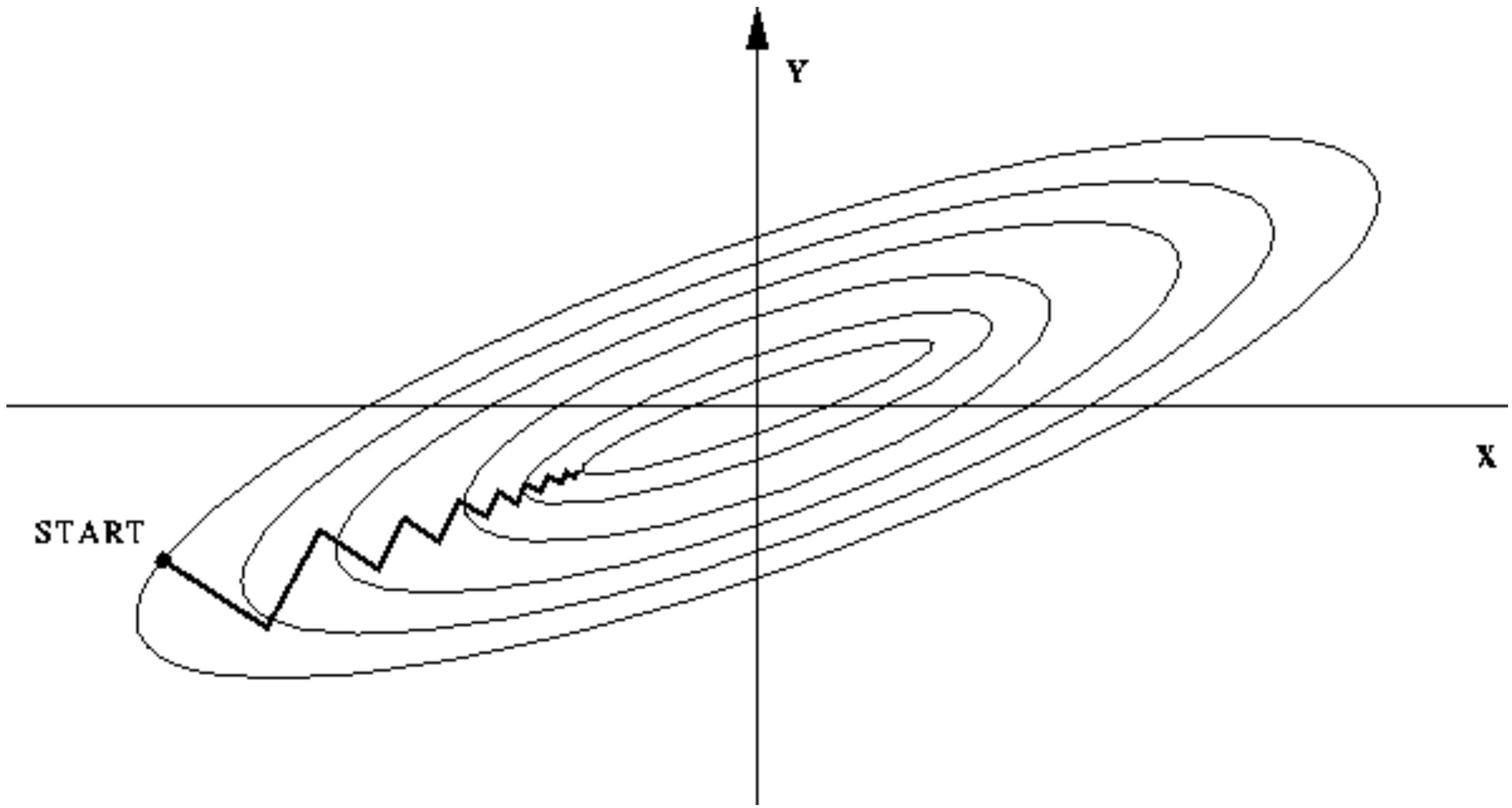
ITERATIVE METHOD
GRADIENT DESCENT

- Goal: approx solution to Ax = b
- Start with initial guess $u^0 = 0$
- Compute new guess

$$u^{(i+1)} = u^{(i)} + (b - Au^{(i)})$$

This maybe slow to converge or not converge at all!

STEEPEST DESCENT



PRECONDITIONED ITERATIVE METHOD

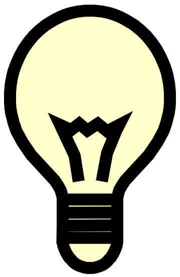
- Goal: approx solution to $B^{-1}Ax = B^{-1}b$
- Start with initial guess $u^0 = 0$
- Compute new guess

$$u^{(i+1)} = u^{(i)} - B^{-1}(b - Au^{(i)})$$

Recursive solve $Bz=y$ where $y=(b\text{-}Au^{(i)})$.

PRECONDITIONING WITH A GRAPH

[Vaidya `91]: Since A is a graph, B should be as well.
Apply graph theoretic techniques!



And use Chebyshev acceleration

PRECONDITIONING WITH A GRAPH

Vaidya `91: Since A is a graph, B should be as well.
Apply graph theoretic techniques!

Vaidya used maximum weight spanning trees

Plus some nontree edges

We will use low stretch trees
and sampled nontree edges

PROPERTIES B NEEDS

2 ways of being smaller:
Reduction in vertex count
Reduction in edge count

- Easier to solve
- Spectrally close to A

Can use partial Gaussian elimination to reduce vertex count if edge count is small

Will only focus on decreasing edge count, but maintaining spectral structure.

