How to Solve Problems on Graphs Using Linear Equations, and How to Solve Linear Equations Using Graphs

Speaker **Rasmus Kyng** Harvard
Theory of Computing

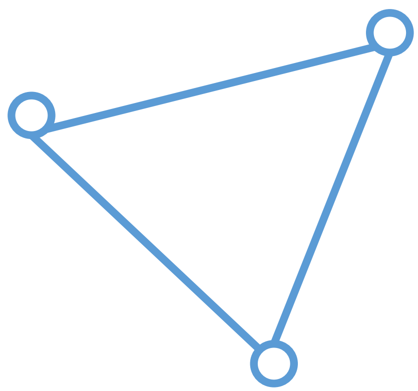
~~How to Solve Problems on Graphs~~
~~Using Linear Equations, and~~
~~How to Solve Linear Equations Using Graphs~~

Optimization on Graphs

Speaker **Rasmus Kyng** Harvard
Theory of Computing

Optimization on Graphs

Graph $G = (V, E)$



Optimization ~~on Graphs~~

An approach to solving problems

~~Optimization on Graphs~~

An approach to solving problems

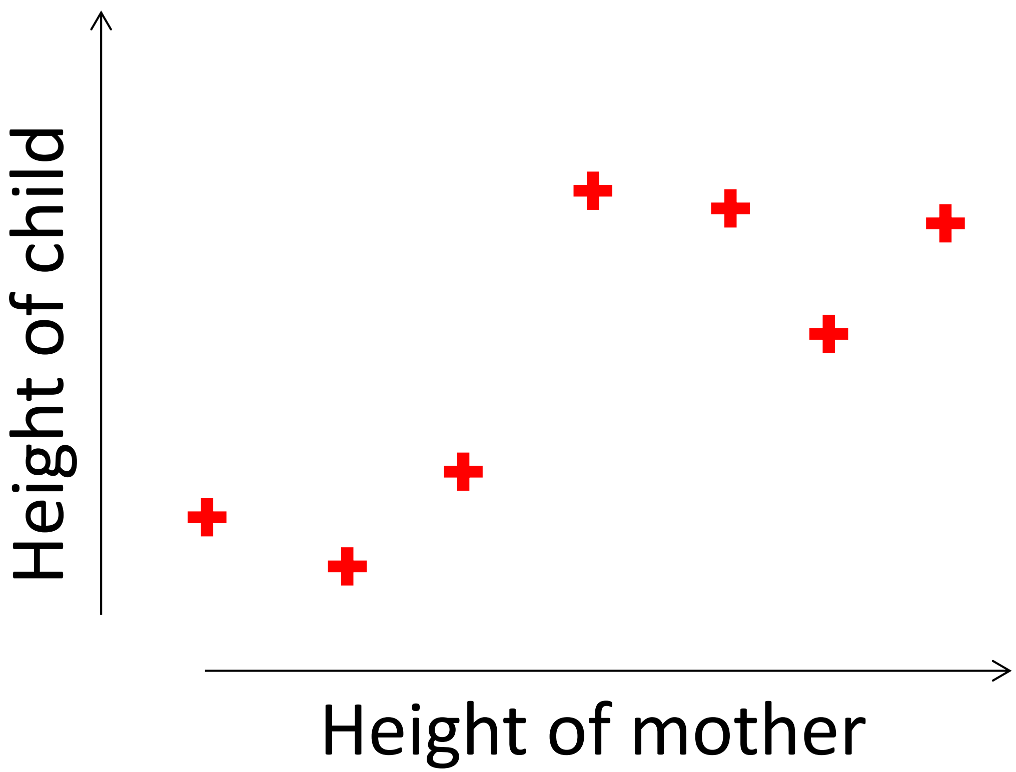
~~Optimization on Graphs~~

Let’s look at an example

Isotonic Regression

Predict child's height from mother's height

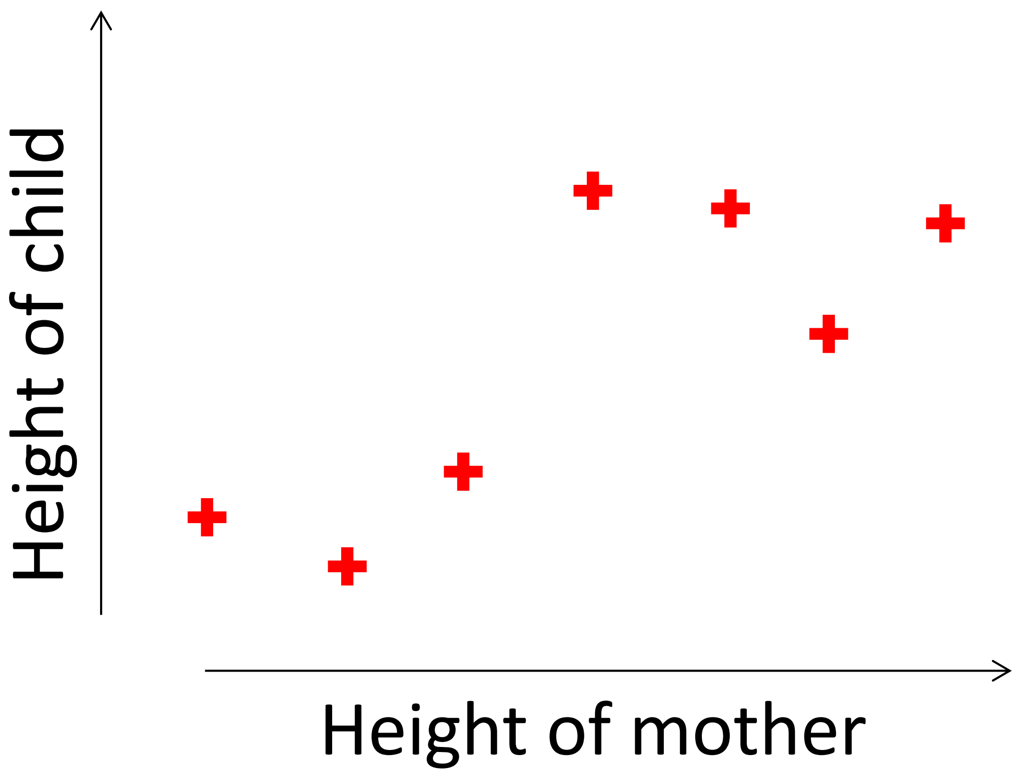
Model?



Isotonic Regression

Predict child's height from mother's height

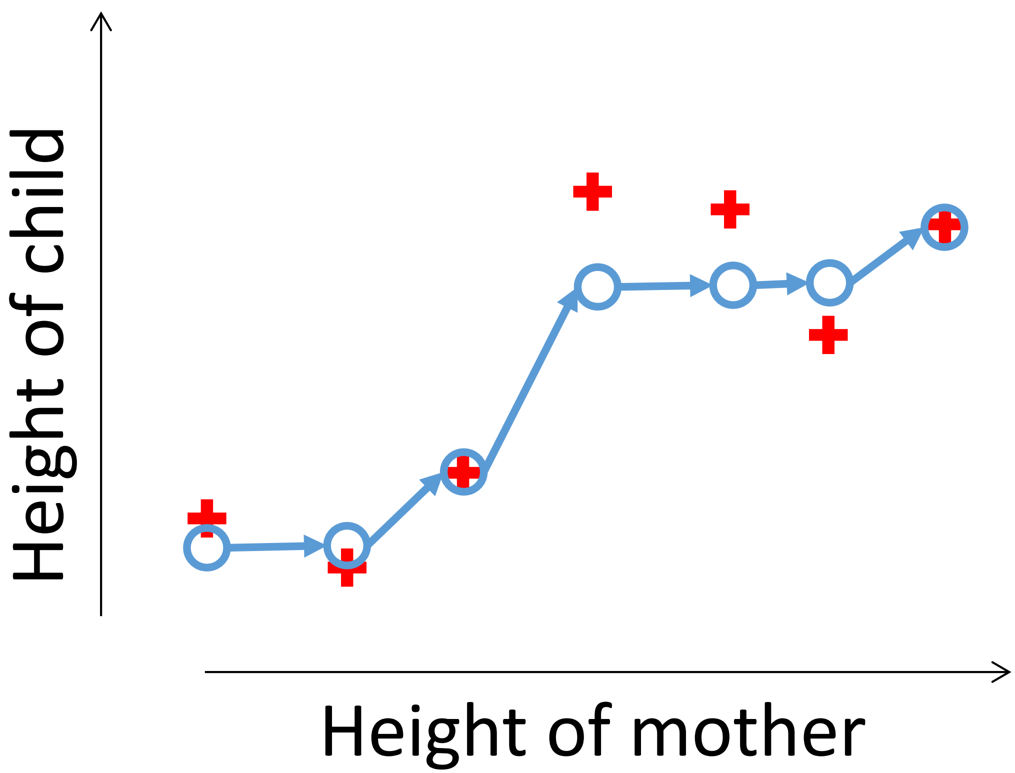
Model? Increasing function?



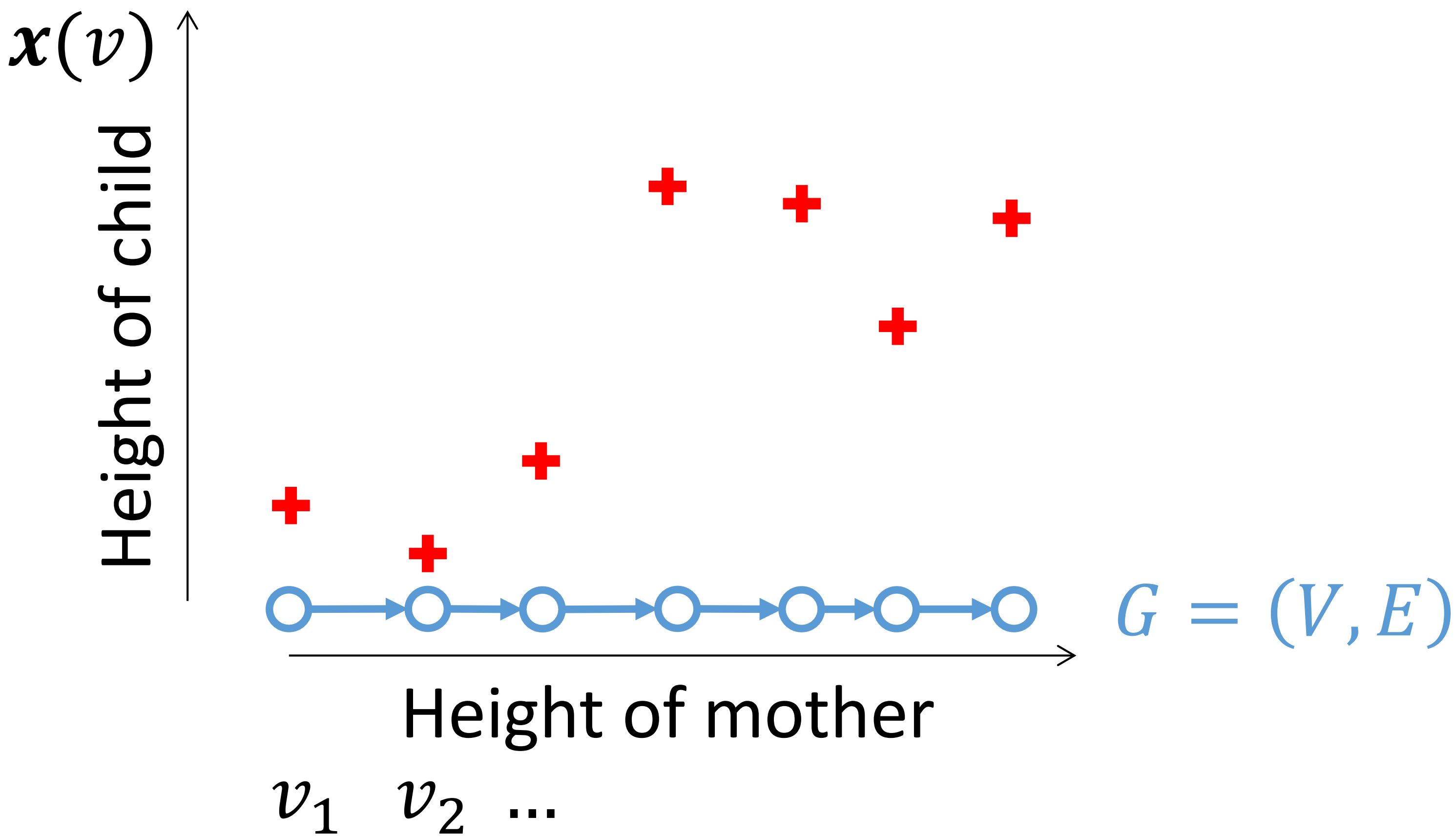
Isotonic Regression

Predict child's height from mother's height

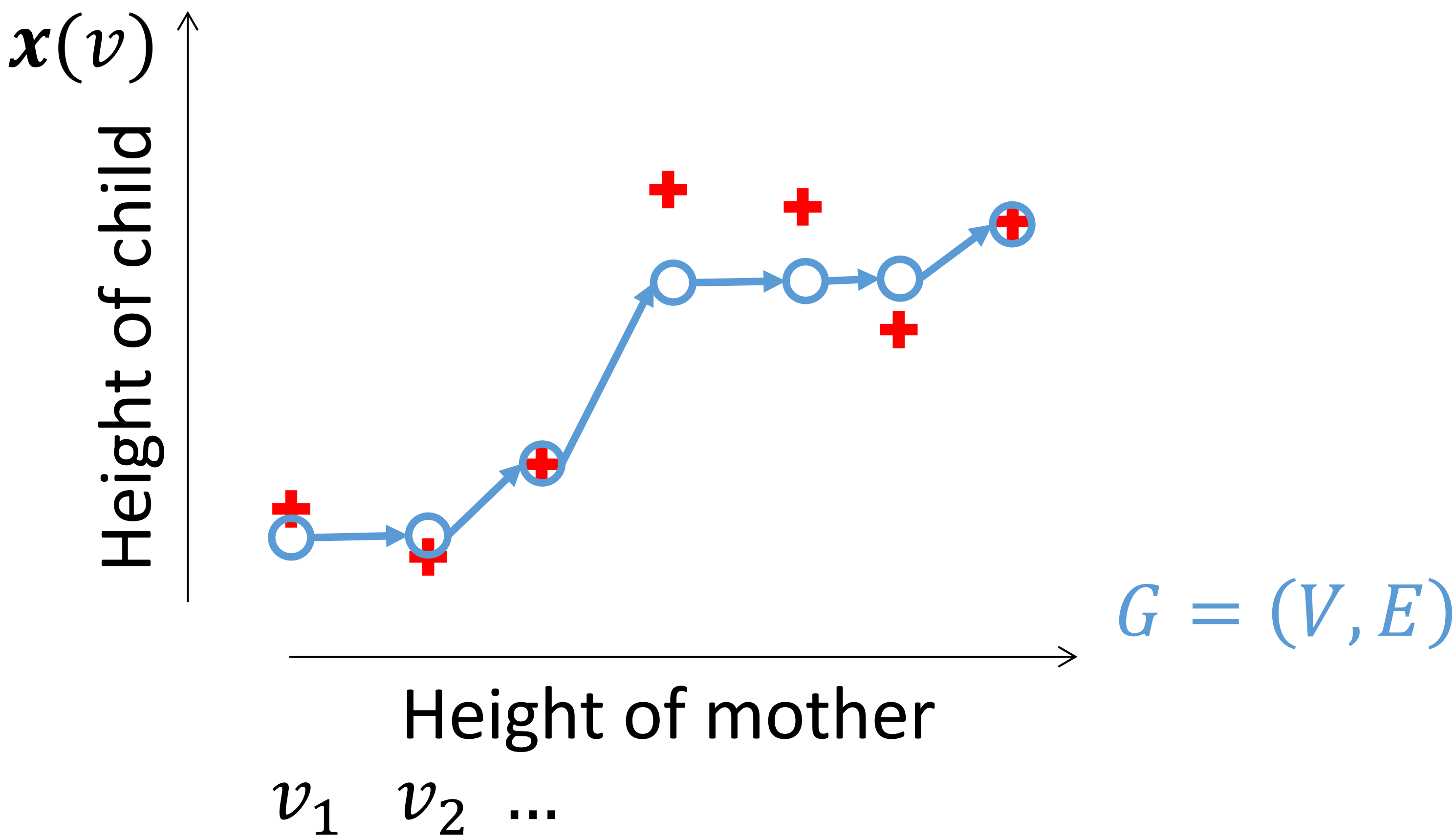
Model? Increasing function?



