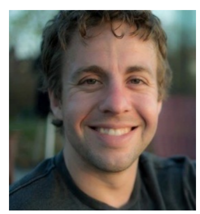
Interior Point Methods for Nearly Linear Time Algorithms

Aaron Sidford



Stanford University
Departments of Management Science
& Engineering and Computer Science

Contact Info:
- email: *sidford@stanford.edu*
- website: *www.aaronsidford.com*

Thank You Prof. Yurii Nesterov!

Laplacian System Solving

- Coordinate Descent

Undirected Maximum Flow

- Nesterov's presentation of first-order methods

Linear Programming

- Self-concordance
- Univeral barrier

Acceleration

- Non-convex optimization
- Ball-constrained optimization oracle
- Max-type functions

Dual Extrapolation

- Optimal transport
- Acceleration

Solving Tall Dense Linear Programs in Nearly Linear Time

joint work with Jan van den Brand, Yin Tat Lee, Zhao Song

Bipartite Matching in Nearly-linear Time on Moderately Dense Graphs

joint work with Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Zhao Song, and Di Wang

Minimum Cost Flows, MDPs, and ℓ_1-Regression in Nearly Linear Time for Dense Instances

joint work with Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, Di Wang

***Note**: will hide many details throughout the talk and simplify many parts. Happy to discuss details after.*

This Talk

Part 1: Overview

- Survey recent history of interior point methods for linear programming
- Present and motivate recent nearly linear time algorithms

Part 2: IPM

- Brief intro to recent IPM advances and our new IPMs

Part 3: Data Structures

- Data structures for efficiently implementing our IPMs
- Highlight difficulty and key techniques

Linear Programming

Primal

$$\min_{x \geq 0:\, A^\top x = b} c^\top x$$

Dual

$$\max_{y:\, Ay \geq c} b^\top y$$

The Picture

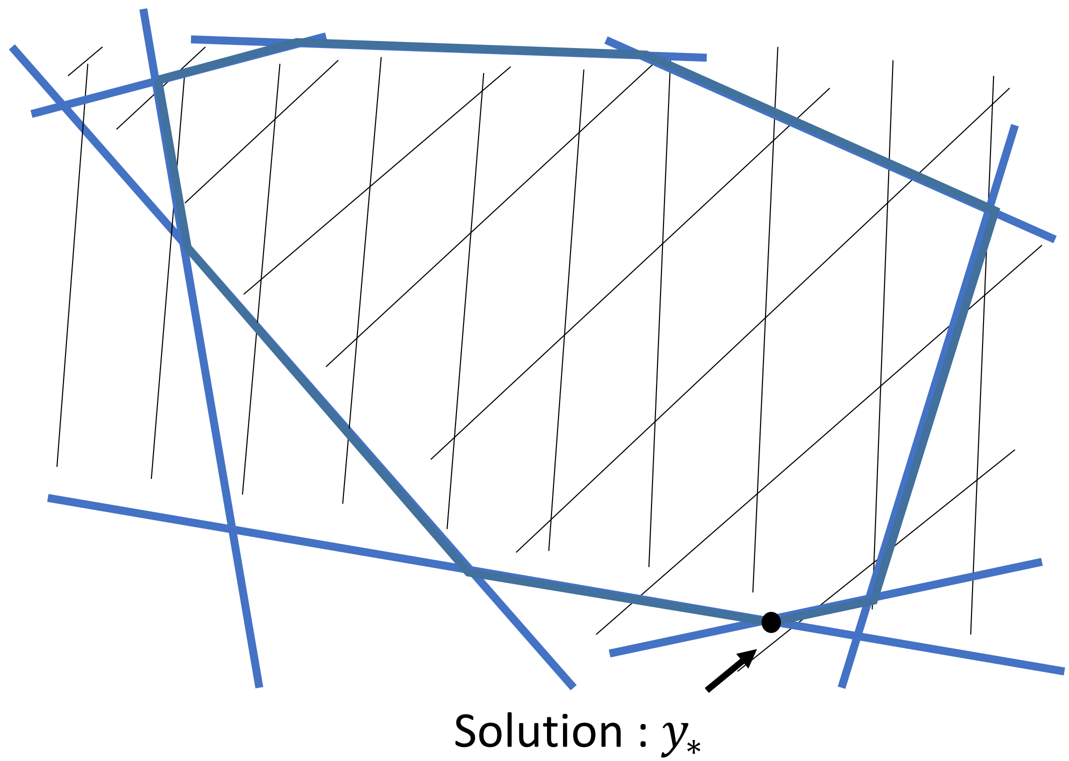
$$\max_{y\,:\,Ay \geq c} \; b^\top y$$

- Variables : $y \in \mathbb{R}^d$
- Cost vector : $b \in \mathbb{R}^d$
- Constraint matrix : $\mathbf{A} \in \mathbb{R}^{n \times d}$
- Constraint vector : $c \in \mathbb{R}^n$

$$Ay \geq c$$

$$\begin{pmatrix} - & a_1 & - \\ - & a_2 & - \\ & \vdots & \\ - & a_k & - \\ & \vdots & \\ - & a_n & - \end{pmatrix} y \geq \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \\ \vdots \\ b_n \end{pmatrix}$$