SPECTRAL SPARSIFICATION BY EFFECTIVE RESISTANCE

What probability $P(e)$ should we sample an edge?

Answer: [Spielman-Srivastava `08]:

- Set $P(e) = R(u,v)$ effective resistance from u to v.
- For each sample set edge set weight to $1/P(e)$
- Sample $O(n \log n)$ times.

spectral sparsifier with $O(n\log n)$ edges for any graph

Foster: $\sum_e R(e) = n-1$

THE CHICKEN AND EGG PROBLEM

How To Calculate Sample Probablities?

[Spielman-Srivastava `08]: Use Solver

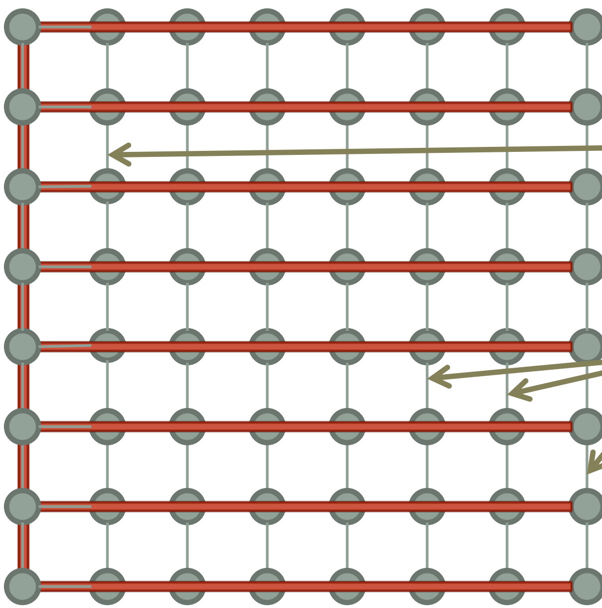
[Spielman-Teng `04]: Need Sparsifier

Workaround: upper bound using Low Stretch Spanning Trees

CHOICE OF TREES MATTER

$n^{1/2}$-by-$n^{1/2}$ unit weighted mesh

'haircomb' tree is both shortest path tree and max weight spanning tree



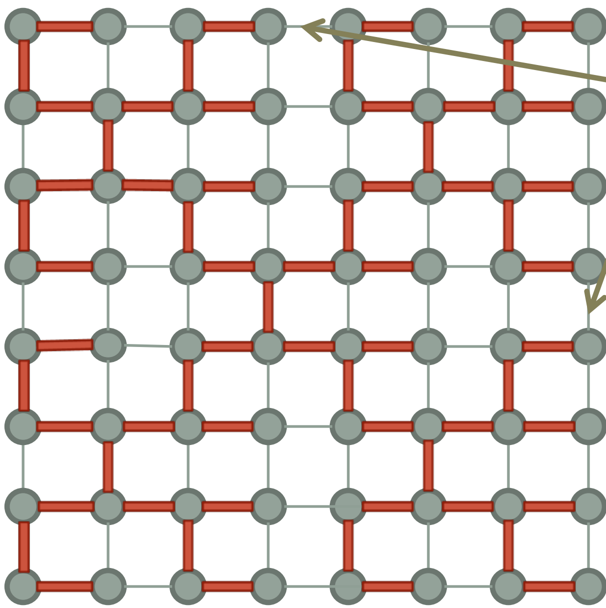
stretch(e)= O(1)

stretch(e)=O($n^{1/2}$)

total stretch = O($n^{3/2}$)

AN O(N LOG N) STRETCH TREE

Recursive 'C' Construction



stretch(e) still $O(n^{1/2})$

But only $O(n^{1/2})$ such edges

logn levels,
total = O(nlogn)

Able to obtain good trees for any graph by leveraging this type of tradeoffs

LOW STRETCH SPANNING TREES

[Abraham-Bartal-Neiman ’08,
Koutis-M-Peng `11,
Abraham-Neiman `12]:
A spanning tree with
total stretch O(m log n) in
O(m log n) time.

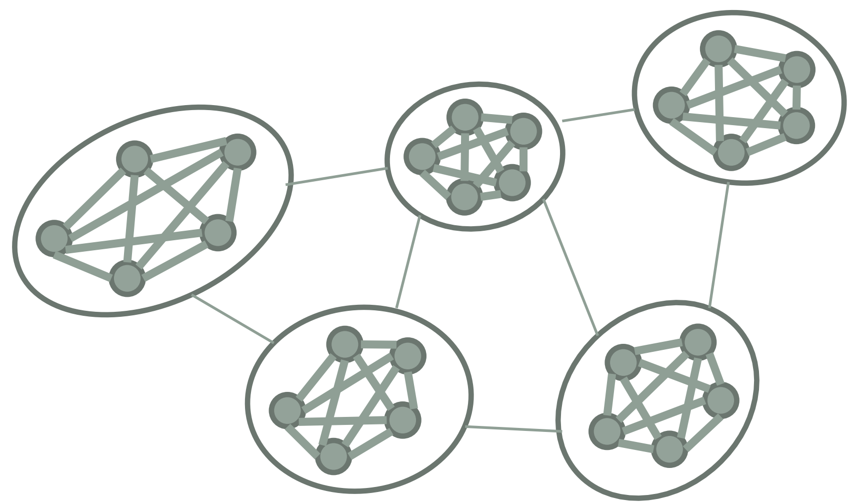
KEY TOOL IN DECOMPOSING GRAPHS

Low Diameter Decompositions

- Partition of V into clusters $S_1, S_2, \ldots, S_k$ s.t.
- The diameter of each S_i is at most d.
- βm edges between clusters.