Isotonic Regression



Isotonic Regression

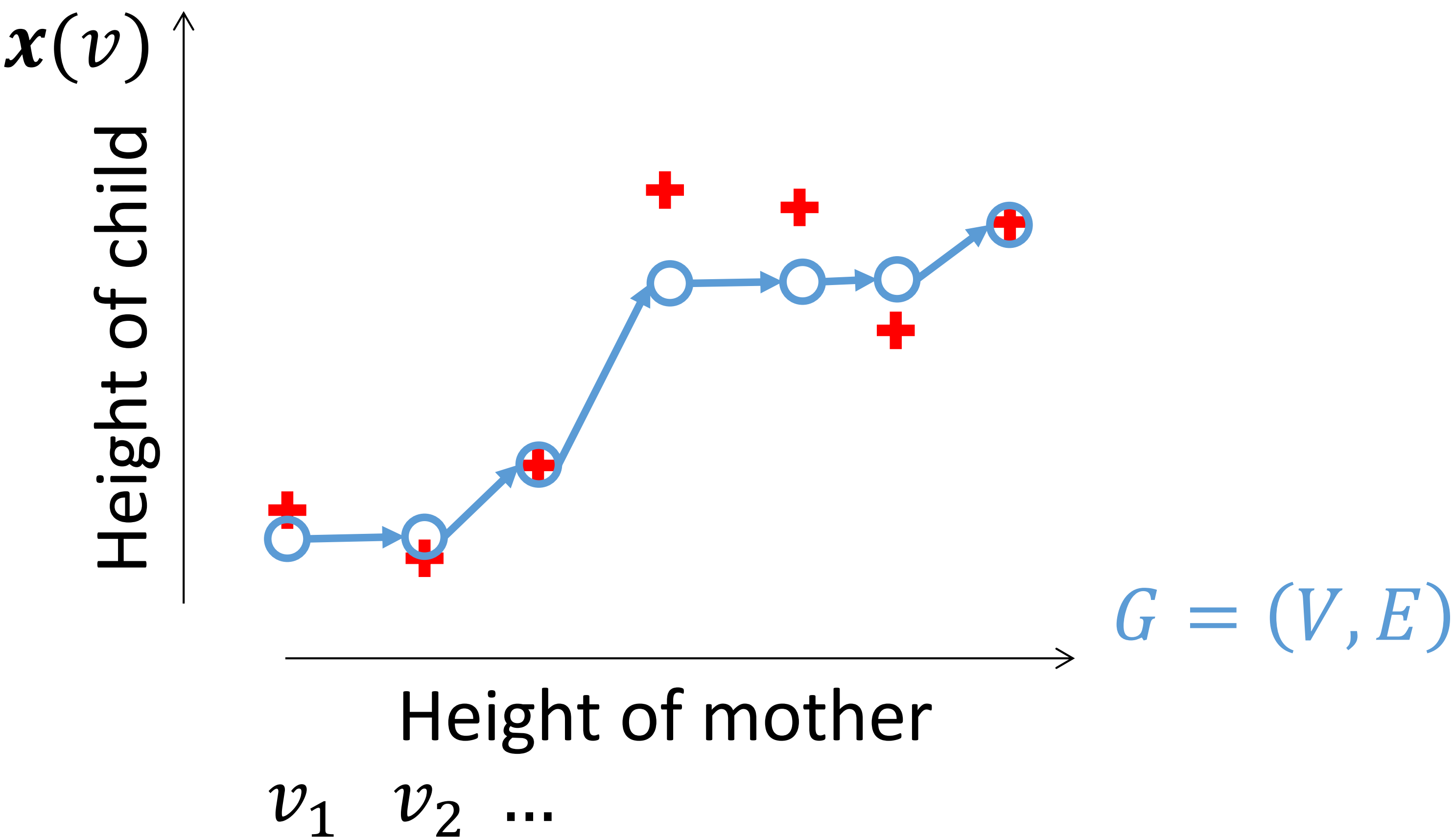


Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:

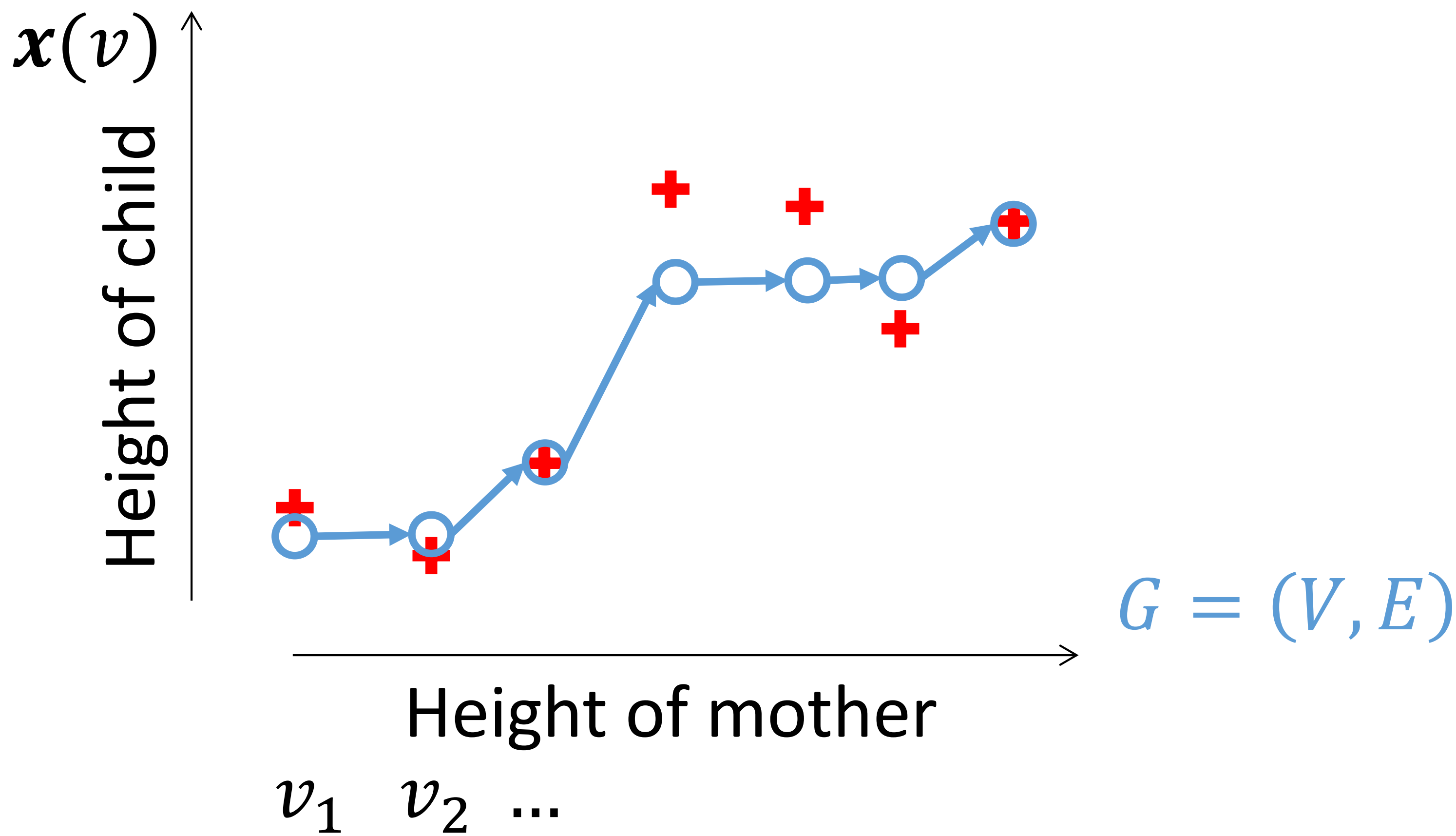
$$(\boldsymbol{x}(v) - \text{child\_height}(v))^2$$



Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:
$$\sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$

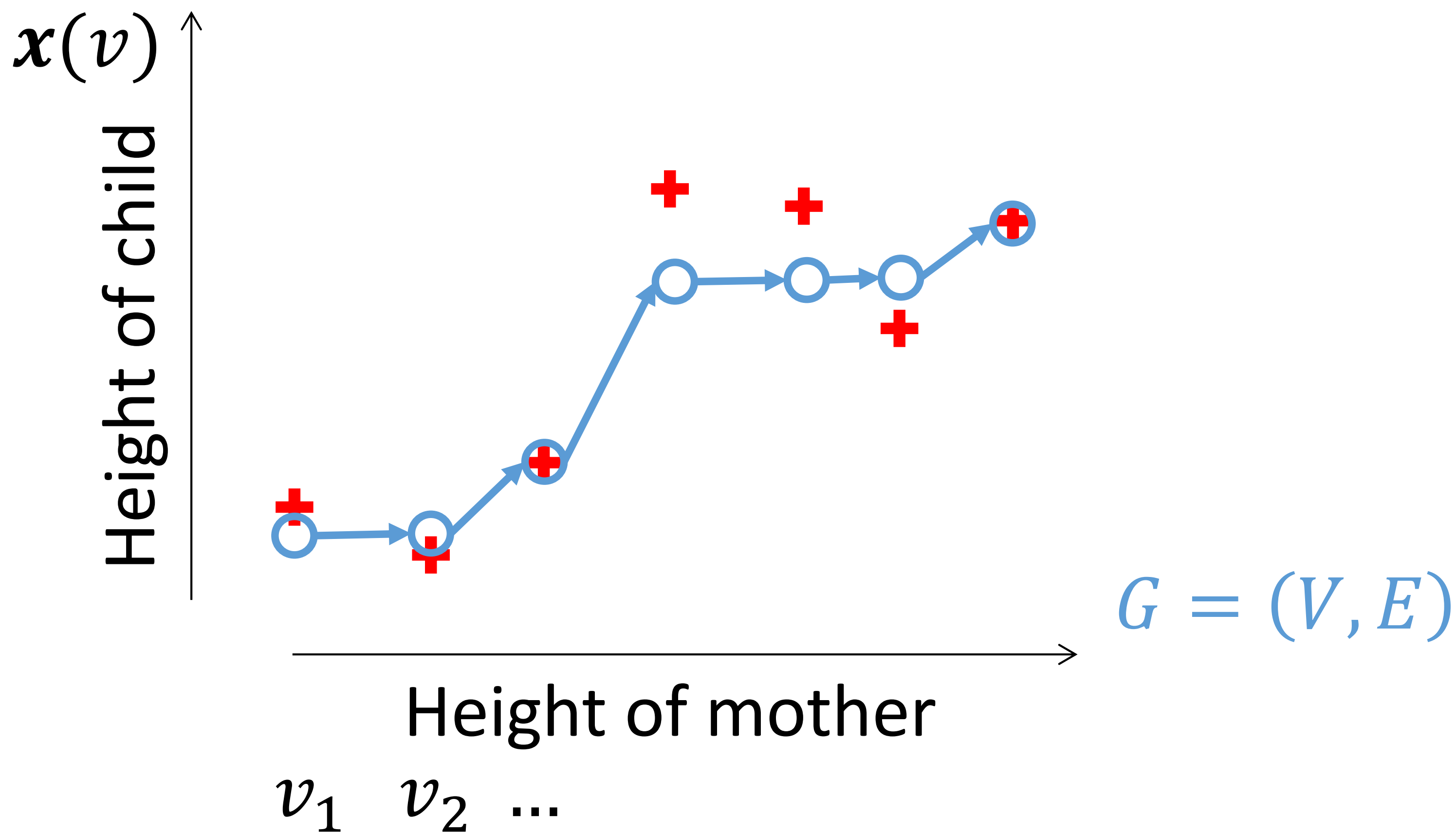


Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:

$$\min_{\boldsymbol{x}} \sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$

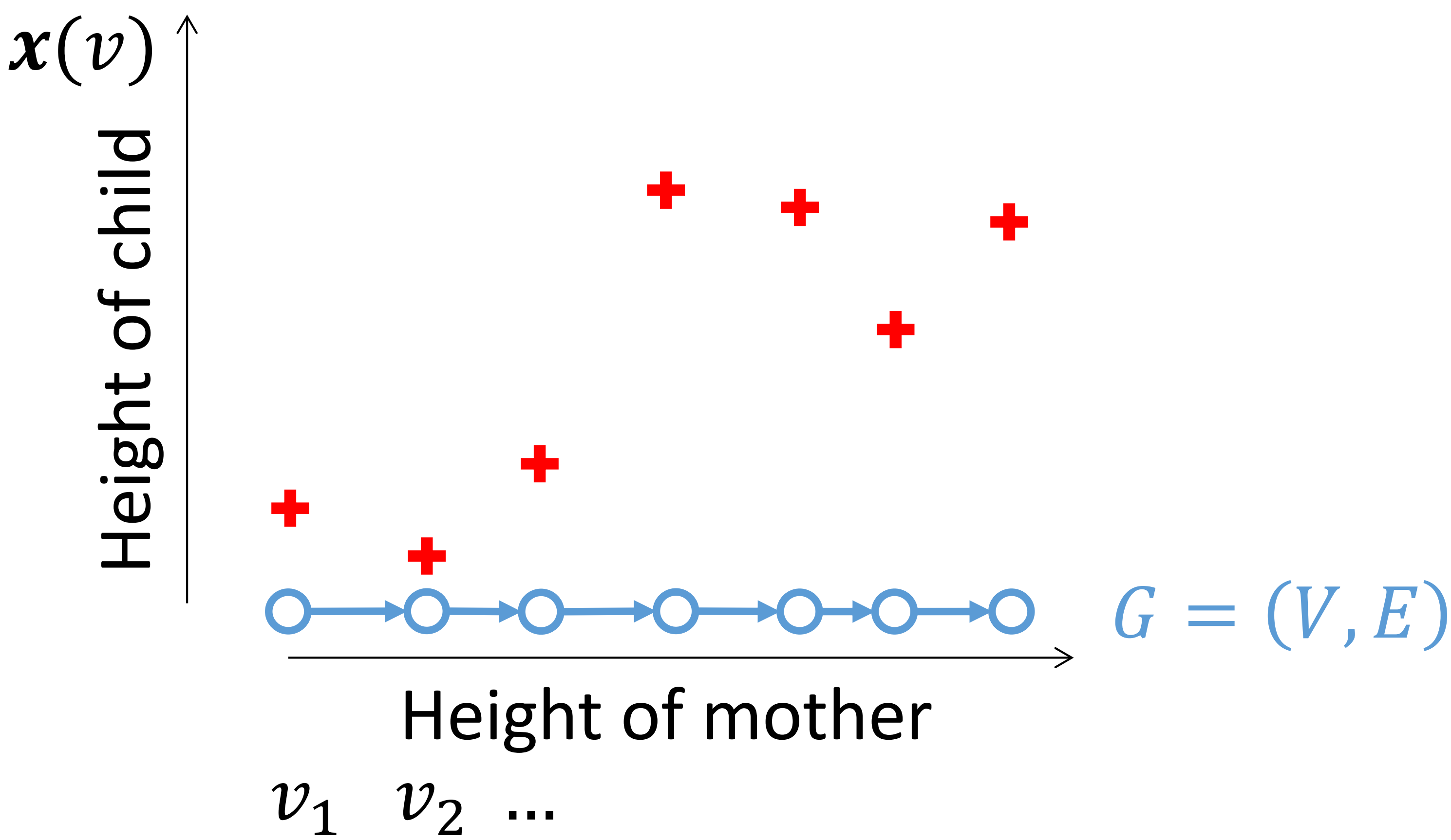


Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:

$$\min_{\boldsymbol{x}} \sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$

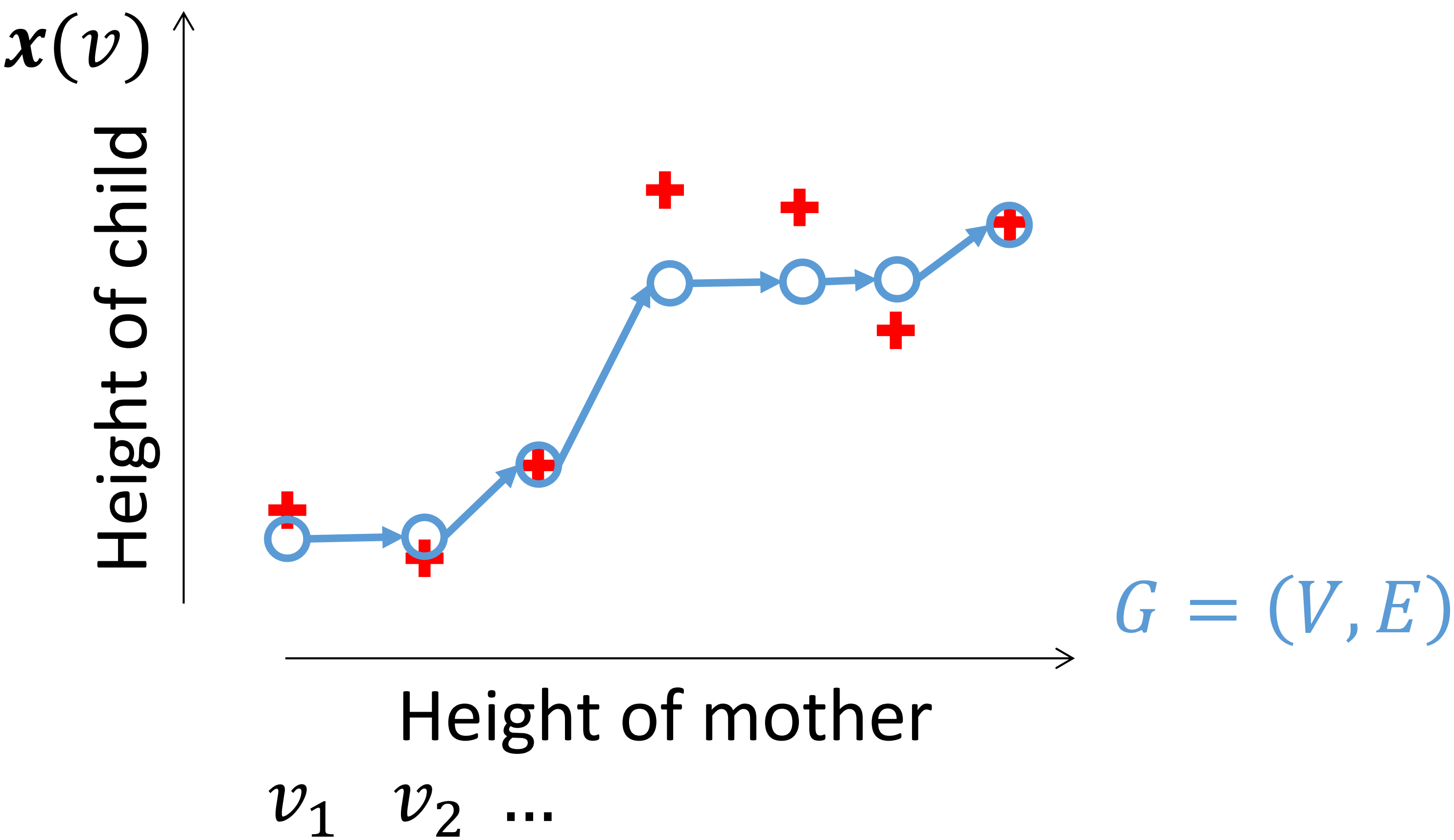


Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

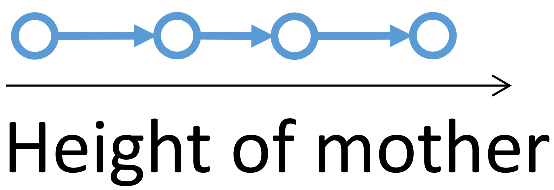
Cost:

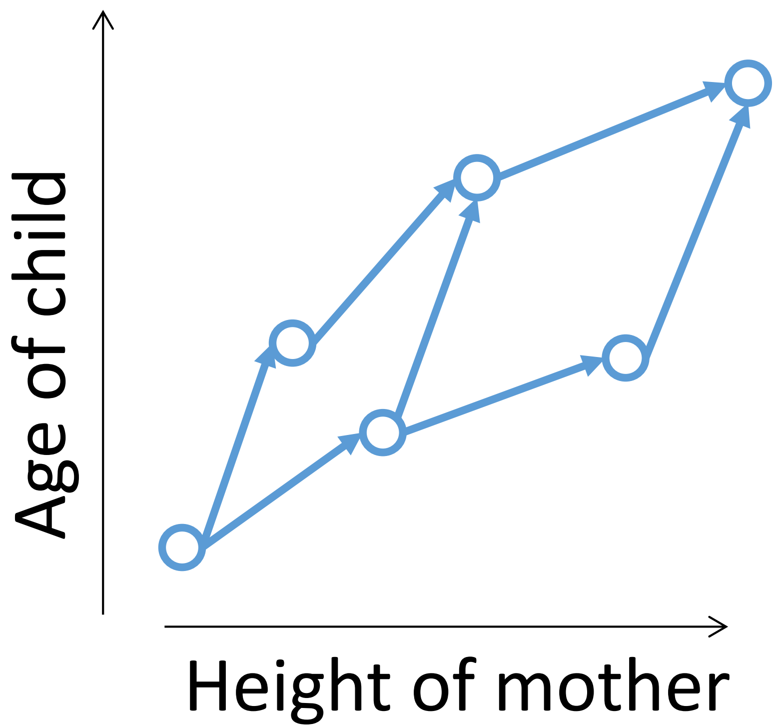
$$\min_{\boldsymbol{x}} \sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$



Isotonic Regression

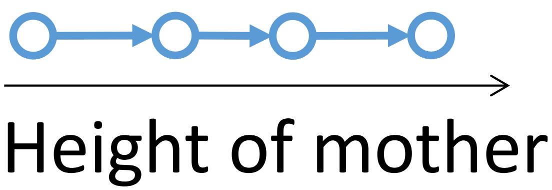
Constraint Graph $G = (V, E)$

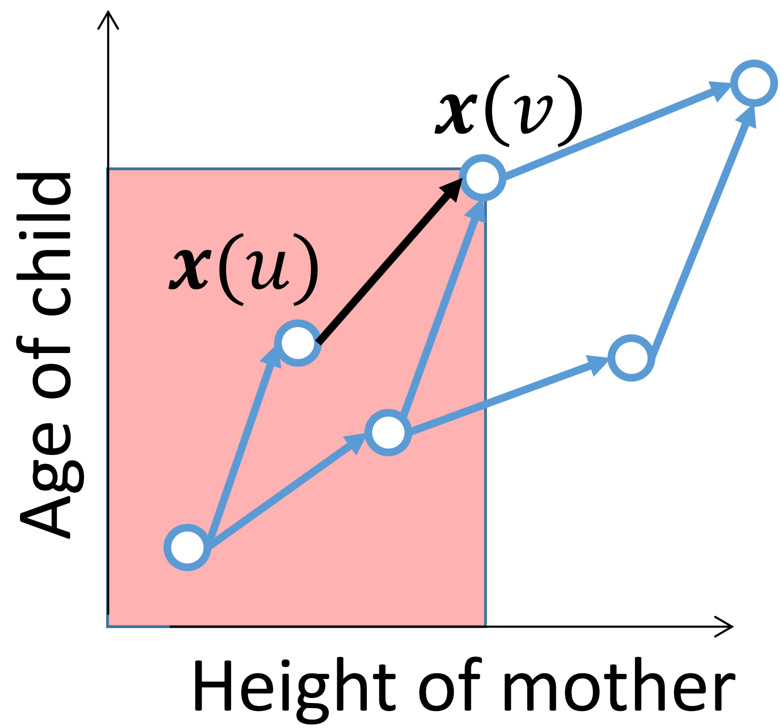




Isotonic Regression

Constraint Graph $G = (V, E)$





$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

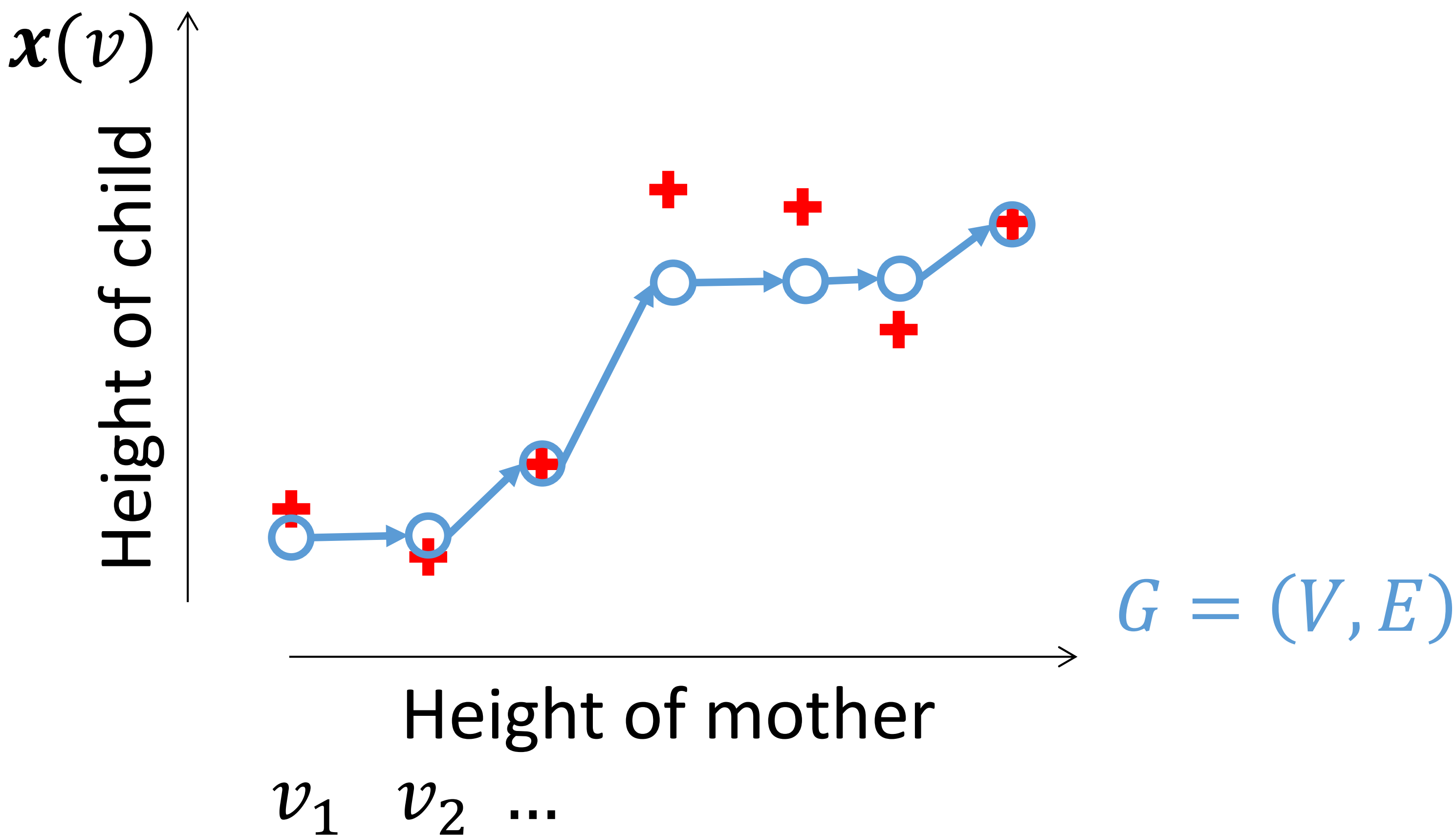
Taller mother AND older child

Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:

$$\min_{\boldsymbol{x}} \sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$



Problem Solving by Optimization

Problem understood through

1. solutions
2. costs of solutions

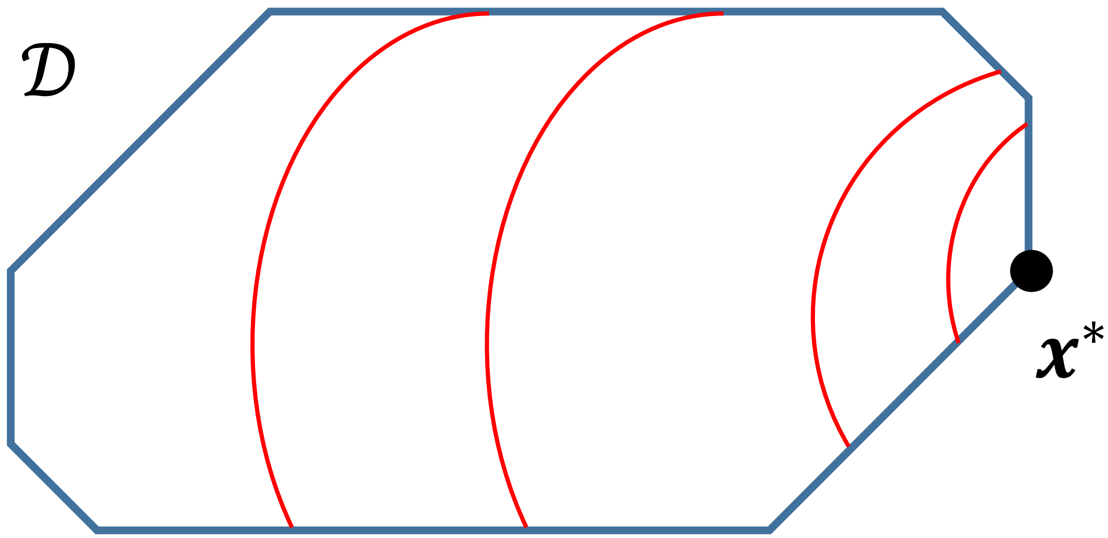
Cost function c : solution → cost

Goal:

$$\min_{\text{solutions } \boldsymbol{x}} \mathrm{c}(\boldsymbol{x})$$

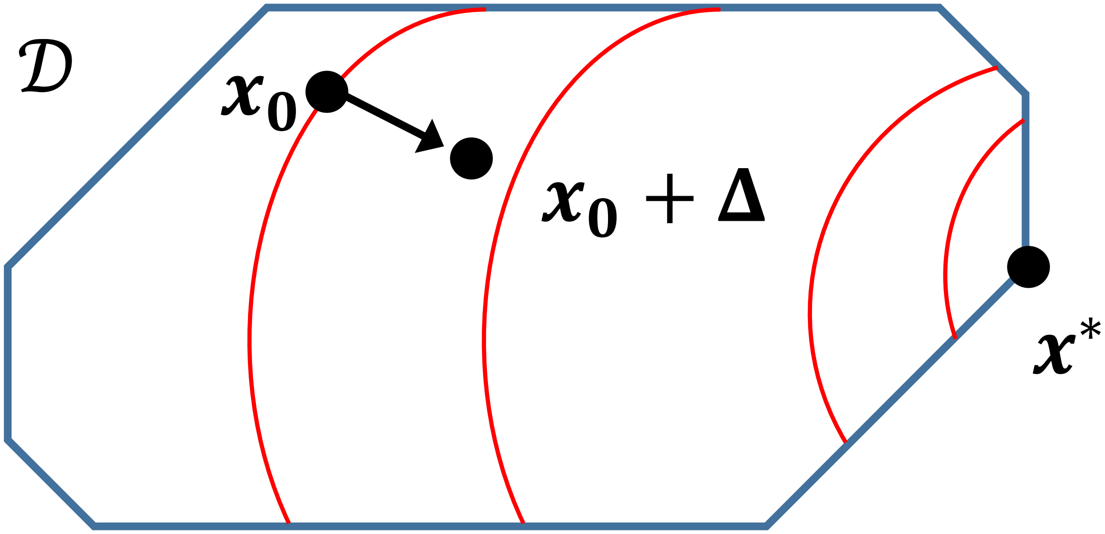
Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} \mathrm{c}(\boldsymbol{x})$$



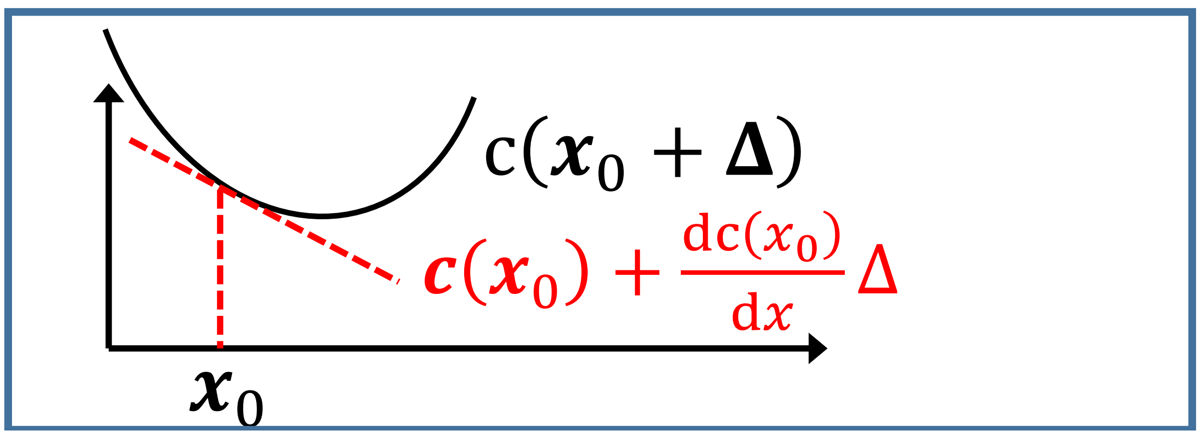
Optimization Primer

$$\min_{\boldsymbol{x}\in\mathcal{D}} \mathrm{c}(\boldsymbol{x})$$



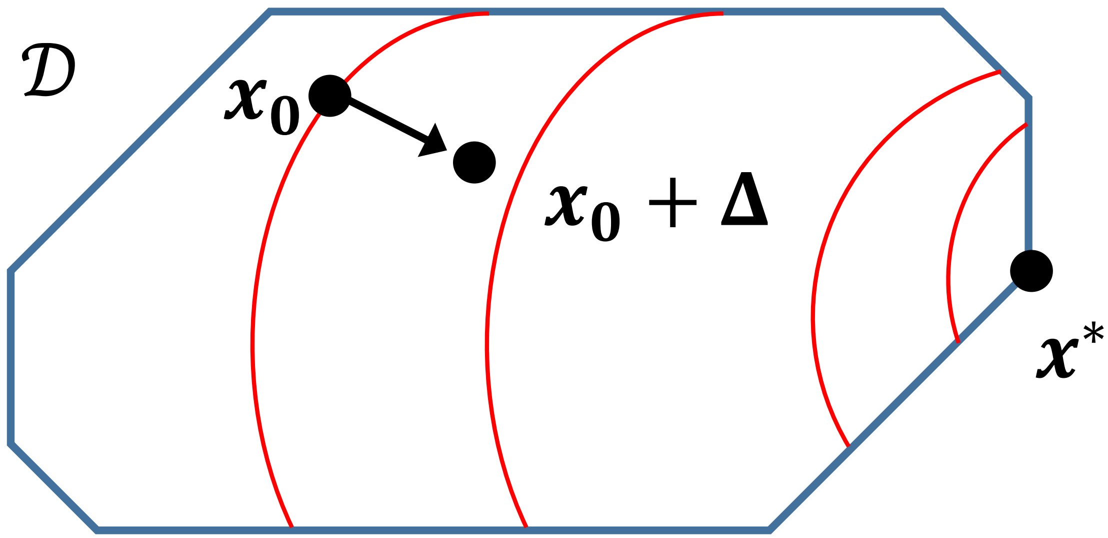
First Order Methods – "Gradient Descent"

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \frac{\mathrm{dc}(x_0)}{\mathrm{d}x}\Delta$$



Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} \mathrm{c}(\boldsymbol{x})$$

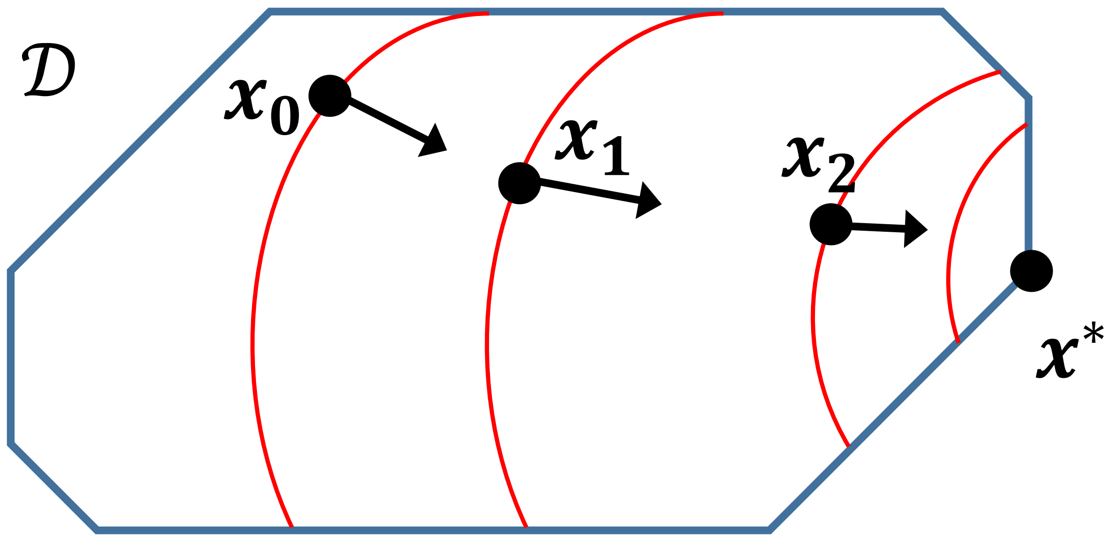


First Order Methods – "Gradient Descent"

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \sum_i \frac{\mathrm{dc}(x_0)}{\mathrm{d}x(i)} \Delta(\mathrm{i})$$

Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} \mathrm{c}(\boldsymbol{x})$$



First Order Methods – "Gradient Descent"

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \nabla \mathrm{c}(\boldsymbol{x}_0) \cdot \boldsymbol{\Delta}$$

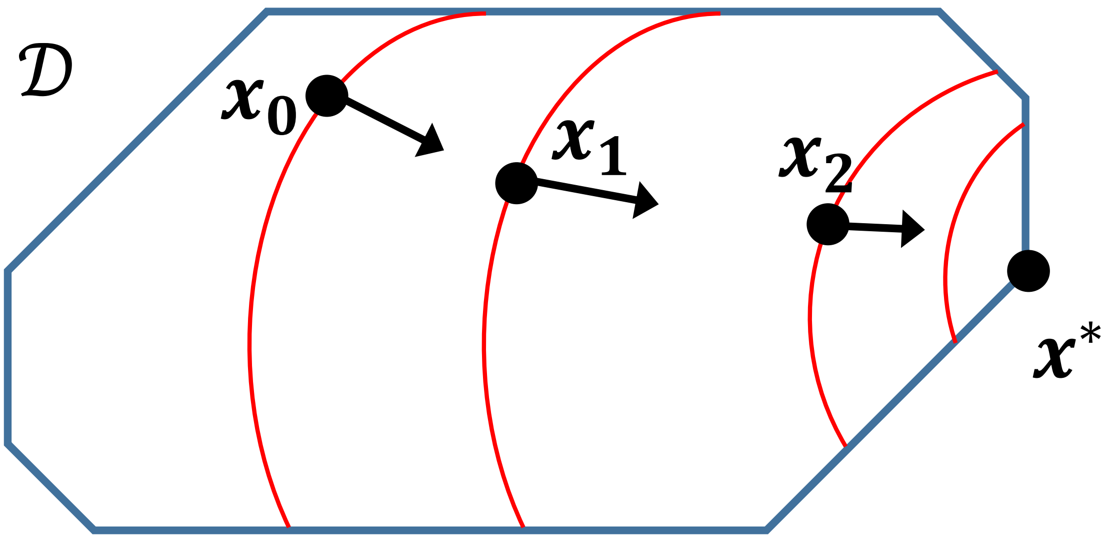
$$\boldsymbol{\Delta} = -0.1 \nabla \mathrm{c}(\boldsymbol{x}_0)$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) - 0.1 \|\nabla \mathrm{c}(\boldsymbol{x}_0)\|^2$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - 0.1 \nabla \mathrm{c}(\boldsymbol{x}_0)$$

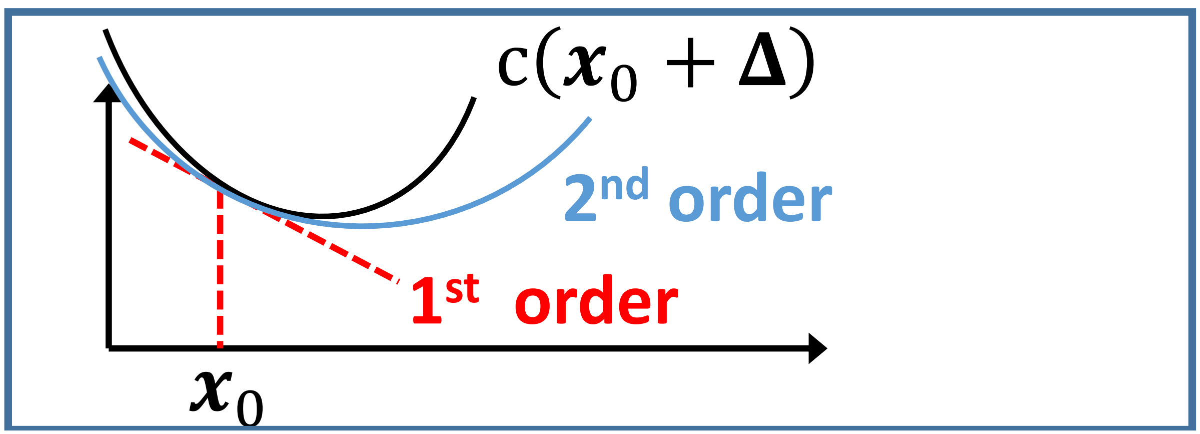
Optimization Primer

$$\min_{\mathbf{x} \in \mathcal{D}} \mathrm{c}(\mathbf{x})$$



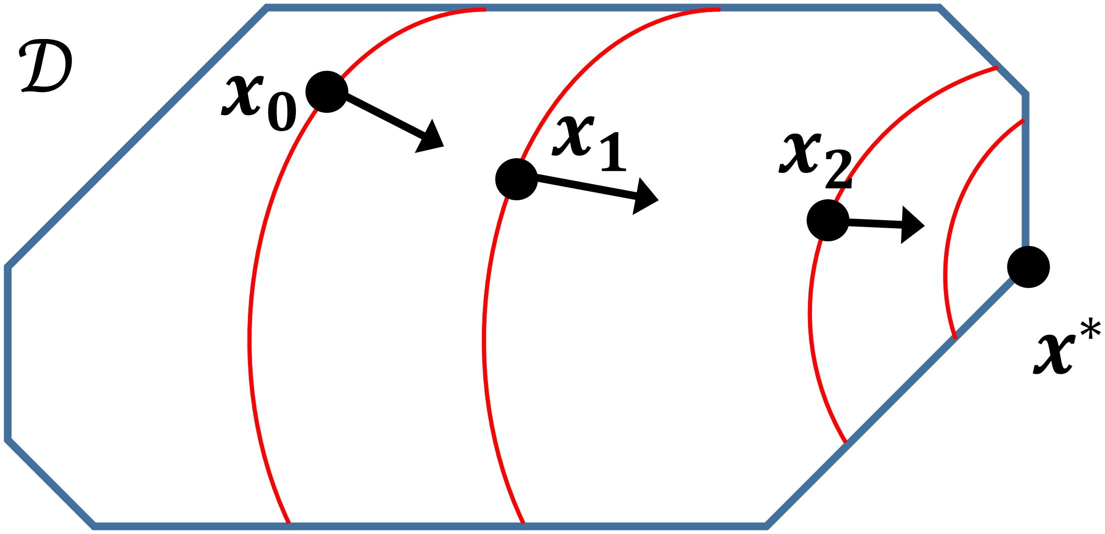
Second Order Methods – “Newton Steps”