$$b^\top y$$

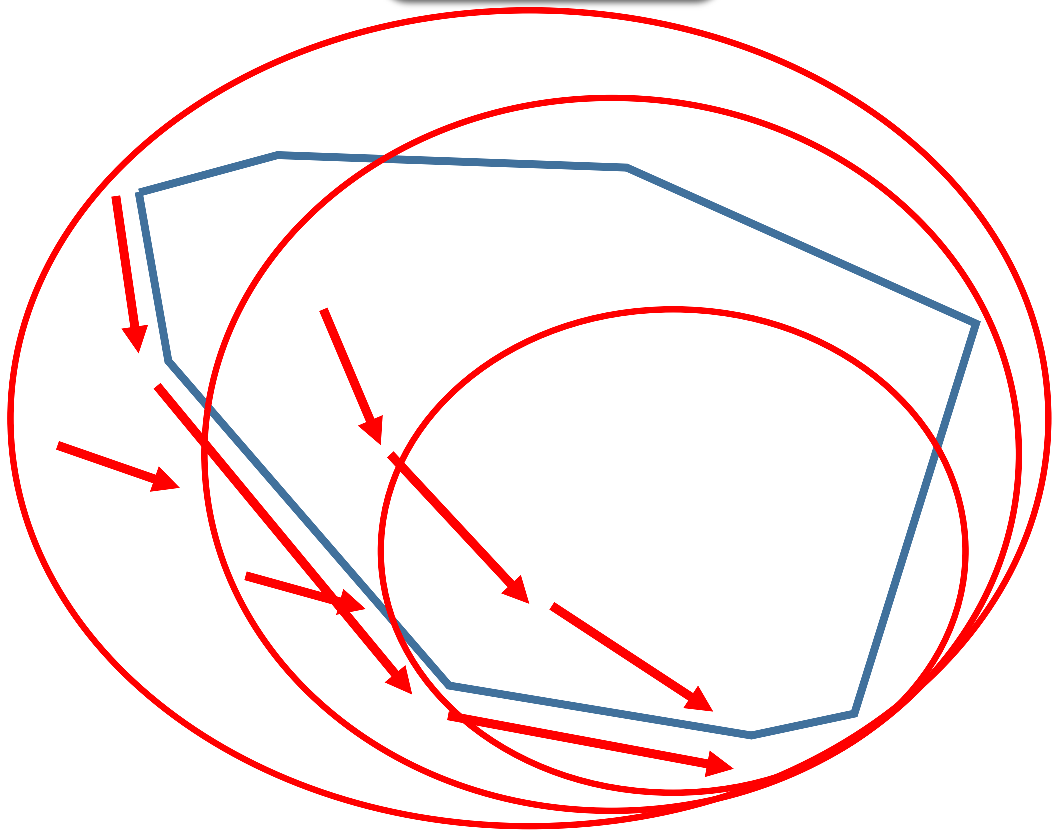


Standard Methods

$$\max_{y\,:\,Ay\geq c} b^\top y$$

Goal
high precision solutions in poly time

- Simplex
  - Fast in practice
  - Slow in theory
- Ellipsoid
  - Moderate in practice
  - Moderate in theory
- Interior Point
  - Often fastest in theory
  - Often fast in practice



Ignoring first order methods with polynomial dependence on condition number or accuracy

Why Study Interior Point Methods (IPM)?

In General?

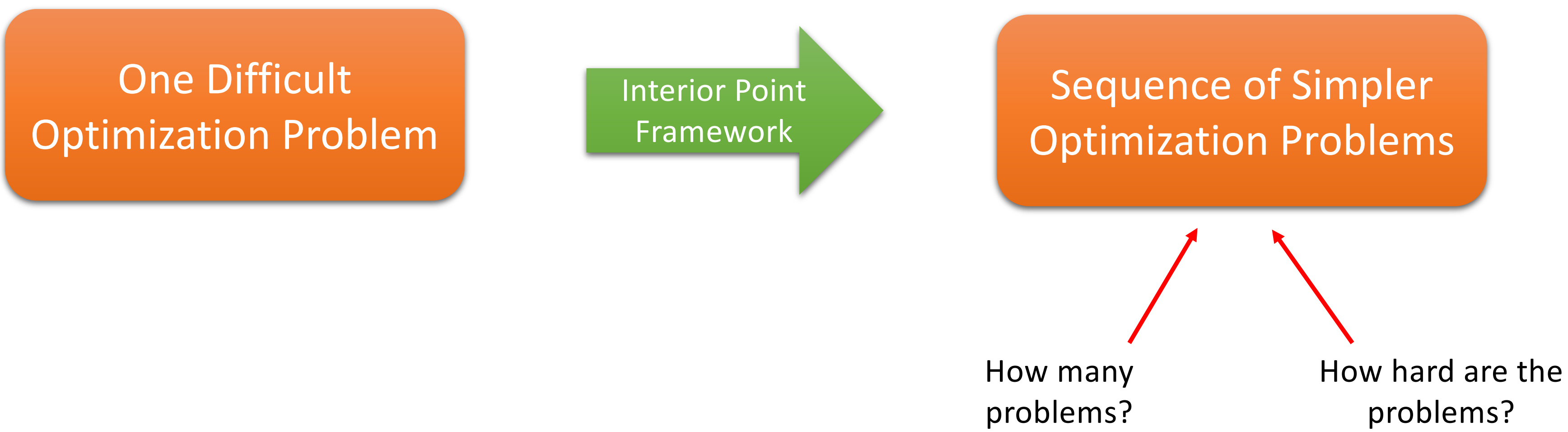
- General robust optimization framework
- Solves to high precision (i.e. linearly convergent)
- Deals with difficult problems (i.e. ill-conditioning)
- Can outperform theory (practice $\geq$ worst case)

Many open problems and active research area

As a Theorist?

- Tool for finding and exploiting problem structure and obtaining faster high-precision methods.
- Combinatorial optimization:
  - Fastest known algorithms in many problem settings
  - Minimum cost flow**:** [DS08,L**S**14]
  - Maximum flow: [M13,M16,L**S**20,KL**S**20,BLLSS**S**SW21,GLP21]
  - Shortest path negative edge lengths: [CMSV17, AMV20]
- Continuous optimization:
  - Fastest known algorithms in many problem settings
  - ℓ_1-Regression [L**S**15, CLS19, B20, JSWZ20, BLLSS**S**SW21]
  - Geometric median [CLMP**S**16]
  - Markov decision process [L**S**15 / **S**WWYY18, BLLSS**S**SW21]
  - Empirical risk minimization [LSZ19]

Interior Point Methods (IPM)



Previous Work

up to the last ~5 years

Primal
$\min_{x\geq 0:\, A^\top x = b} c^\top x$

Dual
$\max_{y:\, Ay\geq c} b^\top y$

$A \in \mathbb{R}^{n\times d}$
$n \geq d$

for diagonal $\boldsymbol{D}$ compute $(\boldsymbol{A}^\top \boldsymbol{D}^2 \boldsymbol{A})^{-1} f$ *or* $\min_{y\in\mathbb{R}^d} \|\boldsymbol{D}\boldsymbol{A}y - e\|_2^2$

Year	Author(s)	Iteration Count $\tilde{O}(\cdot)$	Iteration Cost
1984	Karmarkar	n	Solve 1 linear system
1986	Renegar	$\sqrt{n}$	Solve 1 linear system
1989	Vaidya	d	Solve d linear systems
1989	Vaidya	$(nd)^{1/4}$	Solve n linear systems
1994	Nesterov & Nemirovskii	$\sqrt{d}$	Volume of polar polytope

Project onto image of rescaled $\boldsymbol{A}$

Compute projection matrix
(or its diagonal)

$\mathrm{diag}(\boldsymbol{D}\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{D}^2 \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{D})$

"Universal Barrier"
Applies to all convex optimization

Harder than linear programming

Lee & **Sidford** 14 / 19

An $\tilde{O}(\sqrt{d})$ iteration algorithm that solves $\tilde{O}(1)$ linear systems per iteration.

No general improvement to previous slide in past 5 years.

How do we improve further?

progress over the last ~5 years

One Difficult Optimization Problem

Interior Point Framework

Solve Dynamic Data Structure Problems

Sequence of Simpler Optimization Problems

How many problems?

How hard are the problems?

Key Take Aways

- Improved IPMs ⇒ “easier” data structure problems
- Improved data structure ⇒ “near optimal” algorithms

$w < 2.373$ is current matrix multiplication constant [W13]

Running Times

previous state-of-the art

- Long history of runtime improvements by improving linear system [K84,NN89,V89,L**S**14,L**S**15,CLS19,B19]

[LS14,LS**15,L**S**19]**

$\tilde{O}((\mathrm{nnz}(\boldsymbol{A}) + d^2)\sqrt{d})$

- [LS14 / LS19] IPM , $\tilde{O}(\sqrt{d})$ iteration of system solving
- [LS 15] "inverse maintenance" data structures

Previous best known running time for large (poly-bounded) n.