Typically parameters:

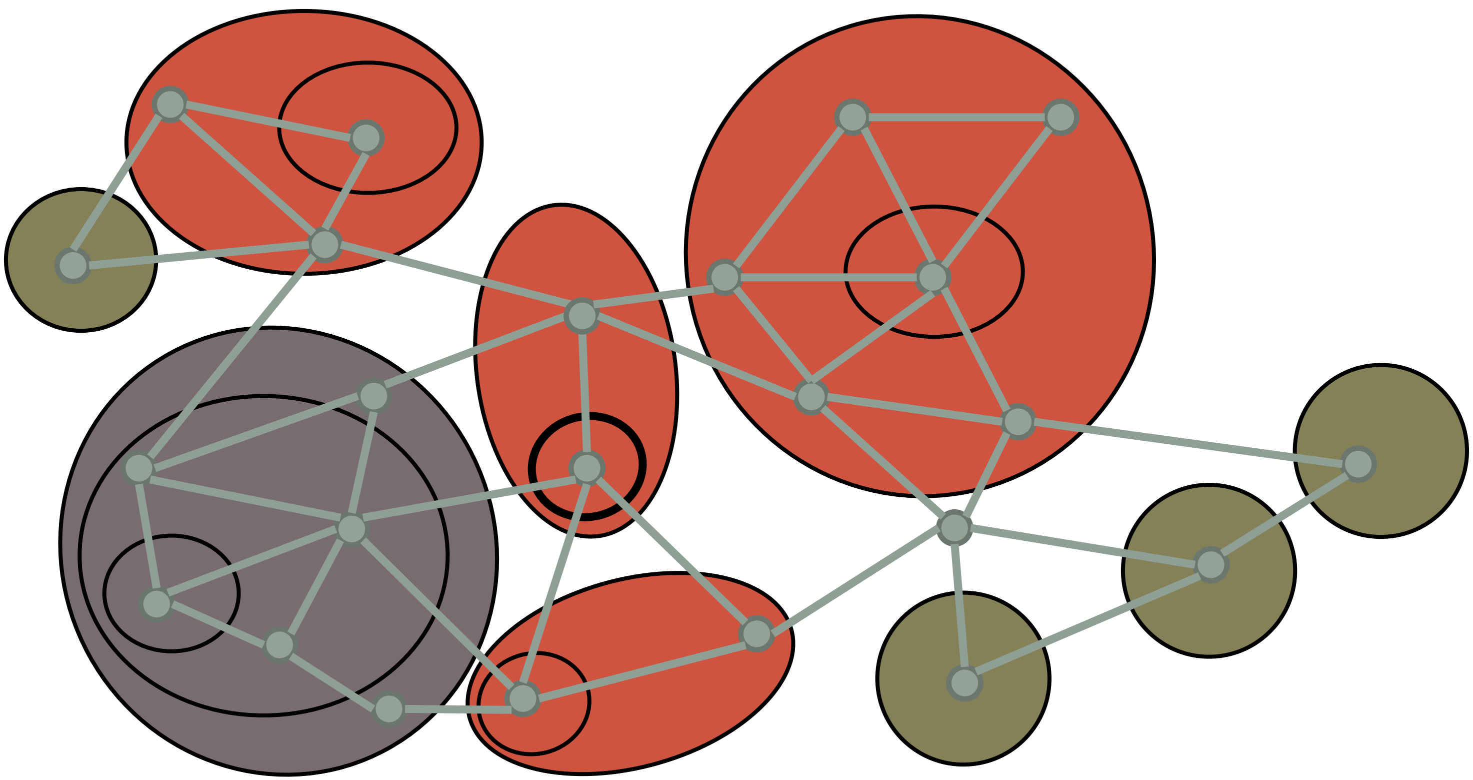
- $\beta = \log^{-O(1)} n$,
- $d = O(\log n / \beta)$



EXP START TIME CLUSTERING

Parallel variant of a clustering scheme in [Bartal `96]

- Each vertex u starts unit speed BFS at time -Exp(β)
- BFS stops at ‘owned’ v, owns any ‘sleeping’ v reached.