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \frac{\mathrm{dc}(x_0)}{\mathrm{d}x}\Delta + \frac{1}{2}\frac{\mathrm{d}^2\mathrm{c}(x_0)}{\mathrm{d}x^2}\Delta^2$$



Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} c(\boldsymbol{x})$$

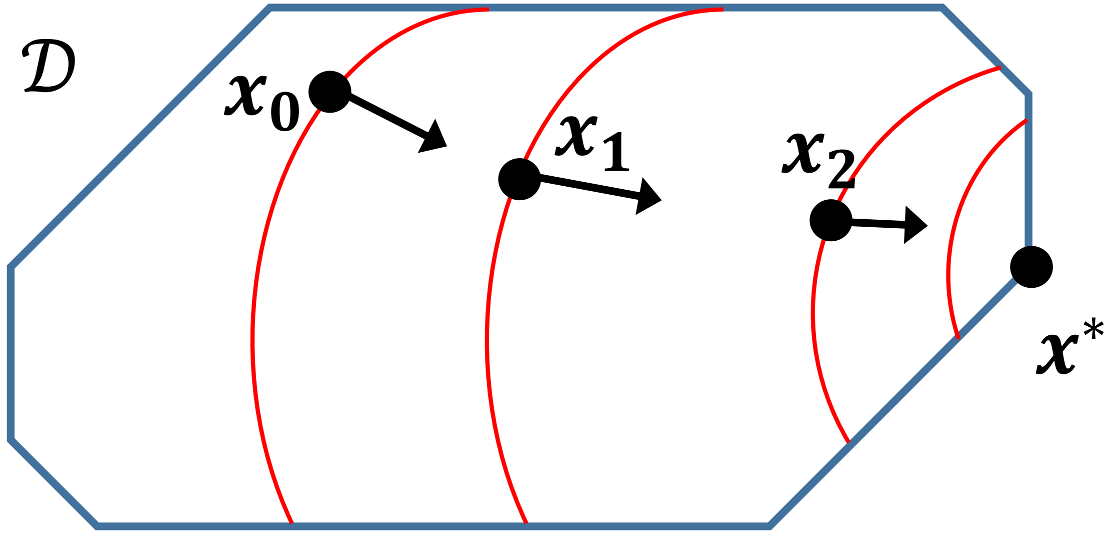


Second Order Methods – “Newton Steps”

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \sum_i \frac{\mathrm{dc}(x_0)}{\mathrm{d}x(i)} \Delta(i) + \frac{1}{2} \sum_{i,j} \frac{\mathrm{d}^2\mathrm{c}(x_0)}{\mathrm{d}x(i)\mathrm{d}x(j)} \Delta(i)\Delta(j)$$

Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} c(\boldsymbol{x})$$

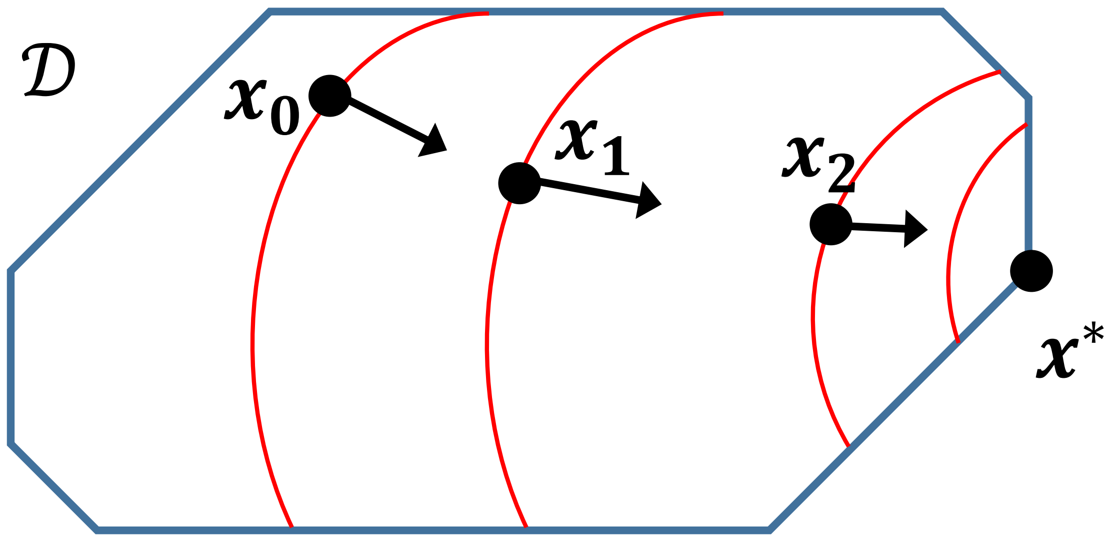


Second Order Methods – “Newton Steps”

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \nabla \mathrm{c}(\boldsymbol{x}_0) \cdot \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta} \cdot \nabla^2 \mathrm{c}(\boldsymbol{x}_0) \boldsymbol{\Delta}$$

Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} c(\boldsymbol{x})$$



Second Order Methods – “Newton Steps”

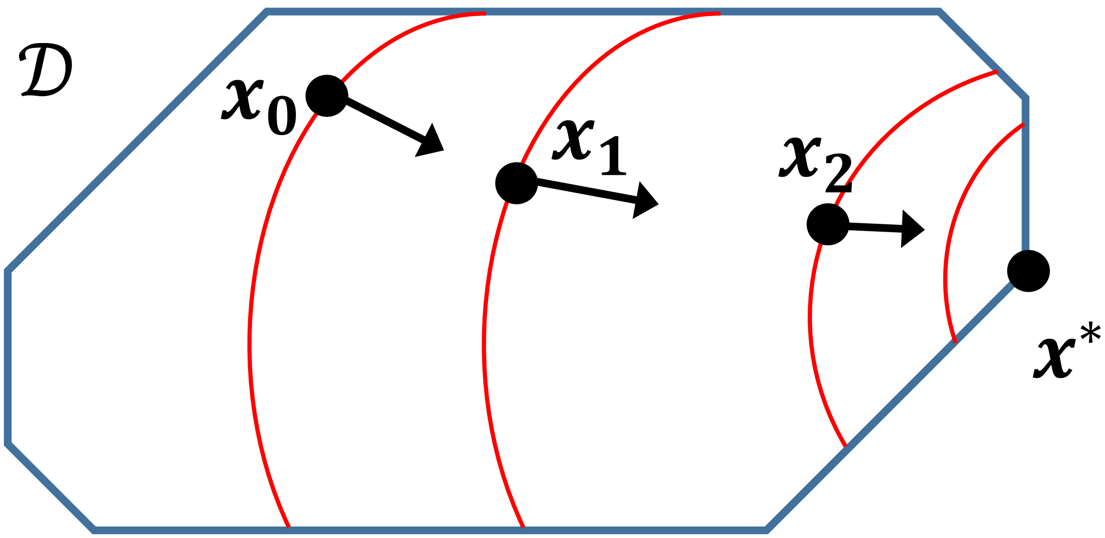
$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta} \cdot H \boldsymbol{\Delta}$$

$$H\boldsymbol{\Delta} = -\boldsymbol{g}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta} \cdot H \boldsymbol{\Delta}$$

Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} c(\boldsymbol{x})$$



Second Order Methods – "Newton Steps"

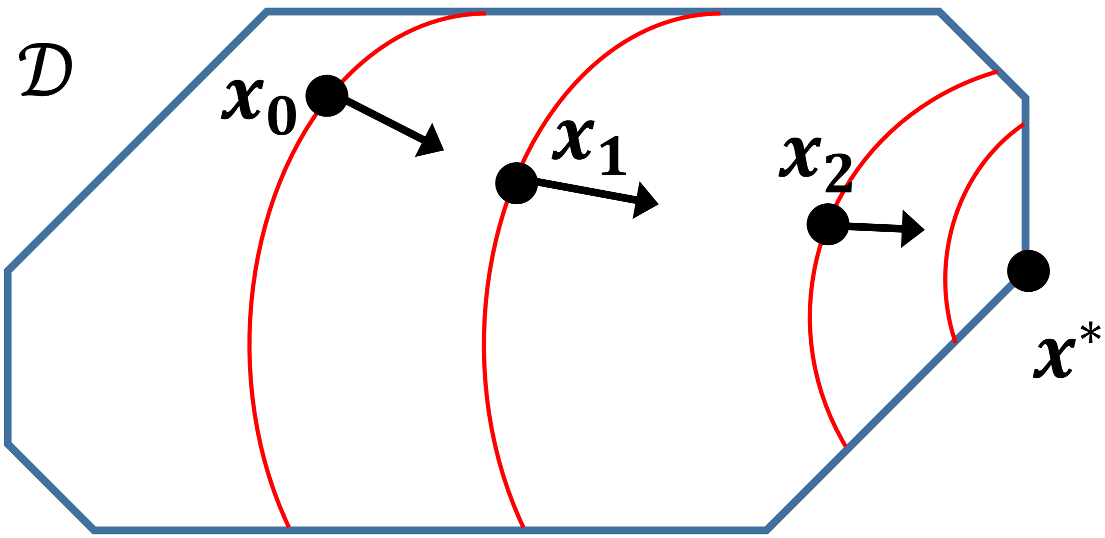
$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta} \cdot H \boldsymbol{\Delta}$$

$$H\boldsymbol{\Delta} = -\boldsymbol{g}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) - \boldsymbol{g} \cdot H^{-1}\boldsymbol{g} + \frac{1}{2} \boldsymbol{\Delta} \cdot H \boldsymbol{\Delta}$$

Optimization Primer

$$\min_{\boldsymbol{x}\in\mathcal{D}} c(\boldsymbol{x})$$



Second Order Methods – "Newton Steps"

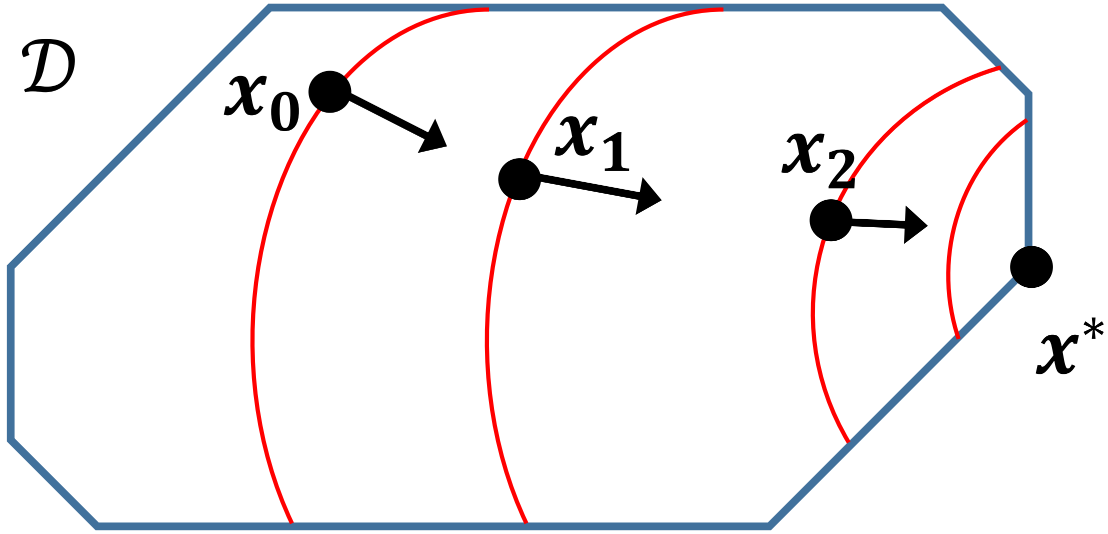
$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g}\cdot\boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta}\cdot H\boldsymbol{\Delta}$$

$$H\boldsymbol{\Delta} = -\boldsymbol{g}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) - \boldsymbol{g}\cdot H^{-1}\boldsymbol{g} + \frac{1}{2}H^{-1}\boldsymbol{g}\cdot H H^{-1}\boldsymbol{g}$$

Optimization Primer

$$\min_{\boldsymbol{x}\in\mathcal{D}} c(\boldsymbol{x})$$



Second Order Methods – "Newton Steps"

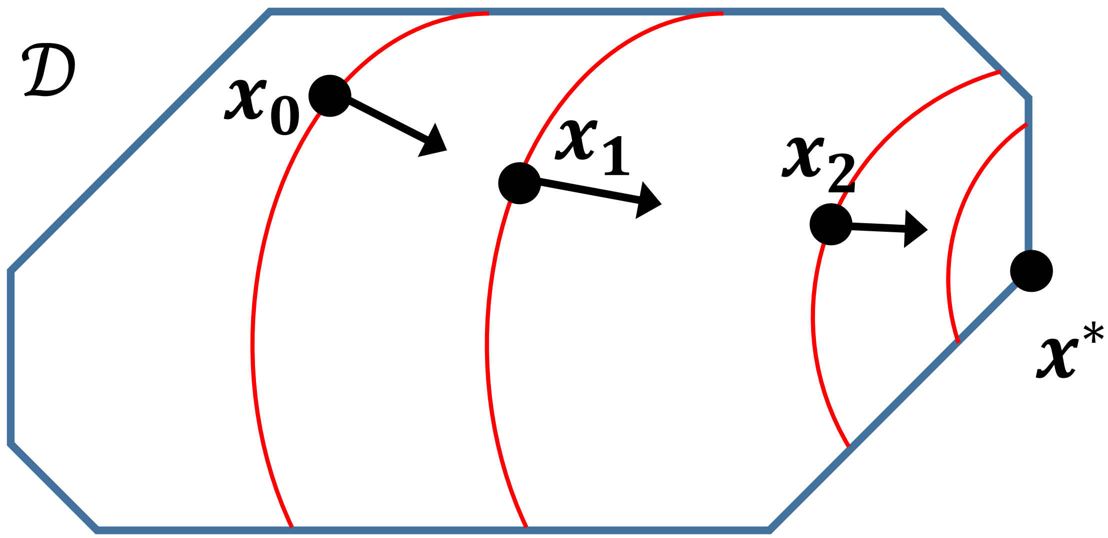
$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g}\cdot\boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta}\cdot H\boldsymbol{\Delta}$$

$$H\boldsymbol{\Delta} = -\boldsymbol{g}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) - \frac{1}{2}\boldsymbol{g}\cdot H^{-1}\boldsymbol{g}$$

Optimization Primer

$$\min_{\boldsymbol{x} \in \mathcal{D}} c(\boldsymbol{x})$$



Second Order Methods – "Newton Steps"

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \mathrm{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta} \cdot H \boldsymbol{\Delta}$$

$$H\boldsymbol{\Delta} = -\boldsymbol{g}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{\Delta}$$

A better step!