Question

Can we improve further?

[CLS19,B19]

$\tilde{O}(n^{\omega})$

- "robust" primal dual variant of [R84], $\tilde{O}(\sqrt{n})$ iteration
- Precise projection matrix maintenance and approximate iterate maintenance

Matches best known running time for finding x with $\boldsymbol{A}^{\top}x = b$ when $n \approx d$ and no sparsity assumptions. [PV20]

Question

Can we solve large instances optimally?

Running Times

Primal

$\min_{x \geq 0:\, A^\top x = b} c^\top x$

Dual

$\max_{y:\, Ay \geq c} b^\top y$

$A \in \mathbb{R}^{n \times d}$
$n \geq d$

[LS14,LS15,LS19]
$\tilde{O}((\mathrm{nnz}(\boldsymbol{A}) + d^2)\sqrt{d})$

[CLS19,B19*]
$\tilde{O}(n^\omega)$

Our Results
$\tilde{O}(nd + \mathrm{poly}(d))$

Suppose

- Input is dense, $\mathrm{nnz}(\boldsymbol{A}) = \tilde{\Omega}(nd)$
- Input is tall, $n = \Omega(\mathrm{poly}(d))$

⇒ Nearly linear / near-optimal time!

"Solving Tall Dense Linear Programs in Nearly Linear Time" $\tilde{O}(nd + d^3)$

"Minimum Cost Flows, MDPs, and ℓ_1-Regression in Nearly Linear Time for Dense Instances" $\tilde{O}(nd + d^{2.5})$

More nearly linear time algorithms!

How?

Primal

$$\min_{x\geq 0:\, A^\top x = b} c^\top x$$

Dual

$$\max_{y:\, Ay \geq c} b^\top y$$

$A \in \mathbb{R}^{n\times d}$

$n \geq d$

[LS14,LS15,LS19]

$\tilde{O}((\text{nnz}(\boldsymbol{A}) + d^2)\sqrt{d})$

[CLS19,B19*]

$\tilde{O}(n^\omega)$

Ingredient #1

- A new IPM
- Simplified primal-dual LS14
- Stable variants that allows approximate projections

Our Results

$\tilde{O}(nd + \text{poly}(d))$

Ingredient #2, 3, 4, 5, …

- New data structures
- Sketching to avoid looking at all constraints each iteration.
- Dealing with adaptive queries (iterates depend on output)

Broadly

- Dynamic sampling / sketching (as opposed to fixed in regression)
- Maintain approximate projections / change of basis (as opposed to high precision in previous)

Why Study Interior Point Methods (IPM)?

In General?

- Pervasive robust optimization framework
- Solves to high precision (i.e. linearly convergent)
- Deals with difficult problems (i.e. ill-conditioning)
- Can outperform theory (practice ≥ worst case)

What about combinatorial optimization?

As a Theorist?

- Tool for finding and exploiting problem structure and obtaining faster high-precision methods.
- Combinatorial optimization:
  - Fastest known algorithms in many problem settings
  - Minimum cost flow: [DS08,L**S**14]
  - Maximum flow: [M13,M16,L**S**20,KL**S**20,BLLSS**S**SW21,GLP21]
  - Shortest path negative edge lengths: [CMSV17, AMV20]
- Continuous optimization:
  - Fastest known algorithms in many problem settings
  - ℓ_1-Regression [L**S**15, CLS19, B20, JSWZ20, BLLSS**S**SW21]
  - Geometric median [CLMP**S**16]
  - Markov decision process [L**S**15 / **S**WWYY18, BLLSS**S**SW21]
  - Empirical risk minimization [LSZ19]

Minimum Cost Transshipment

Primal

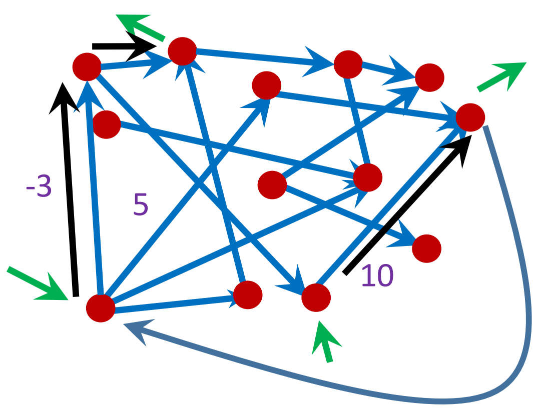
$$\min_{x \geq 0:\, A^\top x = b} c^\top x$$

Dual

$$\max_{y:\, Ay \geq c} b^\top y$$

Continuous

- $A \in \mathbb{R}^{m \times n}$ is graph incidence matrix
- Each row is all-zero except for exactly one 1 and one -1.



Combinatorial

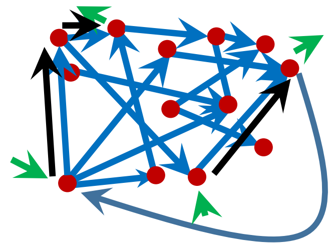
- Directed graph graph $G = (V, E)$
- Vertex imbalances $b \in \mathbb{R}^V$
- Edge costs $c \in \mathbb{R}^E$
- **Goal**: find flow $x \in \mathbb{R}^E_{\geq 0}$ that routes demands b and minimizes cost $c^\top x$

Row j has 1 at i and -1 at j if edge from i to j

Why?

Generalizes minimum cost matching in bipartite graph and shortest path with negative edge lengths.

State of the art



- m-edge, n-node graph
- Integral costs and demands
- $\tilde{O}$ hides $\mathrm{poly}(\log(\max\{n, \|b\|_\infty, \|c\|_\infty\}))$

Minimum Cost Transshipment

Year	Authors	Runtime $\tilde{O}(\cdot)$
1972	Edmonds, Karp	mn
2008	Daitch, Spielman	$m^{3/2}$
2014	Lee, **Sidford**	$m\sqrt{n}$
2020	Brand, Lee, Nanongkai, Peng, Saranurak, **Sidford**, Song, Wang	$\boldsymbol{m + n^{1.5}}$

Bipartite Matching

Year	Authors	Runtime $\tilde{O}(\cdot)$
1969-1973	Dinic, Karzanov, Hopcroft, Karp	$m\sqrt{n}$
1981	Ibarra, Moran	n^{ω}
2013	Mądry	$m^{10/7}$
2020	Liu, **Sidford**	$m^{11/8+o(1)}$
2020	Liu, Kathuria, **Sidford**	$m^{4/3+o(1)}$
2020	Brand, Lee, Nanongkai, Peng, Saranurak, **Sidford**, Song, Wang	$\boldsymbol{m + n^{1.5}}$

- All improvements since 1980s either use IPMs (or are from faster FMM)
- Recent result are first nearly linear time for any density (nearly linear whenever average degree $\Omega(\sqrt{n})$)