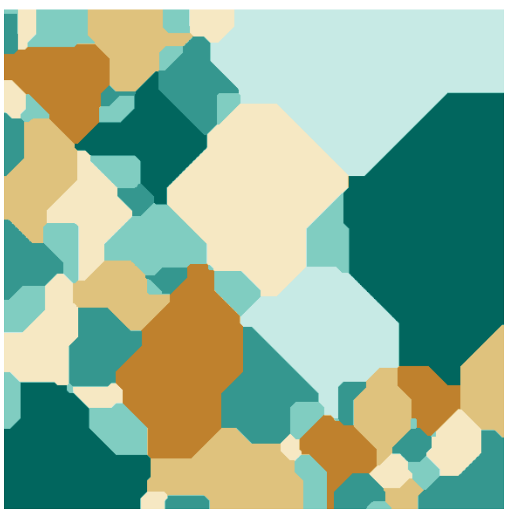


EXP START TIME CLUSTERING ON GRID

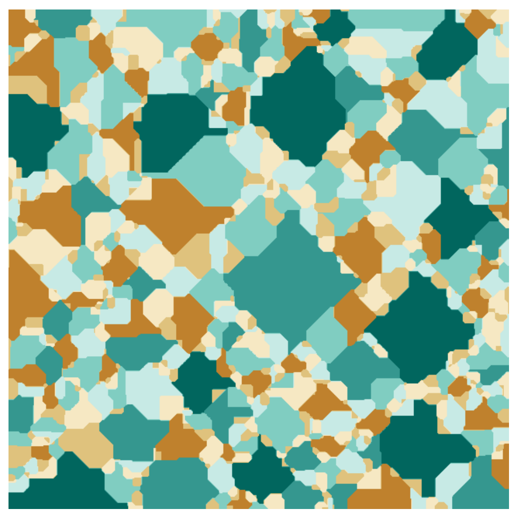
β=0.002

β=0.005

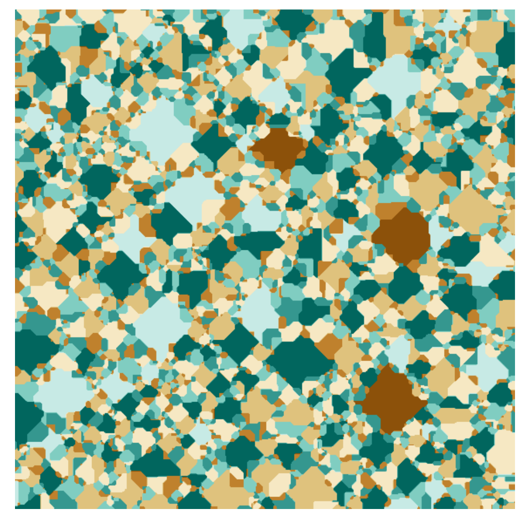
β=0.01



β=0.02



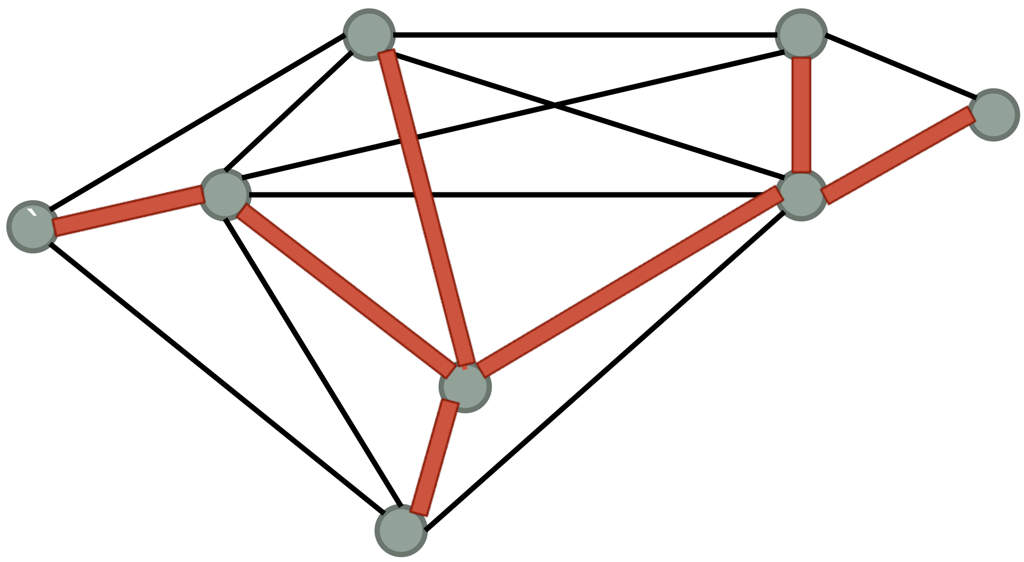
β=0.05



β=0.1

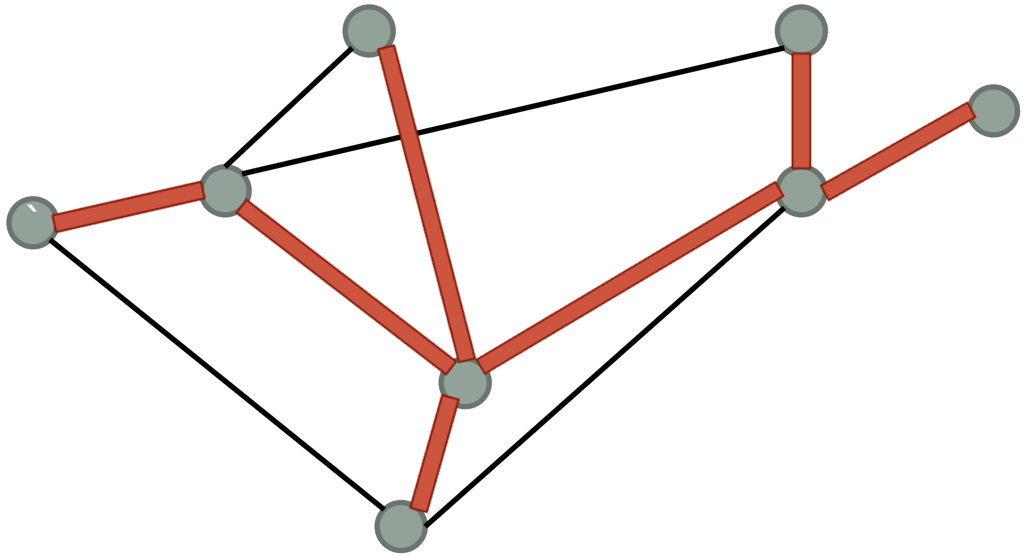
SOLVER IN ACTION

Sample off tree edges where
P(e) = 1/(stretch of edge).