Second Order Methods

Usually finding step $\mathbf{\Delta}$ that solves

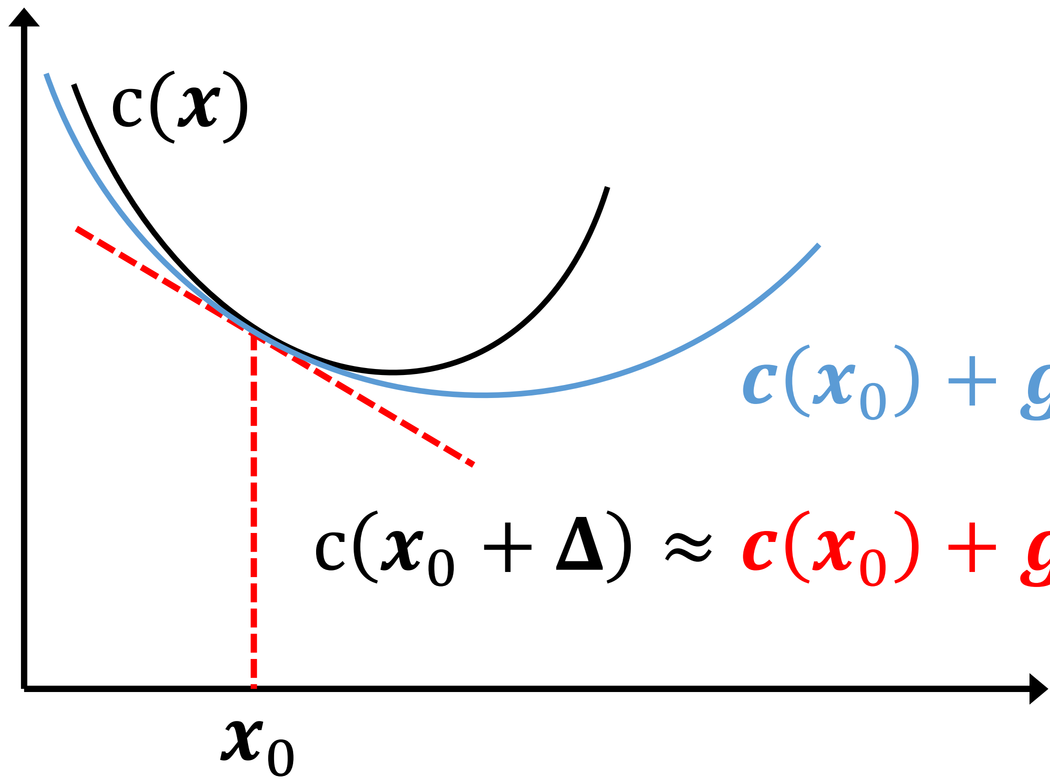
$$H\mathbf{\Delta} = -g$$

is too expensive!

But for optimization on graphs,

solve much faster than general linear equations

Convex Functions



$$c(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta} \cdot H\boldsymbol{\Delta}$$

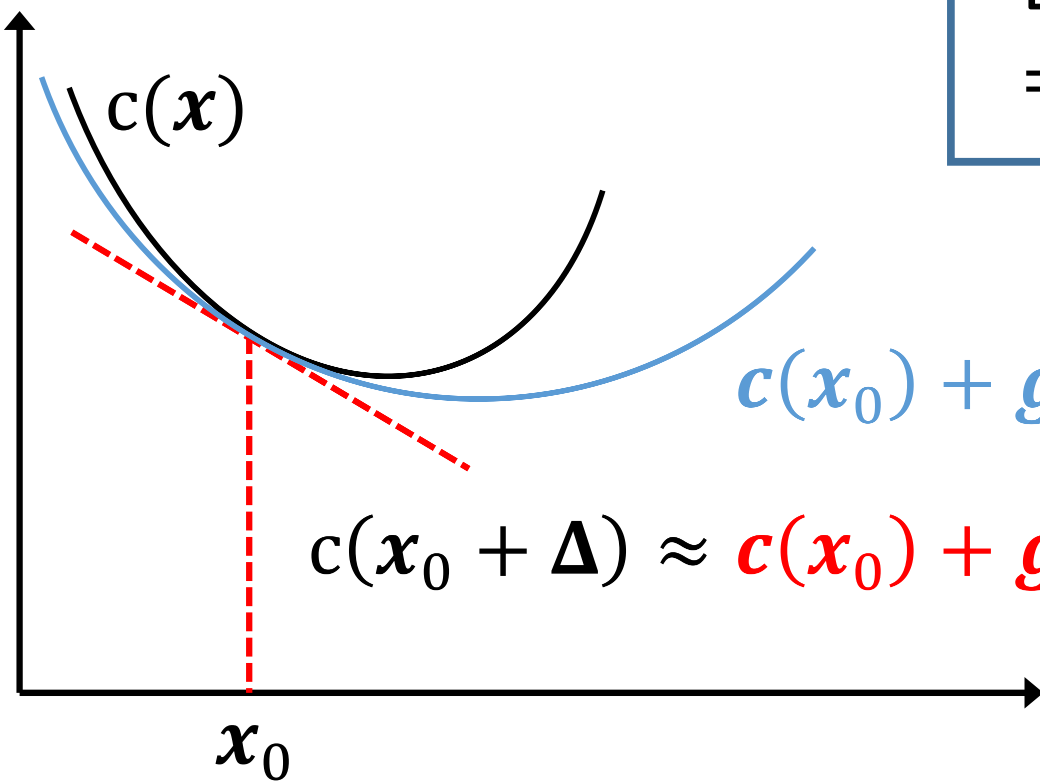
$$c(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx c(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta}$$

$$c(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx c(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \underbrace{\frac{1}{2}\boldsymbol{\Delta} \cdot H\boldsymbol{\Delta}}_{\geq 0}$$

Convex Functions

Why convex?

Local minimum
$\Longrightarrow$ global minimum



$$\boldsymbol{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta} \cdot H\boldsymbol{\Delta}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \boldsymbol{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta}$$

$$\mathrm{c}(\boldsymbol{x}_0 + \boldsymbol{\Delta}) \approx \boldsymbol{c}(\boldsymbol{x}_0) + \boldsymbol{g} \cdot \boldsymbol{\Delta} + \underbrace{\frac{1}{2}\boldsymbol{\Delta} \cdot H\boldsymbol{\Delta}}$$

No negative eigenvalues!

Optimization on Graphs

Graph $G = (V, E)$

$\boldsymbol{x} \in \mathbb{R}^V$

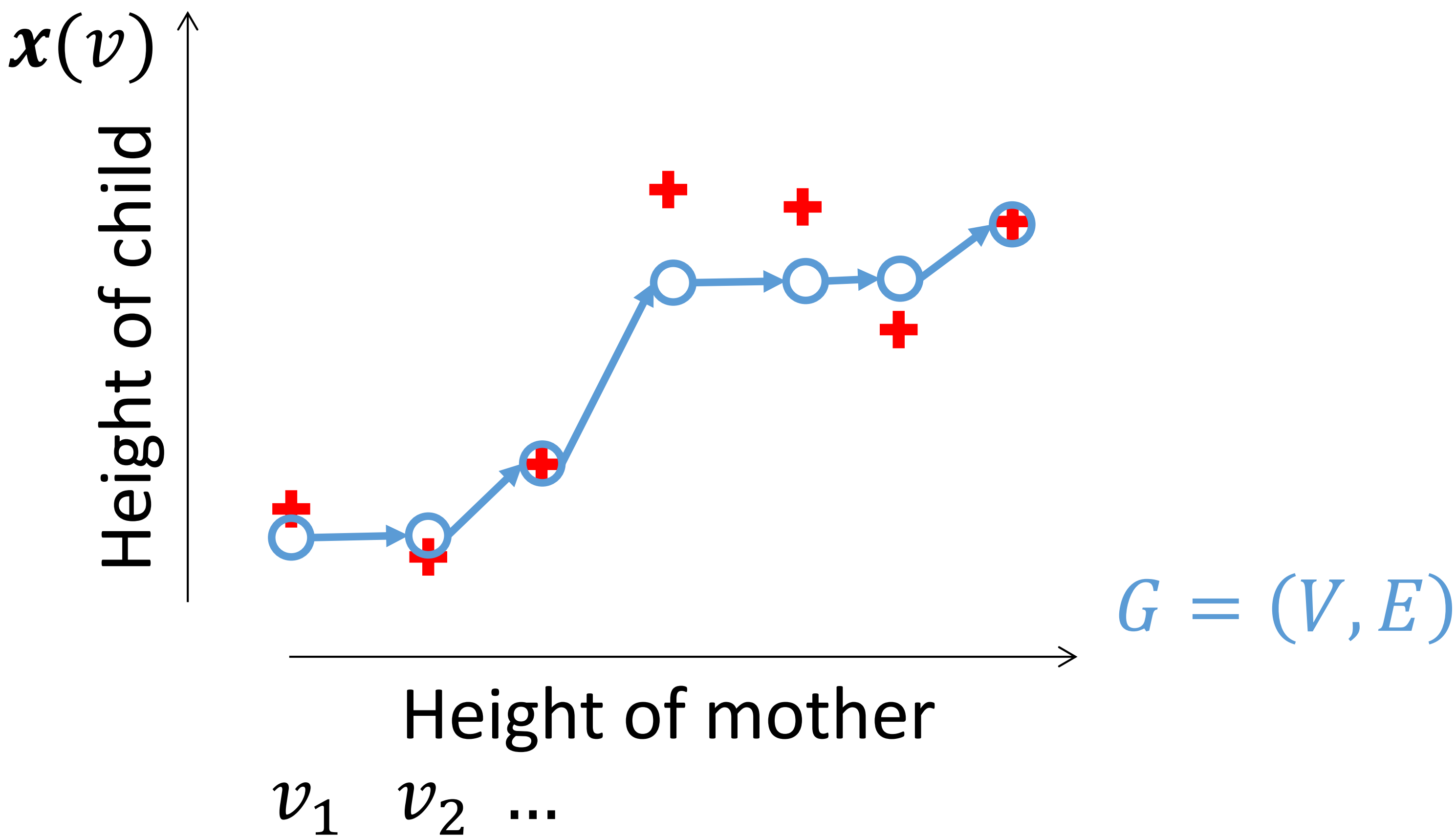
Constraints on pairs $\boldsymbol{x}(v), \boldsymbol{x}(u)$ for $(u, v) \in E$

Isotonic Regression

Constraint:
for all $(u, v) \in E$
$\boldsymbol{x}(u) \leq \boldsymbol{x}(v)$

Cost:

$$\min_{\boldsymbol{x}} \sum_{v \in V} (\boldsymbol{x}(v) - \text{child\_height}(v))^2$$



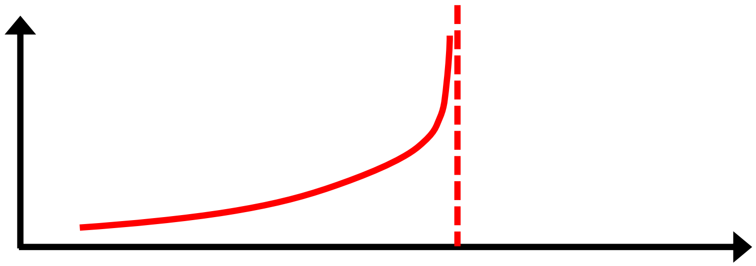
Optimization on Graphs

Graph $G = (V, E)$

$\boldsymbol{x} \in \mathbb{R}^V$

Constraints on pairs $\boldsymbol{x}(v), \boldsymbol{x}(u)$ for $(u, v) \in E$

(+ one weird trick)



Unconstrained Problem with Modified Cost

$\sum_{v \in V} c_v(\boldsymbol{x}(v)) + \sum_{(u,v) \in E} c_{(u,v)}(\boldsymbol{x}(u), \boldsymbol{x}(v))$

Graphs and Hessian Linear Equations

Newton Step: find $\boldsymbol{\Delta}$ s.t. $H\boldsymbol{\Delta} = -\boldsymbol{g}$

Gaussian Elimination: $O(n^3)$ time for $n \times n$ matrix H.
"Faster" methods: $O(n^{2.373})$ time

Hessian from
sum of convex functions on two variables
$=$
Symmetric M-matrix
$\approx$
Laplacian

Spielman-Teng '04:
Laplacian linear equations
can be solved
in $\tilde{O}(\#\text{edges})$ time

Hessian Linear Equations

Find $\boldsymbol{\Delta}$ s.t. $H\boldsymbol{\Delta} = -\boldsymbol{g}$

Gaussian Elimination: $O(n^3)$ time for $n \times n$ matrix H.
"Faster" methods: $O(n^{2.373})$ time

Hessian from

sum of convex functions on two variables

$=$

Symmetric M-matrix

$\approx$

Laplacian

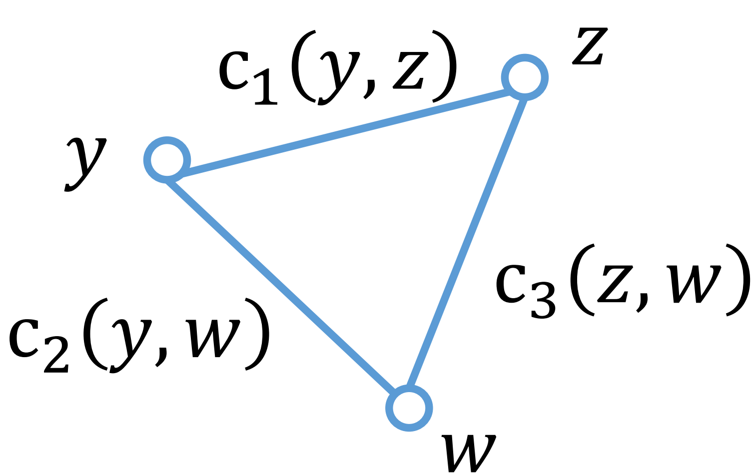
Daitch-Spielman '08:
Symmetric M-matrix
linear equations can be solved
in $\tilde{O}(\#\text{edges})$ time