"Bipartite Matching in Nearly-linear Time on Moderately Dense Graphs"

Approach: graph-based data structures for new IPM

$w < 2.373$ is current fast matrix multiplication (FMM) constant [W13]

***Note**: will hide many details throughout the talk and simplify many parts. Happy to discuss details after.*

This Talk

Part 1: Overview

- Survey recent history of interior point methods for linear programming
- Present and motivate recent nearly linear time algorithms

Part 2: IPM

- Brief intro to recent IPM advances and our new IPMs

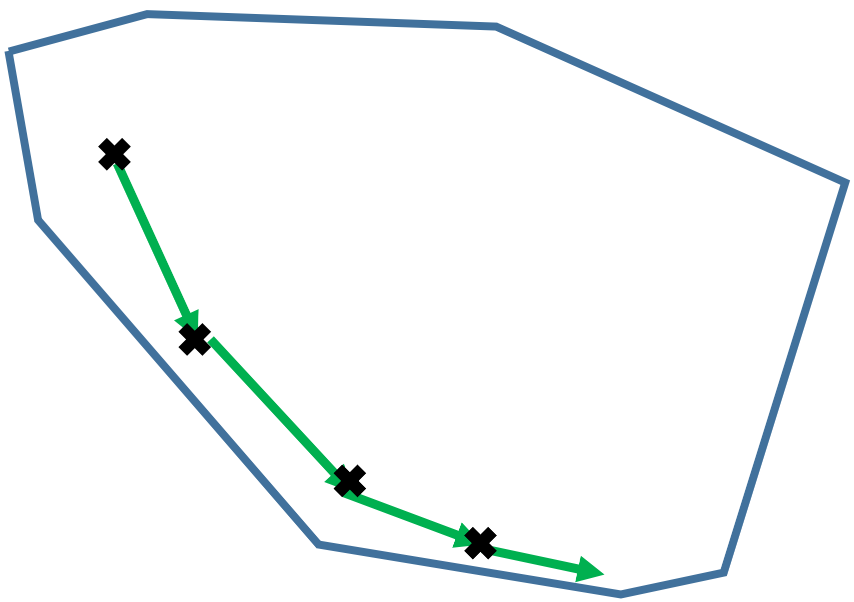
Part 3: Data Structures

- Data structures for efficiently implementing our IPMs
- Highlight difficulty and a key technique

Two Key Ideas for Interior Point Methods

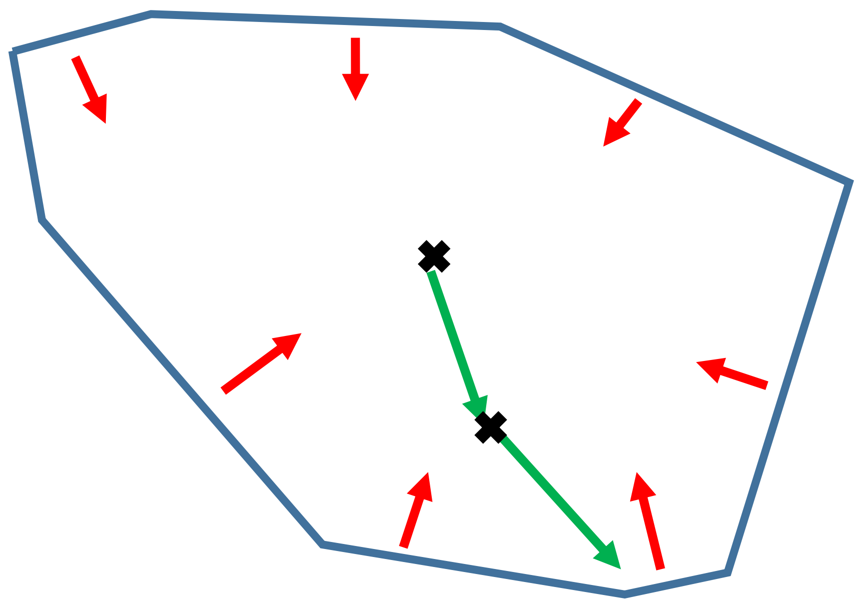
Idea #1: Stay in the Interior

- Maintain a feasible point
- Minimize cost over time



Idea #2: Really Stay in the Interior

- “barrier” to stay away from boundary



Path Following Warm-up

$$\max_{y:\, Ay \geq c} b^\top y$$

Barrier function

A "nice" function p from interior to $\mathbb{R}$ s.t.

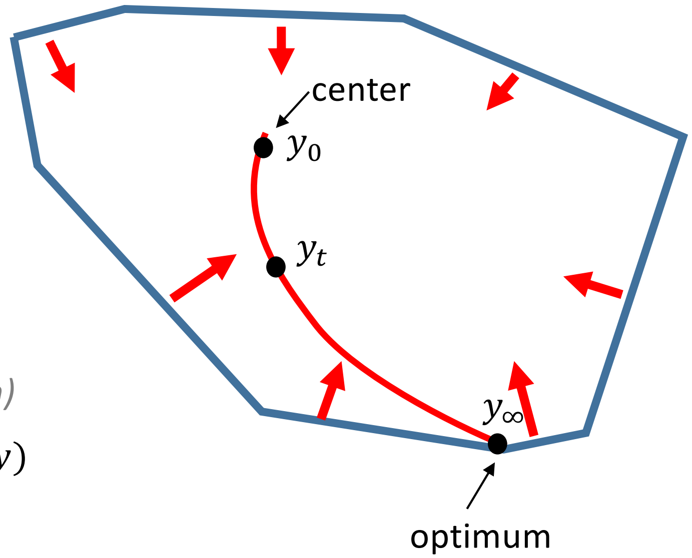
$$\lim_{y \to boundary} p(y) \to \infty$$

Penalized Objective

$$f_t(y) \stackrel{\text{def}}{=} t \cdot c^\top y + p(y)$$

Algorithm

- **Initialize**: $t > 0$ and $y_t \approx x_t$
- **Iterate**: repeat until $c^\top y_t$ small
- **Move path parameter**
  - $t \coloneqq (1 + \eta_t)t$
- **Center (i.e. Newton Steps)**
  - Until $y \approx x_t$ *(one step enough)*
  - $y \coloneqq y - \eta_c \left(\nabla^2 f_t(y)\right)^{-1} \nabla f_t(y)$
- **Idea**
  - Roughly $\tilde{O}(\eta_t^{-1})$ iterations suffice



Central Path

For path parameter $t > 0$ the minimizers $y_t = \operatorname{argmin}_y f_t(y)$ form the *central path* a continuous curve from *center* (y_0) to solution (y_∞).