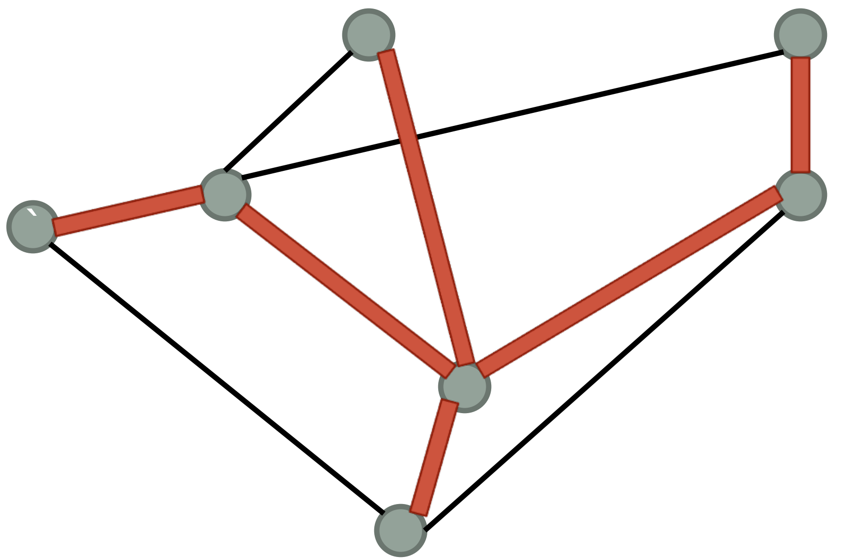
SOLVER IN ACTION

Eliminate degree 1 or 2 nodes



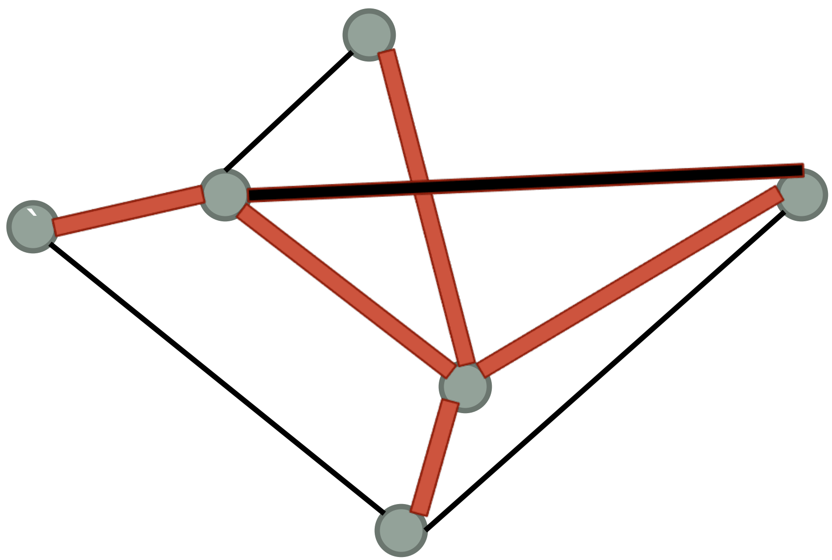
SOLVER IN ACTION

Eliminate degree 1 or 2 nodes



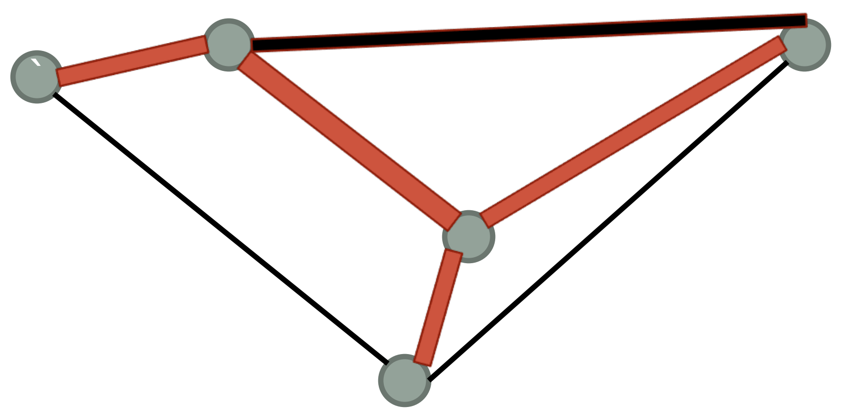
SOLVER IN ACTION

Eliminate degree 1 or 2 nodes



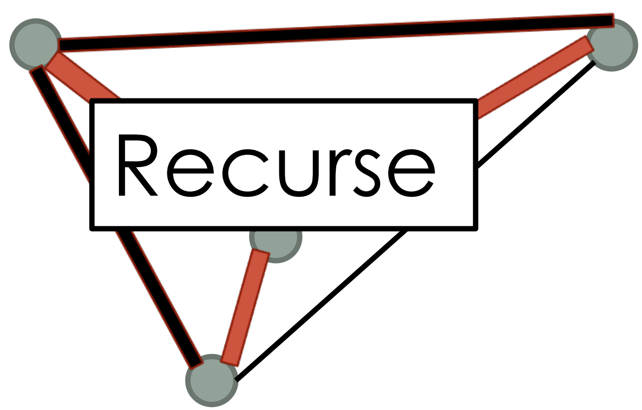
SOLVER IN ACTION

Eliminate degree 1 or 2 nodes

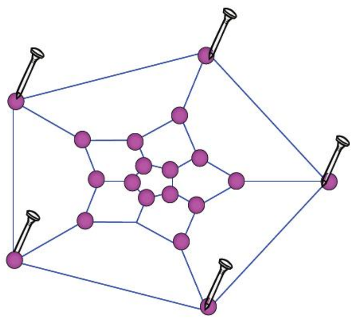


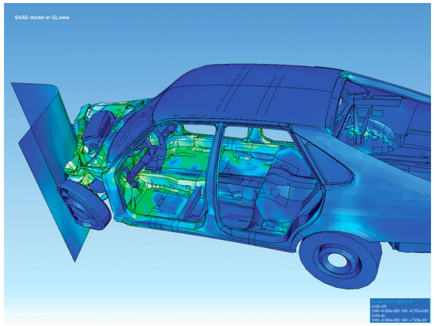
SOLVER IN ACTION

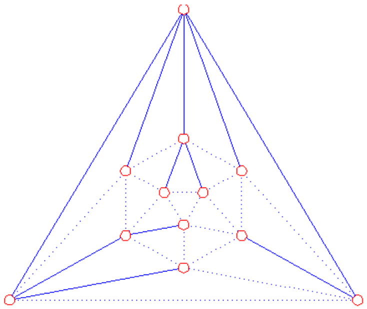
Eliminate degree 1 or 2 nodes

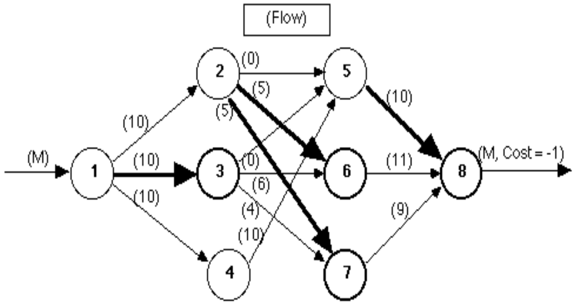


THEORETICAL APPLICATIONS OF SDD SOLVERS: MULTIPLE ITERATIONS



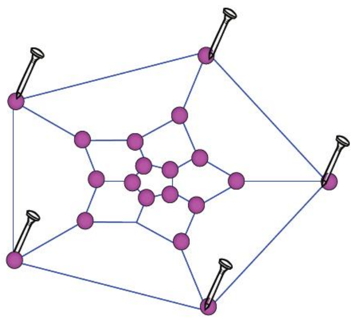


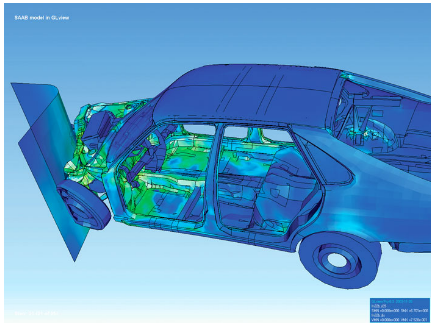


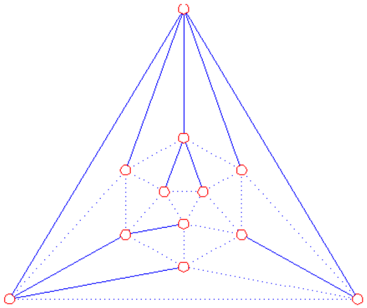


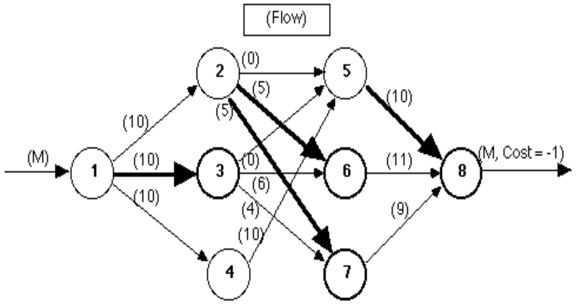
[Tutte `62] Planar graph embeddings.
[Boman-Hendrickson-Vavasis `04] Finite Element PDEs