Hessians & Graphs

$$\boldsymbol{x} = \begin{pmatrix} y \\ z \\ w \end{pmatrix}$$

Graph-Structured Cost Function

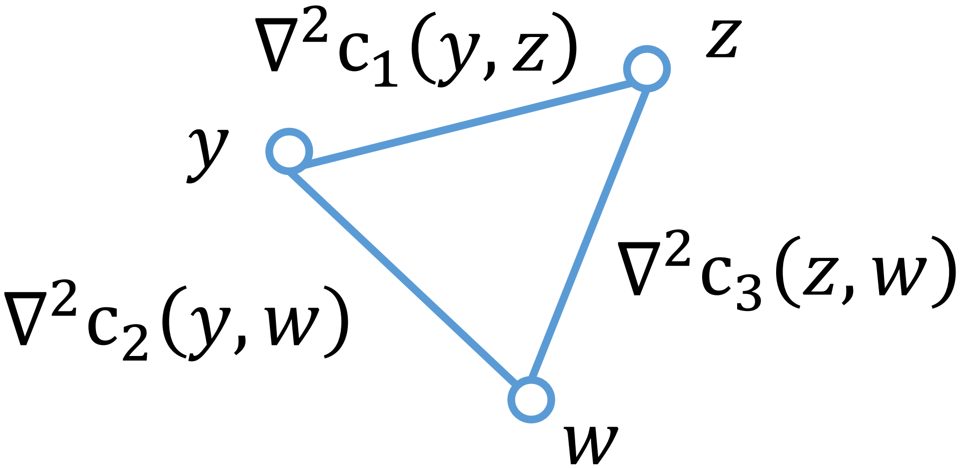
$$\mathrm{c}(\boldsymbol{x}) = \mathrm{c}_1(y, z) + \mathrm{c}_2(y, w) + \mathrm{c}_3(z, w)$$



$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y, z) + \nabla^2 \mathrm{c}_2(z, w) + \nabla^2 \mathrm{c}_3(w, y)$$

Second Derivatives

$$\nabla^2 c(\boldsymbol{x}) = \nabla^2 c_1(y,z) + \nabla^2 c_2(y,w) + \nabla^2 c_3(z,w)$$

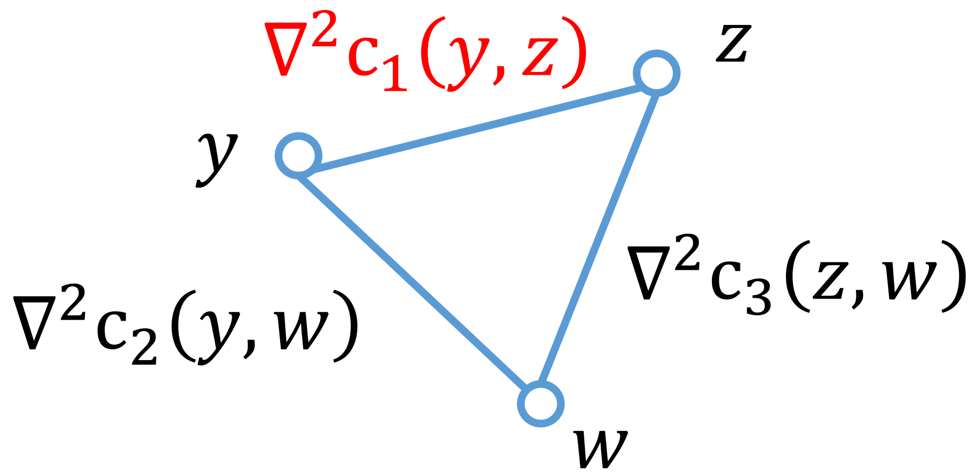


$\nabla^2 c(\boldsymbol{x})$

	y	z	w
y	0	0	0
z	0	0	0
w	0	0	0

Second Derivatives

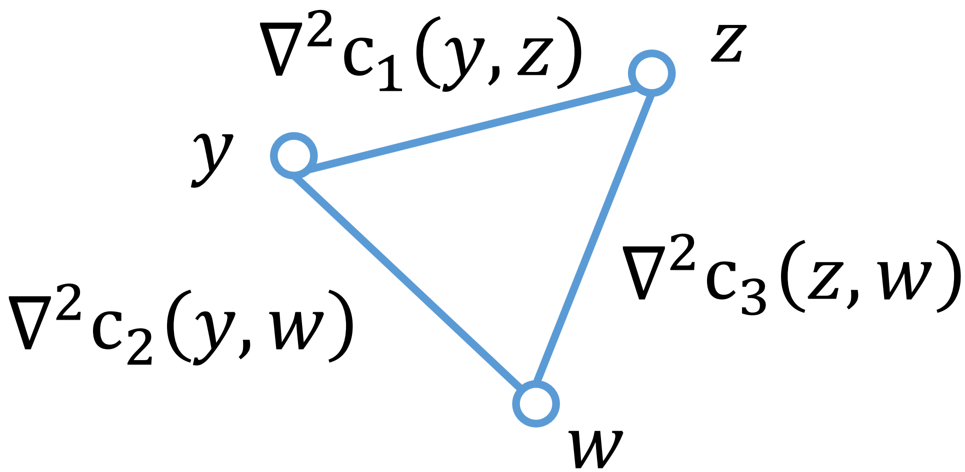
$$\nabla^2 c(\boldsymbol{x}) = \nabla^2 c_1(y,z) + \nabla^2 c_2(y,w) + \nabla^2 c_3(z,w)$$



$$\nabla^2 c(\boldsymbol{x}) \qquad \begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$



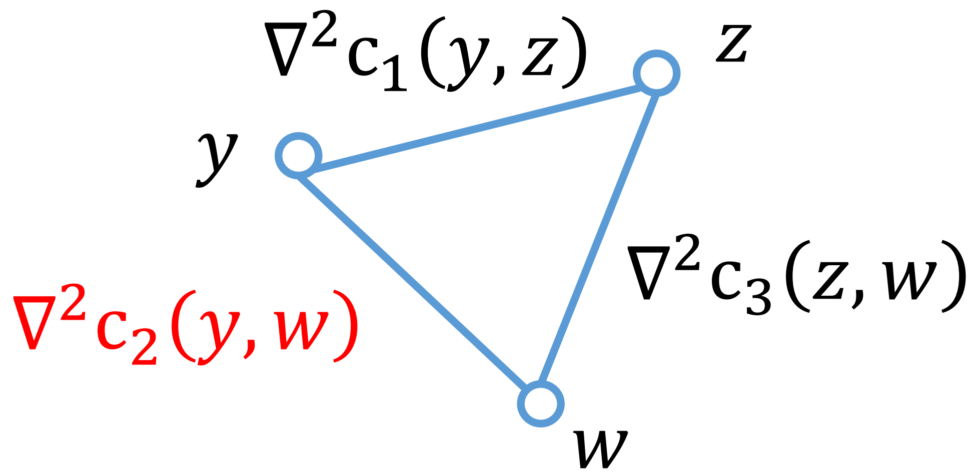
$\nabla^2 \mathrm{c}(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Second Derivatives

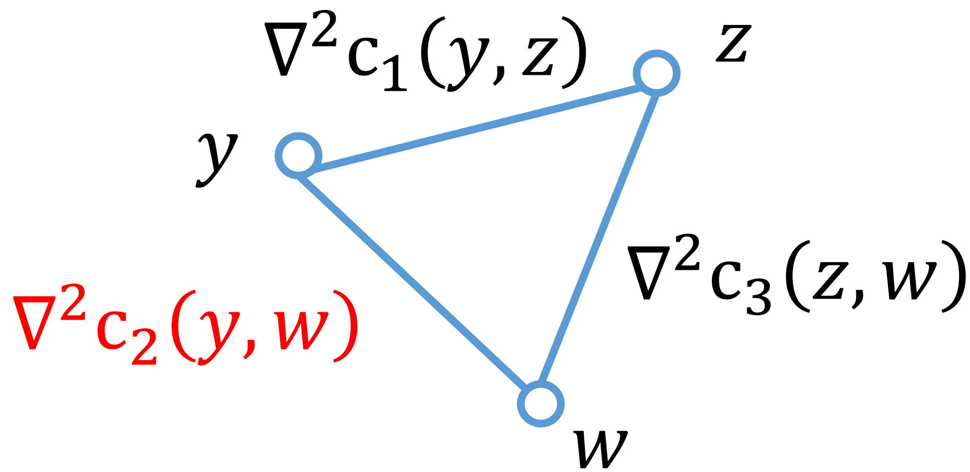
$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$



$$\nabla^2 \mathrm{c}(\boldsymbol{x}) \quad \begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 c(\boldsymbol{x}) = \nabla^2 c_1(y, z) + \nabla^2 c_2(y, w) + \nabla^2 c_3(z, w)$$

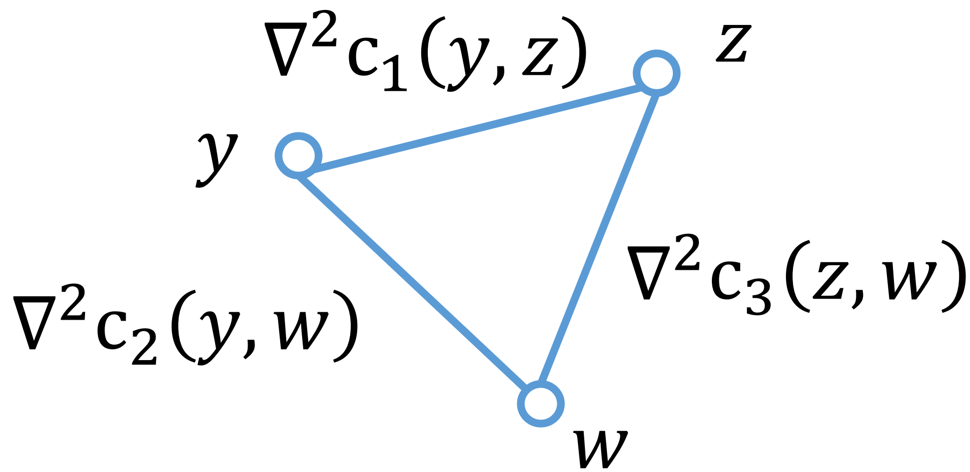


$\nabla^2 c(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 2 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & 2 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 c(\boldsymbol{x}) = \nabla^2 c_1(y, z) + \nabla^2 c_2(y, w) + \nabla^2 c_3(z, w)$$

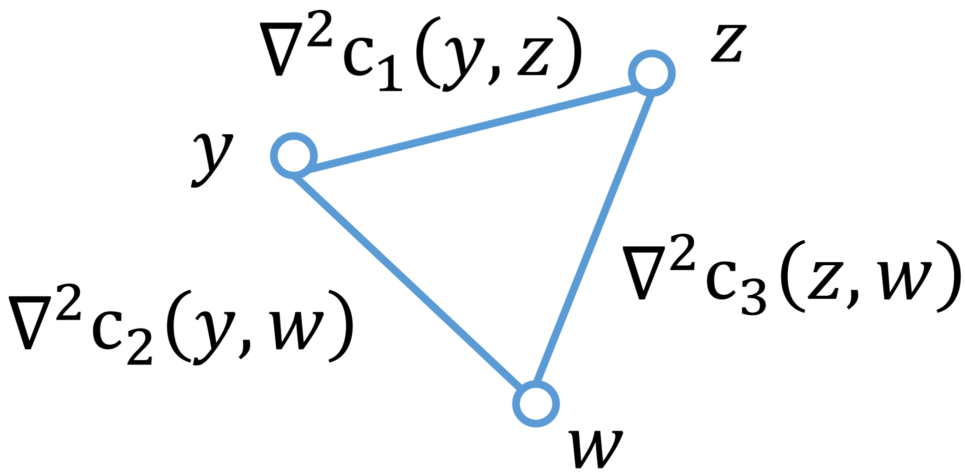


$\nabla^2 c(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 1 & 0 \\ -2 & 0 & 2 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 2 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & 2 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$

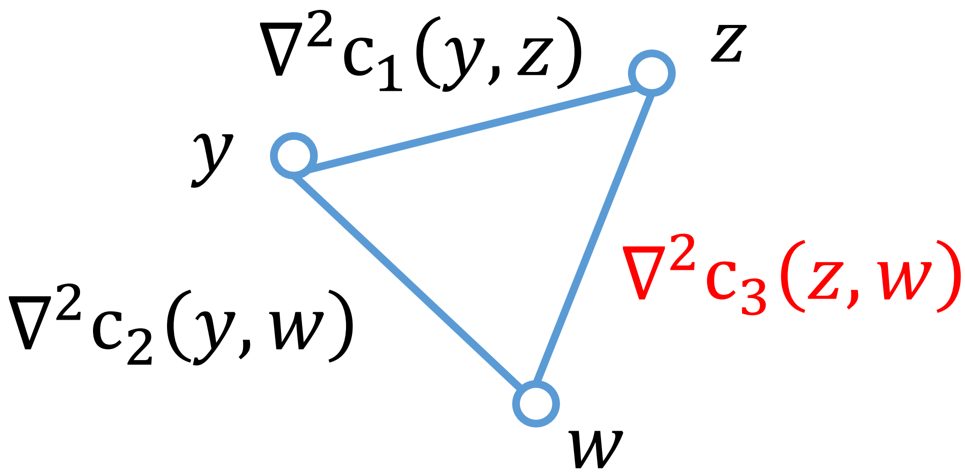


$\nabla^2 \mathrm{c}(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{matrix} \;\;y & \;\;\;z & \;\;\;w \end{matrix} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 1 & 0 \\ -2 & 0 & 2 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$

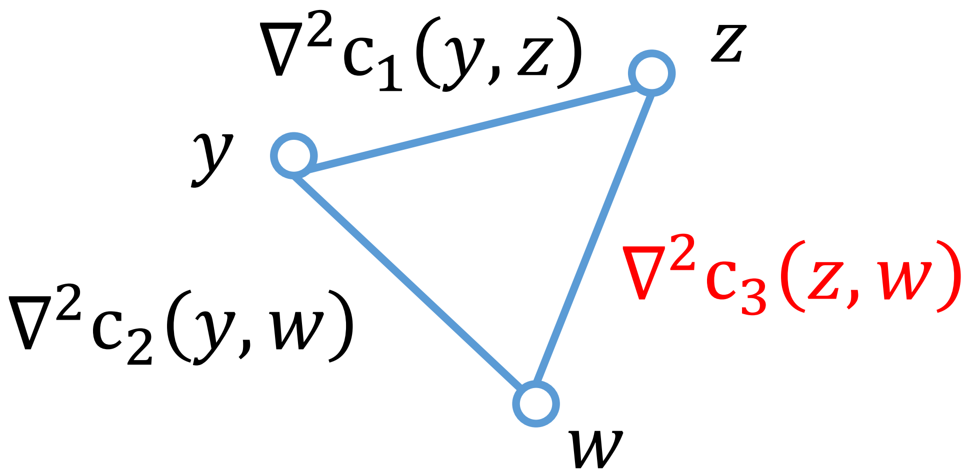


$\nabla^2 \mathrm{c}(\boldsymbol{x})$

	y	z	w
y	3	−1	−2
z	−1	1	0
w	−2	0	2

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$

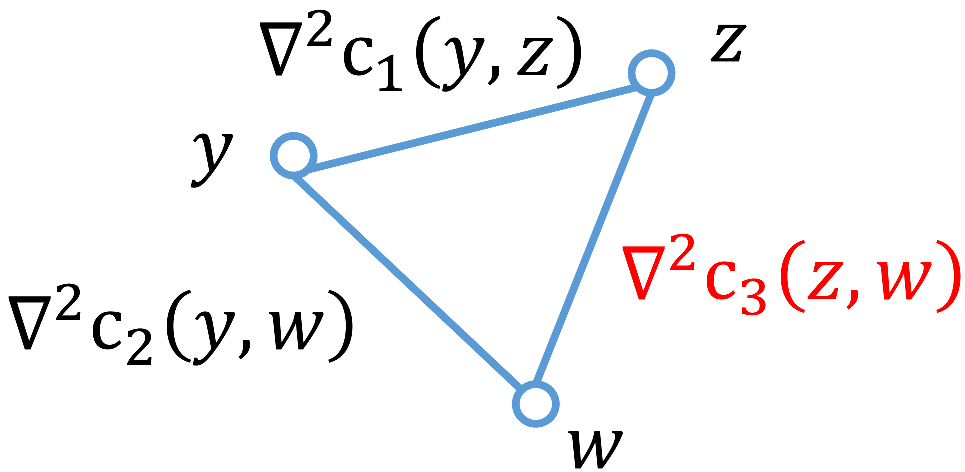


$\nabla^2 \mathrm{c}(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 1 & 0 \\ -2 & 0 & 2 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & \color{red}{1} & \color{red}{-1} \\ 0 & \color{red}{-1} & \color{red}{1} \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y, z) + \nabla^2 \mathrm{c}_2(y, w) + \nabla^2 \mathrm{c}_3(z, w)$$