Discretization* of the **central path**

Upshot

IPM reduces LP ⇒ solving sequence of linear systems

One of simplest instantiations

Renegar's Algorithm

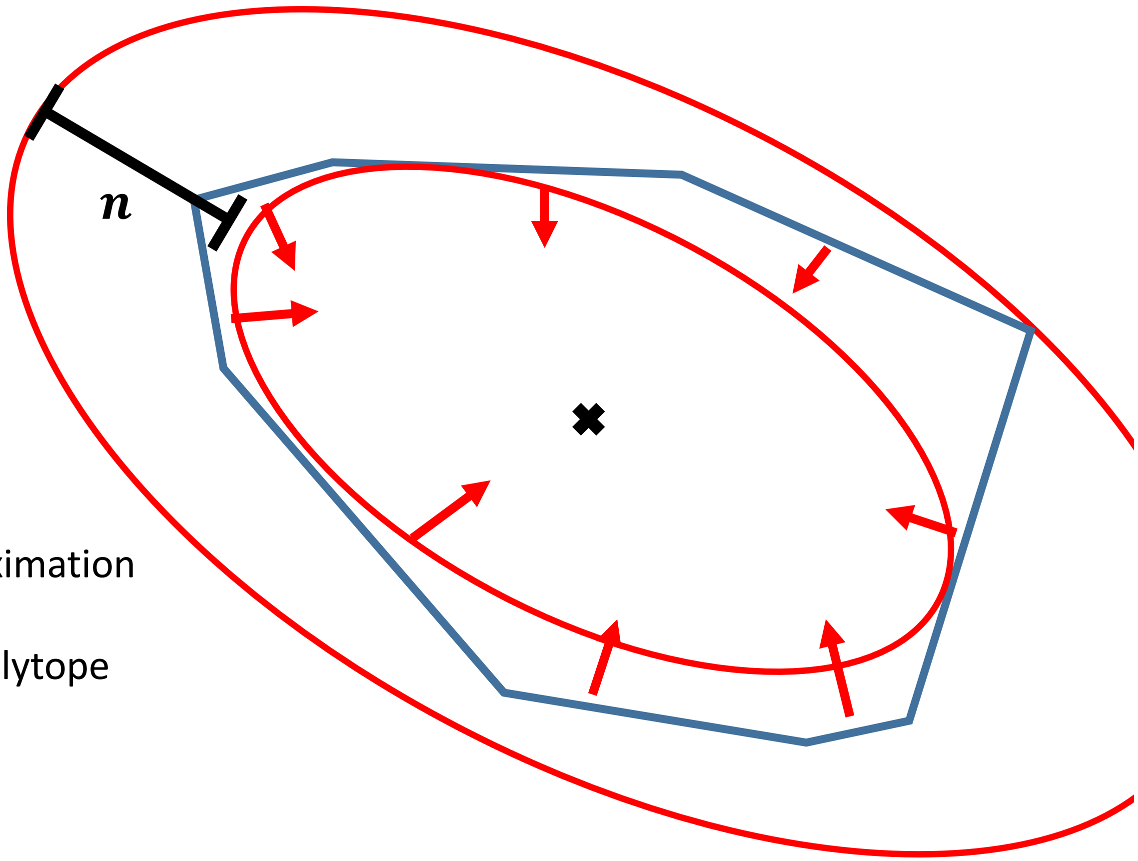
$$\min_{x\,:\,Ax\geq b} \; c^\top x$$

Log Barrier

$$p_\ell(y) = \sum_{i\in[n]} \ln\left(\frac{1}{a_i^\top y - b_i}\right)$$

Analytic Center

$$y_c = \operatorname{argmin}_y p_\ell(y)$$



- Dikin ellipse : contour of second order approximation
- n-rounding: scaling by factor of n contains polytope
- Yields $\tilde{O}(\sqrt{n})$ iteration algorithm

How improve?

$$\min_{y\,:\,Ay \geq b} c^\top y$$

Log Barrier

$$p_l(y) = \sum_{i \in [n]} \ln\left(\frac{1}{a_i^\top y - b_i}\right)$$

- **Problem** : adversary can effectively re-weight barrier

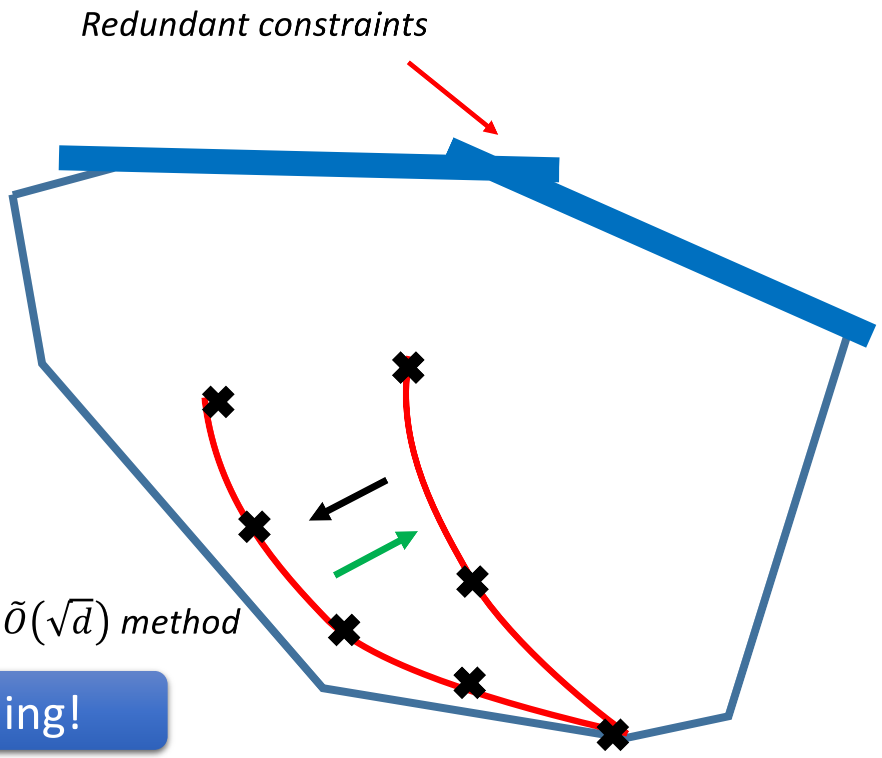
Weighted Log Barrier

$$p_w(y) = \sum_{i \in [n]} w_i \ln\left(\frac{1}{a_i^\top y - b_i}\right)$$

Leveraged for $\tilde{O}(\sqrt{d})$ method

- **Solution** : optimize over weights

[LS14] Path Finding!