[Zhu-Ghahramani-Lafferty, Zhou-Huang-Scholkopf `03,05] learning on graphical models.
[Kelner-Mądry `09 `15] Generating random spanning trees in $O(mn^{4/3})$ time by speeding up random walks.

THEORETICAL APPLICATIONS OF SDD SOLVERS: MULTIPLE ITERATIONS









[Daitsch-Spielman `08] Directed maximum flow, Min-cost-max-flow, lossy flow all can be solved via LP interior point where pivots are SDD systems in $O(m^{3/2})$ time.

BACK TO IMAGE DENOISING

PROBLEM WITH QUADRATIC OBJECTIVE

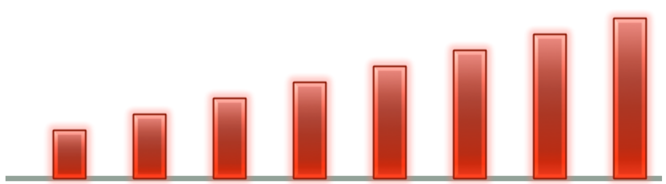
- Result too ‘smooth’, objects become blurred
- Quadratic functions favor the removal of boundaries

FUNCTION ACCENTUATING BOUNDARIES

L_1 smoothness term

If $a<b<c$, $|a-b|+|b-c|$ doesn't depend on b

s: sharp boundary



Same smoothness term

Better fidelity

TOTAL VARIATION OBJECTIVE

- [Rudin-Osher-Fatemi, 92] Total Variation objective: L_2^2 fidelity term, L_1 smoothness.

Fidelity(x, s) + Smoothness(x)

$(x_i-s_i)^2$ summed over all pixels i.