Core to many recent combinatorial improvements [M13, M16, CMSV17, LS20, LST20, GLP21]

Helpful in many recent advances

A Primal-Dual View

Primal

$$\min_{x\geq 0:\, A^\top x = b} c^\top x$$

Dual

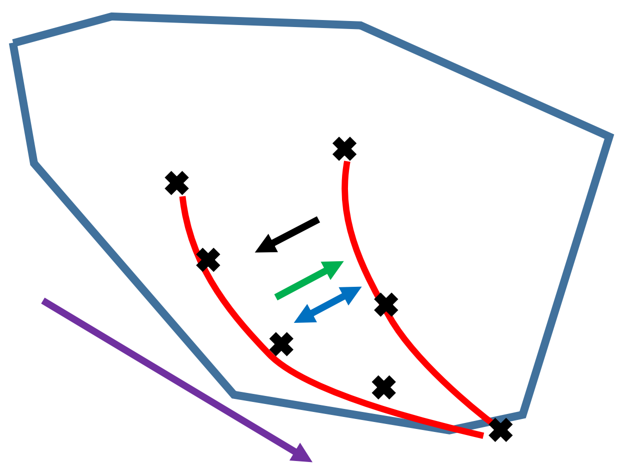
$$\max_{y:\, Ay \geq c} b^\top y$$

Weighted Log Barrier

$$p_w(y) = \sum_{i\in[n]} w_i \ln\left(\frac{1}{a_i^\top y - b_i}\right)$$

Weighted central path point

$$y_w = \text{argmin}_y\, t \cdot b^\top y + p_w(y)$$



Equivalent Optimality Criteria

- Primal dual feasible points
  - $(x_w, s_w, y_w) \in \mathbb{R}^n_{\geq 0} \times \mathbb{R}^n_{\geq 0} \times \mathbb{R}^d$
  - $A^\top x_w = b$ and $Ay + s = c$ $(Ay \geq c)$
- Optimality condition
  - $x_i s_i = w_i / t$ for all $i \in [n]$

Idea

- Increase t for bounded w

Algorithms

Primal

$$\min_{x\geq 0:\, A^\top x = b} c^\top x$$

Dual

$$\max_{y:\, Ay\geq c} b^\top y$$

[R84] Log Barrier ($\tilde{O}(\sqrt{n})$-iteration)

- $w = \vec{1}_m$
- Dikin ellipse is $\tilde{O}(n)$ rounding

[LS14,19] Lewis Weight Barrier

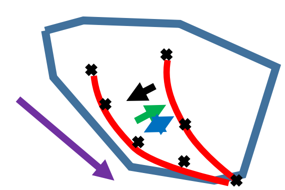
- $\tilde{O}(\sqrt{d})$-iteration)
- $w \approx \sigma_p(S^{-1/2}X^{1/2}A)$ for $p = \tilde{O}(1)$
- $\sigma_p(A)$ are ℓ_p Lewis Weights
  - (relative row importance in ℓ_p)
- Dikin ellipse is $\tilde{O}(d)$ rounding

Equivalent Optimality Criteria

- Primal dual feasible points
  - $(x_w, s_w, y_w) \in \mathbb{R}^n_{\geq 0} \times \mathbb{R}^n_{\geq 0} \times \mathbb{R}^d$
  - $A^\top x_w = b$ and $Ay + s = c$ $(Ay \geq c)$
- Optimality condition
  - $x_i s_i = w_i / t$ for all $i \in [n]$

Idea

- Increase t for bounded w



Algorithms

Primal
$$\min_{x \geq 0:\, A^\top x = b} c^\top x$$

Dual
$$\max_{y:\, Ay \geq c} b^\top y$$

[R84] Log Barrier ($\tilde{O}(\sqrt{n})$-iteration)

- $w = \vec{1}_m$
- Dikin ellipse is $\tilde{O}(n)$ rounding

[LS14,19] Lewis Weight Barrier

- $\tilde{O}(\sqrt{d})$-iteration
- $w \approx \sigma_p(S^{-1/2}X^{1/2}A)$ for $p = \tilde{O}(1)$
- $\sigma_p(A)$ are ℓ_p Lewis Weights
  - (relative row importance in ℓ_p)
- Dikin ellipse is $\tilde{O}(d)$ rounding

"Robust" Path [CLS19,B19*] ($\tilde{O}(n^\omega)$-time)

- Potential crudely penalizing $x_i s_i \neq w_i/t$
- Newton step in direction of gradient of potential to decrease
- Maintain x, s approximately!

``Robust LS'' [BL**S**S19, BLNP**S**SW20]

- $w \approx \sigma_2(S^{-1/2+\alpha}X^{1/2+\alpha})$ for $\alpha = 1/\tilde{O}(1)$
- $\sigma_2(A)$ = leverage scores. Can approximate with linear system solves!
- Similar path, different approximation!

Algorithms

Primal
$\min_{x \geq 0:\, A^\top x = b} c^\top x$

Dual
$\max_{y:\, Ay \geq c} b^\top y$

Our Iteration *(roughly)*

- Approximately feasible (x_t, s_t, y_t)
- Approximations $\bar{x}_t \approx x_t$ and $\bar{s}_t \approx s_t$
- Improvement direction d_t induced by σ_t
- Approximate Hessian $\bar{H}_t \approx A^\top \bar{X}_t \bar{S}_t^{-1} A$
- Approximate Newton step
  - $x_{t+1} \approx x_t + \eta_x (I - \overline{X}_t \overline{S}_t^{-1} A H_t^{-1} A^\top) \overline{X}_t d_t$
  - $s_{t+1} \approx s_t + \eta_s A \overline{H}_t^{-1} A^\top \overline{X}_t d$
- Issue:
  - $\bar{H} \approx A^\top \bar{X} \bar{S}^{-1} A$ loses feasibility
  - [BLSS19, BLNPSSW20] address differently

"Robust" Path [CLS19,B19*] ($\tilde{O}(n^\omega)$-time)

- Potential crudely penalizing $x_i s_i \neq w_i / t$
- Newton step in direction of gradient of potential to decrease
- Maintain x, s approximately!

``Robust LS'' [BL**S**S19, BLNP**S**SW20]

- $w \approx \sigma_2(S^{-1/2+\alpha} X^{1/2+\alpha})$ for $\alpha = 1/\tilde{O}(1)$
- $\sigma_2(A)$ = leverage scores. Can approximate with linear system solves!
- Similar path, different approximation!

[BLSS19]
Fairly short proof of primal-dual $\tilde{O}(\sqrt{d})$ iteration algorithm.

[BLNPSSW21]
Simplifies handling of feasibility issues.

Primal
$$\min_{x \geq 0:\, A^\top x = b} c^\top x$$

Dual
$$\max_{y:\, Ay \geq c} b^\top y$$

[BLLSSSW21] More general constraints and new data structures.

Our Iteration *(roughly)*

- Approximately feasible (x_t, s_t, y_t)
- Approximations $\bar{x}_t \approx x_t$ and $\bar{s}_t \approx s_t$
- Improvement direction d_t induced by σ_t
- Approximate Hessian $\bar{H}_t \approx A^\top \bar{X}_t \bar{S}_t^{-1} A$
- Approximate Newton step
  - $x_{t+1} \approx x_t + \eta_x (I - \overline{X}_t \overline{S}_t^{-1} A H_t^{-1} A^\top) \overline{X}_t d_t$
  - $s_{t+1} \approx s_t + \eta_s A \overline{H}_t^{-1} A^\top \overline{X}_t d$
- Issue:
  - $\bar{H} \approx A^\top \bar{X} \bar{S}^{-1} A$ loses feasibility
  - [BLSS19, BLNPSSW20] address differently

Approach

- This a $\tilde{O}(\sqrt{d})$ iteration method!
- Goal: $\tilde{O}(nd + \mathrm{poly}(d))$ runtime
- Need to implement steps in $o(nd)$ on average!
- Idea: leverage structure of the IPM and the flexibility of approximating to design a fast method.

Note*: will hide many details throughout the talk and simplify many parts. Happy to discuss details after.*

This Talk

Part 1: Overview
- Survey recent history of interior point methods for linear programming
- Present and motivate recent nearly linear time algorithms

Part 2: IPM
- Brief intro to recent IPM advances and our new IPMs

Part 3: Data Structures
- Data structures for efficiently implementing our IPMs
- Highlight difficulty and key techniques

… and why is each paper ≥100 pages?

How to implement?

Primal
$$\min_{x \geq 0:\, A^\top x = b} c^\top x$$

Dual
$$\max_{y:\, Ay \geq c} b^\top y$$

Initial point

Our Iteration *(roughly)*

- Approximately feasible (x_t, s_t, y_t)
- Approximations $\bar{x}_t \approx x_t$ and $\bar{s}_t \approx s_t$
- Improvement direction d_t induced by σ_t
- Approximate Hessian $\overline{H}_t \approx A^\top \overline{X}_t \overline{S}_t^{-1} A$
- Newton step
  - $x_{t+1} \approx x_t + \eta_x (I - \overline{X}_t \overline{S}_t^{-1} A H_t^{-1} A^\top) \overline{X}_t d_t$
  - $s_{t+1} \approx s_t + \eta_s A \overline{H}_t^{-1} A^\top \overline{X}_t d_t$
- Issue:
  - $\overline{H} \approx A^\top \overline{X}_t \overline{S}_t^{-1} A$ loses feasibility

[BLSS19]

[BLNPSSW20]

- Vector maintenance data structure
- Leverage score maintenance data structure
- Gradient maintenance data structure
- Inverse maintenance data structure
- Feasibility maintenance analysis
- Sampling

… and why is each paper ≥100 pages?

How to implement?

Primal
$$\min_{x\ge 0:\, A^\top x = b} c^\top x$$

Dual
$$\max_{y:\, Ay\ge c} b^\top y$$

Initial point

Our Iteration *(roughly)*

- Approximately feasible (x_t, s_t, y_t)
- Approximations $\bar{x}_t \approx x_t$ and $\bar{s}_t \approx s_t$
- Improvement direction d_t induced by σ_t
- Approximate Hessian $\overline{H}_t \approx A^\top \overline{X}_t \overline{S}_t^{-1} A$
- Newton step
  - $x_{t+1} \approx x_t + \eta_x (I - \overline{X}_t \overline{S}_t^{-1} A H_t^{-1} A^\top) \overline{X}_t d_t$
  - $s_{t+1} \approx s_t + \eta_s A \overline{H}_t^{-1} A^\top \overline{X}_t d_t$
- Issue:
  - $\overline{H} \approx A^\top \overline{X}_t \overline{S}_t^{-1} A$ loses feasibility

Vector maintenance data structure

Leverage score maintenance data structure

Gradient maintenance data structure

Inverse maintenance data structure

[BLSS19]

Feasibility maintenance analysis

Sampling

[BLNPSSW20]

Difficulty in achieving $\tilde{O}(nd + poly(d))$

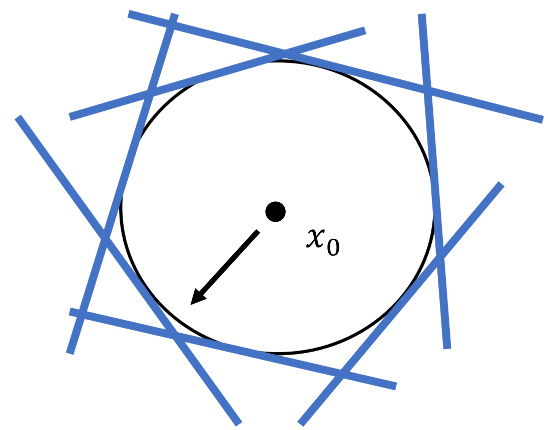
Interior Point Method

- $\tilde{O}(\sqrt{d})$ iteration method
- Goal: maintain $\bar{s}_t \approx s_t = \boldsymbol{A}y_t - c \in \mathbb{R}^n_{\geq 0}$ $(Ay_t \geq c)$
- Need know when $\boldsymbol{A}y_t$ closer to c
- $\tilde{O}(n\sqrt{d} + \text{poly}(d))$ on average per iteration

In contrast to problems like regression where
"easier to obtain $\tilde{O}(nd + \text{poly}(d))$ methods.

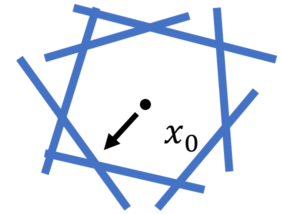
Problem

- Unknown how to efficiently compress $\{y \mid Ay \geq b\}$
- Unknown how to sample or build data structure for arbitrary changes to y
- Hard case:



Why Any Hope?

study slacks for simplicity



Setup (*simplified*)

- $s_{t+1} \stackrel{\text{def}}{=} s_t + \boldsymbol{A} d_t$
- $s_{t+1} \geq 0$

Goal

- Maintain $\bar{s} \approx s_t$

Problem

- What if this change is arbitrary?

Classic Observation

- In most IPMs changes are sparse on average (enables inverse maintenance)
- Renegar: $\|\boldsymbol{S}_t^{-1}(s_{t+1} - s_t)\|_2 = O(1)$
- Ours: $\|\boldsymbol{S}_t^{-1}(s_{t+1} - s_t)\|_{\approx\sigma_t} = O(1)$
  - $\|\boldsymbol{S}_t^{-1}(s_{t+1} - s_t)\|_2 = O\left(\sqrt{n/d}\right)$
  - Every $\tilde{O}(\sqrt{d})$ iterations $\tilde{O}(n)$ coordinates change by constant

Goal: detect in $\tilde{O}(d)$ time on average!

Vector Maintenance

simplified version of requisite data structure problem

Data Structure Goal

- Maintain $\overline{s}_t \approx s_t \geq 0$
- $s_t \stackrel{\text{def}}{=} s_0 + \sum_{i \in [t]} \boldsymbol{A} d_i$

Result

- Can maintain after T step in $\tilde{O}(nd + T(n + d \sum_{i \in [T]} \|\boldsymbol{S}_i^{-1} \boldsymbol{A} d_i\|_2^2)$
- Is $\tilde{O}(nd)$ when $T = O(\sqrt{d})$

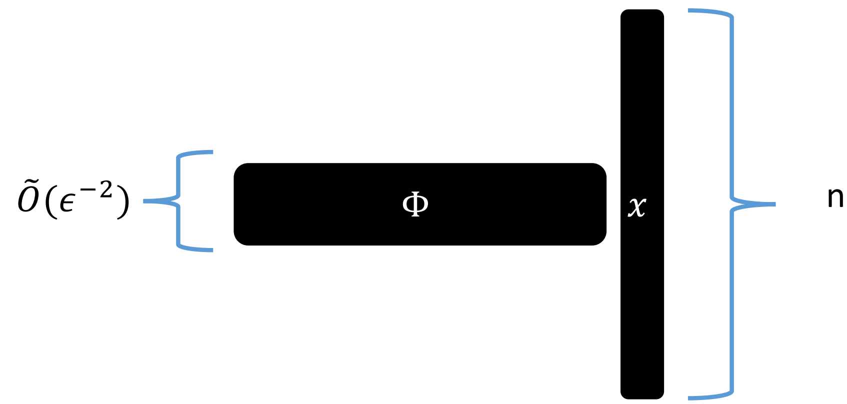
Idea: Warmup
Detect when single coordinate changes by a constant in one step.

very well-studied problem

How Compute Large Entries of a Vector?