$|x_i-x_j|$ summed over all neighbor pixels i, j.

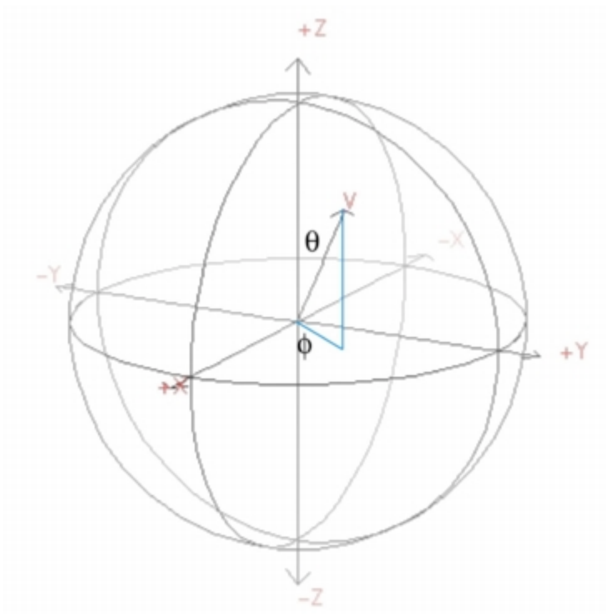
TOTAL VARIATION MINIMIZATION

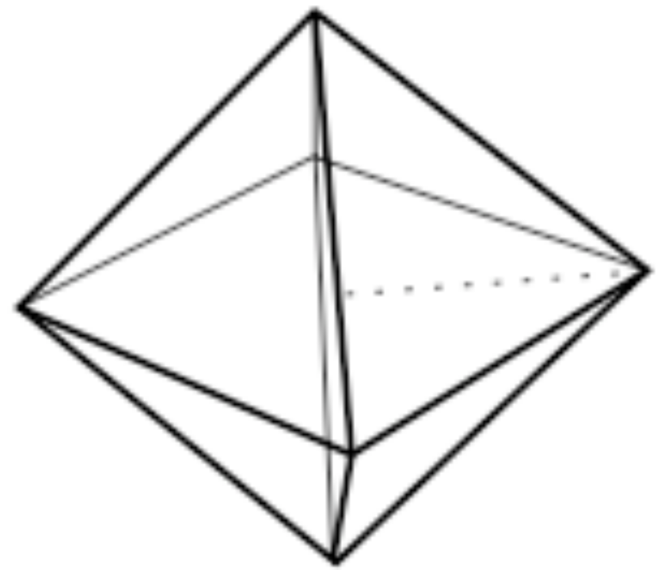
Higher weight on smoothness term

Effect: sharpen boundaries

Overdoing makes image cartoon like

WHAT'S HARD ABOUT L_1?





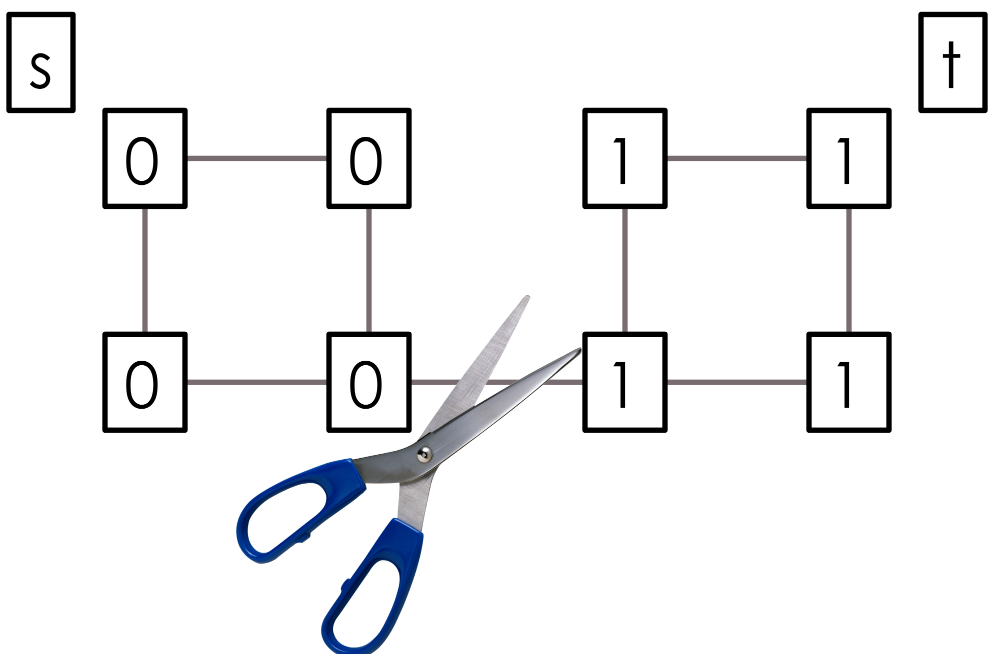
Absolute value function on n variables has 2^n points of discontinuity, $L_2{}^2$ has none.