ℓ_∞/ℓ_2-Heavy Hitters Sketch Theorem

$\forall \epsilon, n > 0$ can build $\mathbf{\Phi} \in \mathbb{R}^{\tilde{O}(\epsilon^{-2}) \times n}$ with $\tilde{O}(1)$ non-zeros per column where from $\mathbf{\Phi}x$ can get in $\tilde{O}(\epsilon^{-2})$ all $i \in [n]$ with $|x_i| \geq \epsilon \cdot \|x\|_2$.



[Charikar, Chen, Farach-Colton'04, Cormode, Muthukrishnan'05, Cormode, Hadjieleftheriou'08, Kane, Nelson, Porat, Woodruff'11, Pagh'13, Nelson, Nguyen, Woodruff'12, Indyk, Kapralov'14, Larsen, Nelson, Nguyen, Thorup'16, Nakos, Song, Wang'19]

Vector Maintenance

Serious recurring issue in using data structures for optimization.

Data Structure Goal

- Maintain $\overline{s}_t \approx s_t \geq 0$
- $s_t \stackrel{\text{def}}{=} s_0 + \sum_{i \in [t]} \boldsymbol{A} d_i$

Result

- Can maintain after T step in

$$\tilde{O}(nd + T(n + d \sum_{i \in [T]} \|\boldsymbol{S}_i^{-1} \boldsymbol{A} d_i\|_2^2)$$

- Is $\tilde{O}(nd)$ when $T = O(\sqrt{d})$

Problem!

Oblivious Adversary Issue: Φ only works if x independent of randomness in Φ!

Solution: only output if true change is large! Output independent of randomness! (sketch just saves time!)

Key Tool: $\ell_\infty - \ell_2$ Heavy Hitters

- $\forall \epsilon, n > 0$ can build $\boldsymbol{\Phi} \in \mathbb{R}^{\tilde{O}(\epsilon^{-2}) \times n}$ with $\tilde{O}(1)$ non-zeros per column where from $\boldsymbol{\Phi} x$ can get in $\tilde{O}(\epsilon^{-2})$ all $i \in [n]$ with $|x_i| \geq \epsilon \cdot \|x\|_2$.

Idea

- Precompute $\boldsymbol{\Phi}_\epsilon \boldsymbol{S}_t^{-1} \boldsymbol{A}$ for different all $\epsilon = 2^i$
- If no-multiplicative change so far

$$\left\|\sum_{i \in [t]} \boldsymbol{S}_i^{-1} \boldsymbol{A} d_i\right\|_2^2 \leq O\left(t \sum_{i \in [t]} \left\|\boldsymbol{S}_i^{-1} \boldsymbol{A} d_i\right\|_2^2\right) \stackrel{\text{def}}{=} \Theta(\epsilon_t^{-2})$$

- Checking $\boldsymbol{\Phi}_{\epsilon_t} \boldsymbol{S}_t^{-1} \boldsymbol{A} \sum_{i \in [t]} d_i$ gives all multiplicative change within time budget!
- Apply for every power of 2, check if row change every step, every 2 steps, every 4, etc. and update S_t as changes

Graph Vector Maintenance

Data Structure Goal

- Maintain $\overline{s}_t \approx s_t \geq 0$
- $s_t \stackrel{\text{def}}{=} s_0 + \sum_{i \in [t]} \boldsymbol{A} d_i$
- A is incidence matrix of n-node m-edge graph
- $[Ad_i]_e = [d_i]_j - [d_i]_k$ for $e = (j, k)$

Result

- Can maintain after T step in

$$\tilde{O}(m + T(n + \sum_{i \in [T]} \|\boldsymbol{S}_i^{-1} \boldsymbol{A} d_i\|_2^2))$$

- Is $\tilde{O}(m + n^{1.5})$ when $T = O(\sqrt{n})$

Key Tool #1: Dynamic Expander Decomposition

- [N17,Wul17,NSW17,W19,CCGLLNPS20,BBGNS**S**S20]
- Can maintain decomposition of graph into expanders (well-connected subgraphs) with $\tilde{O}(n)$ total vertices in $\tilde{O}(1)$ per edge insertion / deletion

Key Tool #2 : Cheeger's Inequality

- Expanders $\approx$ diagonal matrices
- If large multiplicative change, then large d_i relative to degree on that vertex.
- Apply for every power of 2, checking if change every step, every 2 steps, every 4, etc.

Hiding many details

Remaining Data Structures

Upshot

Sketching the Central Path: can make repeated use of sketches to save iteration cost

Leverage Score Maintenance

- JL + vector maintenance
- Matrix generalization

Inverse Maintenance

- Sparsification and low-rank updates [V89,L**S**15,CLS19,B19]
- Leverage score sampling [L**S**15]
- Randomness hiding [L**S**15]

Upshot

Open the door to more data structure use.

Feasibility Maintenance

- Make iterates correct in expectation
- Take some steps to help feasibility

Gradient Maintenance

- Leverage structure of steepest descent steps on Φ
- Leverage discrete structure of approximate iterates

"Simpler" IPM in later results.
Requires "sampling" data structures.

Note*: will hide many details throughout the talk and simplify many parts. Happy to discuss details after.*

This Talk

Part 1: Overview

- Survey recent history of interior point methods for linear programming
- Present and motivate recent nearly linear time algorithms

Part 2: IPM

- Brief intro to recent IPM advances and our new IPMs

Part 3: Data Structures

- Data structures for efficiently implementing our IPMs
- Highlight difficulty and key techniques

Minimum Cost Flows, MDPs, and ℓ_1-Regression in Nearly Linear Time for Dense Instances

Further improvements and applications
Using additional techniques.

ℓ_1-Regression

- $\min_{x \in \mathbb{R}^d} \|\boldsymbol{A}x - b\|_1$ for $\boldsymbol{A} \in \mathbb{R}^{n \times d}$
- $\tilde{O}(nd + d^{2.5})$

Markov Decision Process

- S-states A-actions
- $\tilde{O}(S^2 A + S^{2.5})$

Maximum Flow

- m-edges, n-nodes, integer capacities at most U
- $\tilde{O}(m + n^{1.5})$

Thank you

Questions?

Aaron Sidford

Contact Info:

- email: *sidford@stanford.edu*
- website: *www.aaronsidford.com*