MIN CUT PROBLEM AS L_1 MINIMIZATION

Minimum s-t cut:

minimize $\sum |x_i - x_j|$

subject $x_s=0$, $x_t=1$



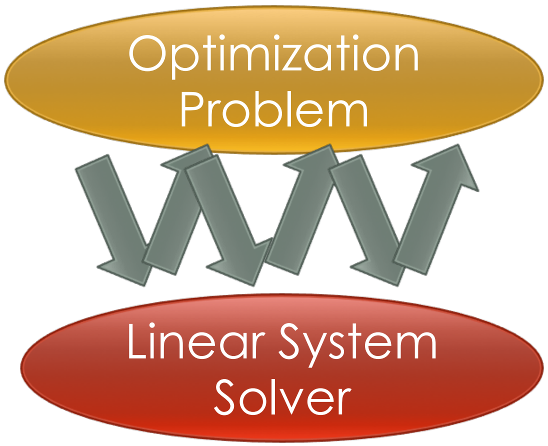
MINCUT VIA. L_2 MINIMIZATION

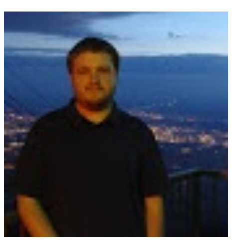
[Christiano-Kelner-Mądry-Spielman-Teng `11]: undirected max flow and mincut can be approximated using $\tilde{O}(m^{1/3})$ SDD solves.

- Multiplicative weights update method
- Repeatedly update the edge weights of the linear systems being solved

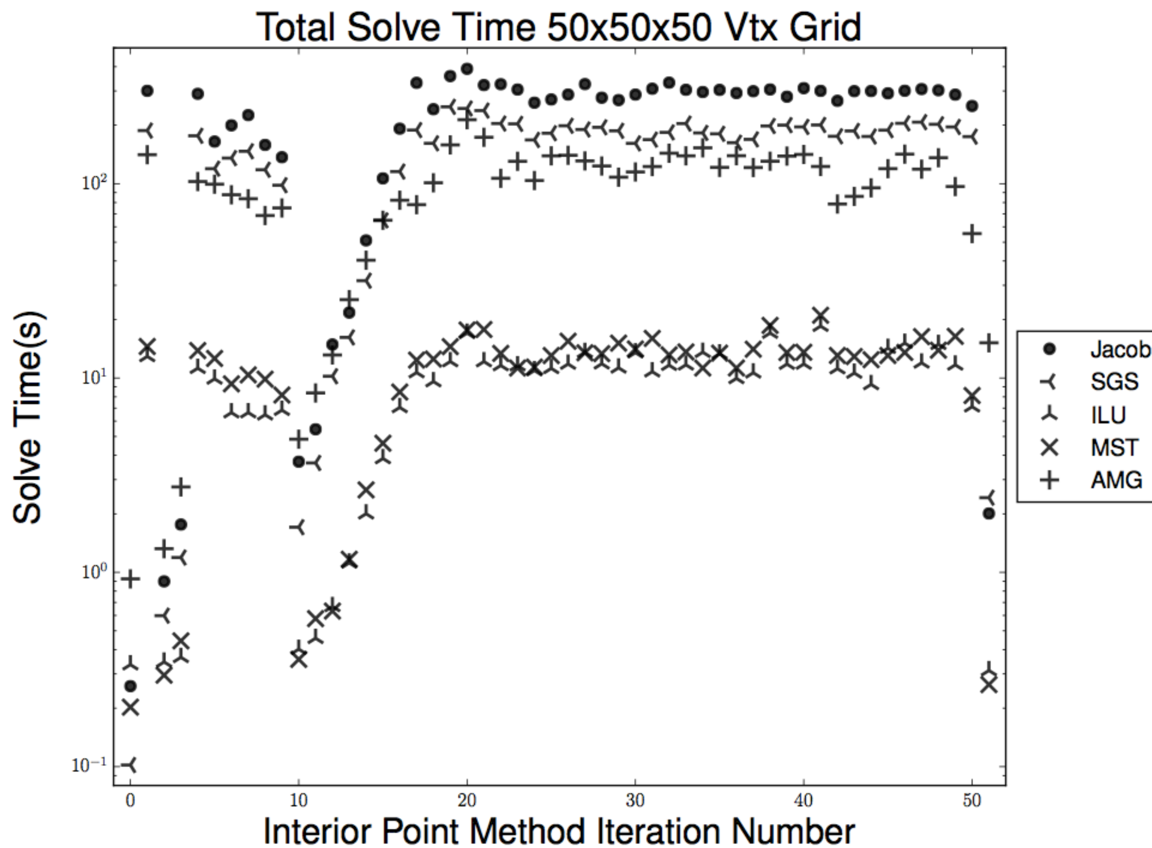
Total: $\tilde{O}(m^{4/3})$

SEQUENCE OF (ADPATIVELY) GENERATED LINEAR SYSTEMS





Kevin Deweese



EVEN FASTER SOLVERS

- Cohen-Kyng-Pachocki-Peng-Rao `13
  SDD linear systems Faster solver in
  - $O(m\log^{1/2} n)$ time given a LSST.

- The log appears in two places in KMP:

1. Matrix Chernoff Bounds
2. LSST tree construction

FASTER TREE GENERATION

- Koutis-M-Peng `11, Abraham-Neiman `12]:

LSST with stretch O(m log n) in O(m log n) time.

We do not know how to beat these bounds!

We find a tree that is good enough!

FUTURE WORK

- Practical/parallel implementations?
  - The win over sequential is parallel!
- Near linear time exact max flow?
  - $\log(1/\varepsilon)$ dependency in runtime?
- Sub-quadratic SPD solver?

JOINT WORK

Guy Blelloch,
Hui Han Chin,
Michael Cohen,
Anupam Gupta,
Jonathan Kelner,
Yiannis Koutis,
Alexsander Madry,
Jakub Pachocki,
Richard Peng,
Kanat Tangwongsan,
Shen Chen Xu