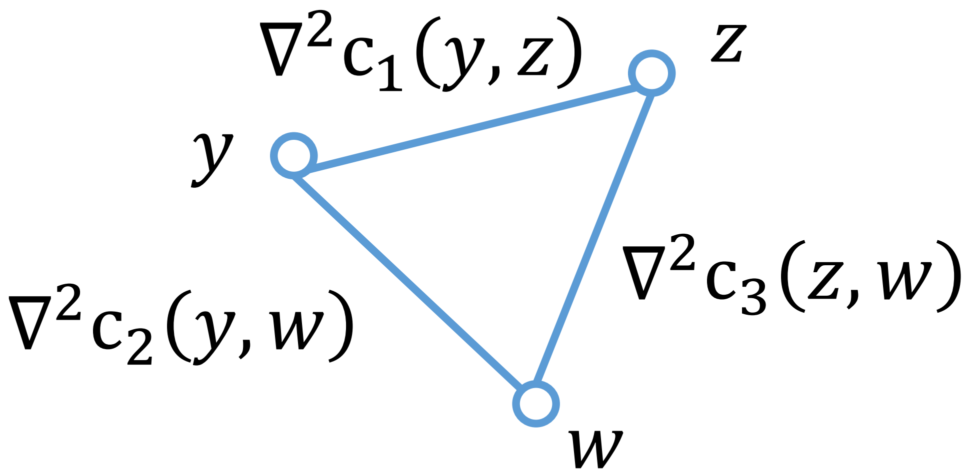


$\nabla^2 \mathrm{c}(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 2 & -1 \\ -2 & -1 & 3 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y, z) + \nabla^2 \mathrm{c}_2(y, w) + \nabla^2 \mathrm{c}_3(z, w)$$

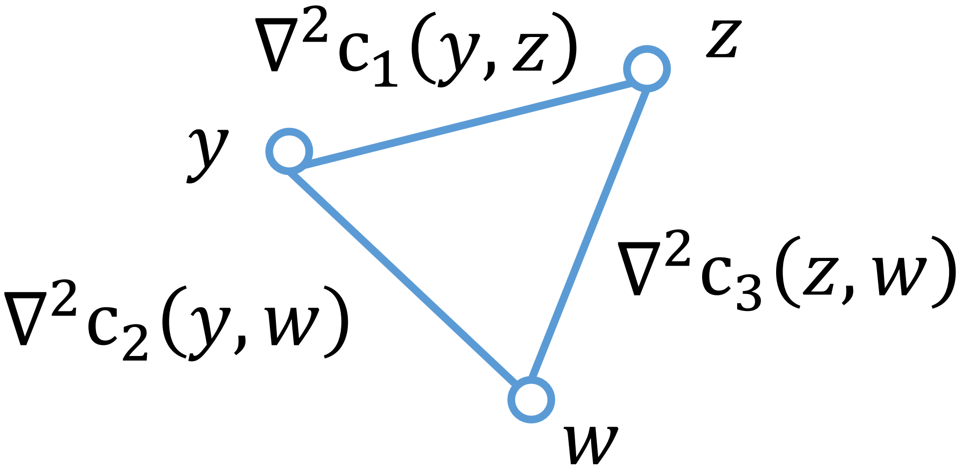


$\nabla^2 \mathrm{c}(\boldsymbol{x})$

$$\begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 2 & -1 \\ -2 & -1 & 3 \end{pmatrix} \end{array}$$

Second Derivatives

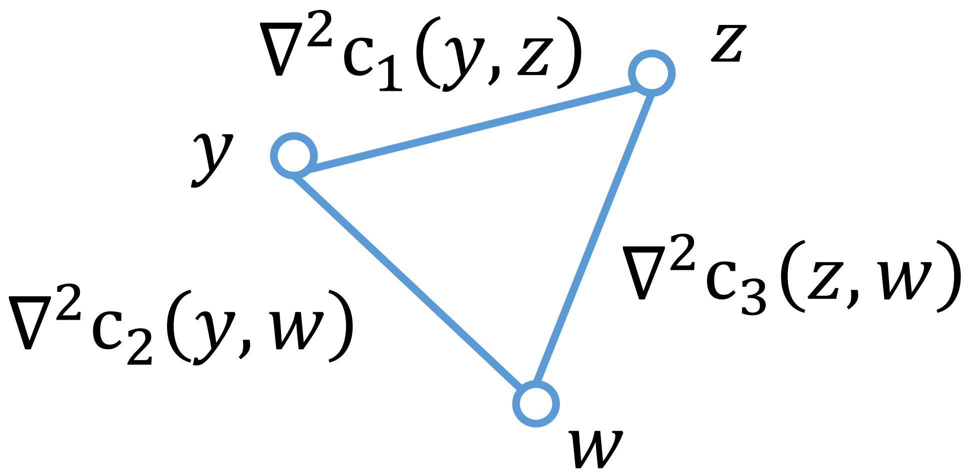
$$\nabla^2 \mathrm{c}(\boldsymbol{x}) = \nabla^2 \mathrm{c}_1(y,z) + \nabla^2 \mathrm{c}_2(y,w) + \nabla^2 \mathrm{c}_3(z,w)$$



$$\nabla^2 \mathrm{c}(\boldsymbol{x}) \quad \begin{array}{c} \\ y \\ z \\ w \end{array} \begin{array}{c} \begin{array}{ccc} y & z & w \end{array} \\ \begin{pmatrix} 3 & -1 & -2 \\ -1 & 2 & -1 \\ -2 & -1 & 3 \end{pmatrix} \end{array}$$

Second Derivatives

$$\nabla^2 c(\boldsymbol{x}) = \nabla^2 c_1(y,z) + \nabla^2 c_2(y,w) + \nabla^2 c_3(z,w)$$



If every term looks like then matrix is Laplacian

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Solving a PSD System

$\boldsymbol{Ax} = \boldsymbol{b}$

Gaussian Elimination

$$\boldsymbol{A} = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1/4 & 1 & 0 & 0 \\ -1/2 & 0 & 1 & 0 \\ -1/4 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 16 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Solving a PSD System

$\boldsymbol{Ax} = \boldsymbol{b}$

Gaussian Elimination

Find $\boldsymbol{U}$, upper triangular matrix, s.t.

$$\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{A}$$

Then solve

$$\boldsymbol{U}^\top \boldsymbol{y} = \boldsymbol{b}$$

$$\boldsymbol{U} \;\; \boldsymbol{x} = \boldsymbol{y}$$

Easy to solve in $\boldsymbol{U}^\top$ and $\boldsymbol{U}$

Solving a Laplacian System

$\boldsymbol{Lx = b}$

Approximate Gaussian Elimination [**K**S16]
Find $\boldsymbol{U}$, upper triangular matrix, s.t.

$$\boldsymbol{U}^\top \boldsymbol{U} \approx \boldsymbol{L}$$

$\boldsymbol{U}$ is sparse.

A few iterative solves to get
approximate solution to $\boldsymbol{Lx = b}$

Gaussian Elimination on Laplacians

What is special about Gaussian Elimination on Laplacians?

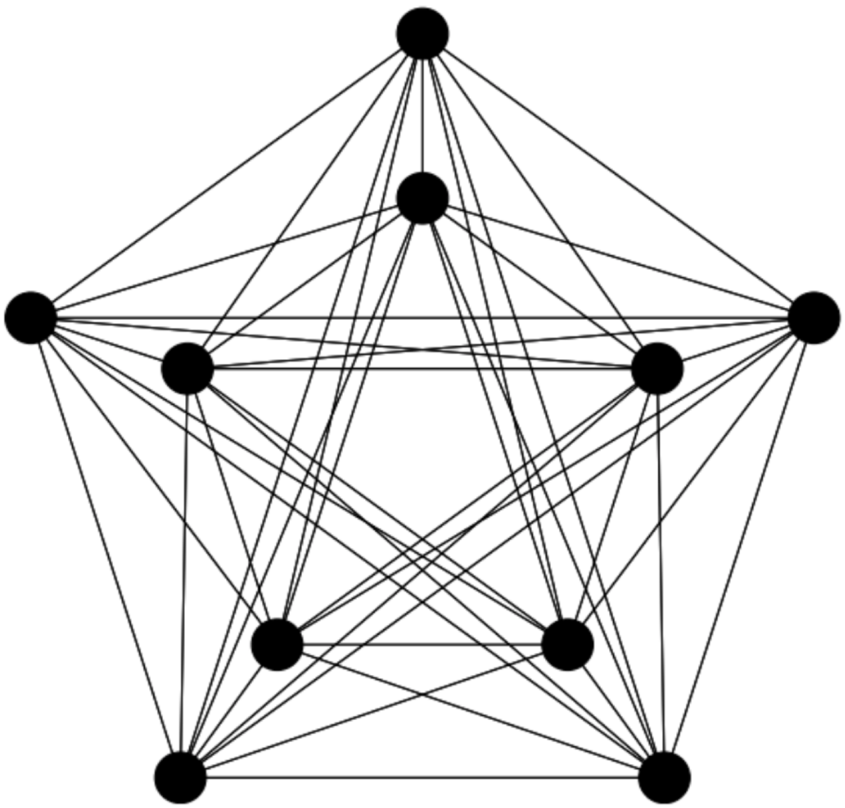
The **remaining matrix** is always Laplacian.

$$\boldsymbol{L} = \begin{pmatrix} 16 & -8 & -4 & -4 \\ -8 & 8 & 0 & 0 \\ -4 & 0 & 4 & 0 \\ -4 & 0 & 0 & 4 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1/2 & 1 & 0 & 0 \\ -1/4 & 0 & 1 & 0 \\ -1/4 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 16 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 3 & -1 \\ 0 & -2 & -1 & 3 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A new Laplacian!

Sparse Approximation of Laplacians



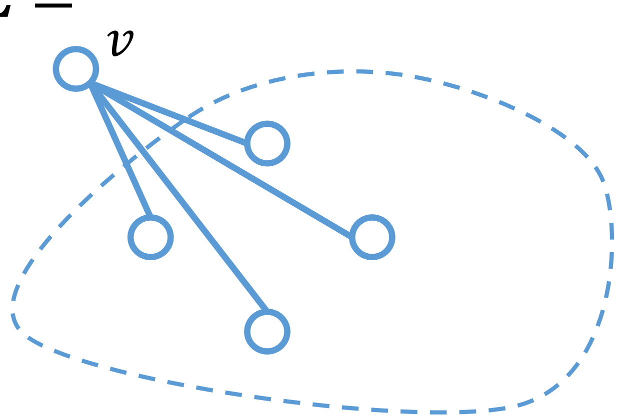
$\approx$

Why is Gaussian Elimination Slow?

Solving $\boldsymbol{Lx} = \boldsymbol{b}$ by Gaussian Elimination can take $\Omega(n^3)$ time.

The main issue is **fill**

$\boldsymbol{L} =$

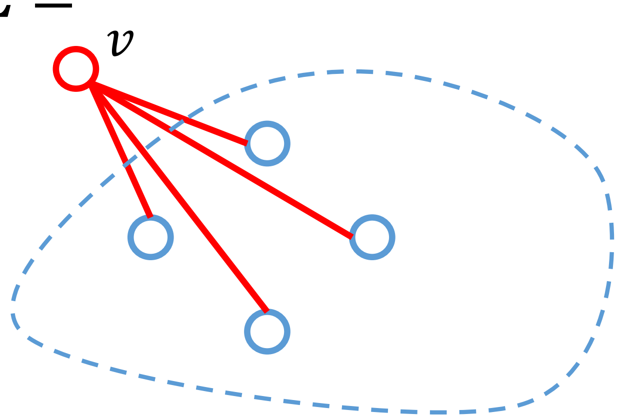


Why is Gaussian Elimination Slow?

Solving $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$ by Gaussian Elimination can take $\Omega(n^3)$ time.

The main issue is **fill**

$\boldsymbol{L} =$

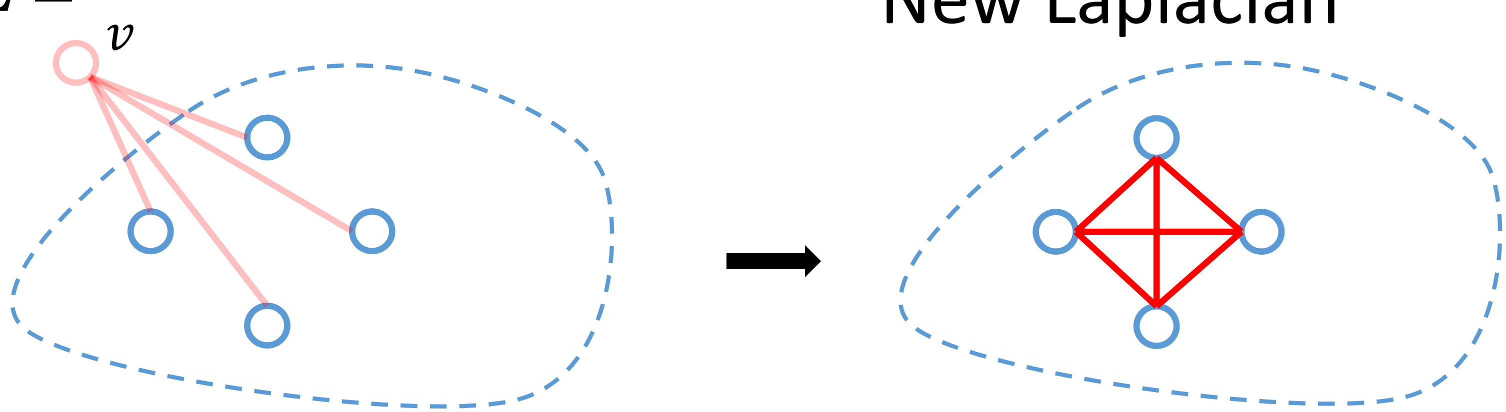


Why is Gaussian Elimination Slow?

Solving $\boldsymbol{Lx} = \boldsymbol{b}$ by Gaussian Elimination can take $\Omega(n^3)$ time.

The main issue is **fill**

$\boldsymbol{L} =$ New Laplacian

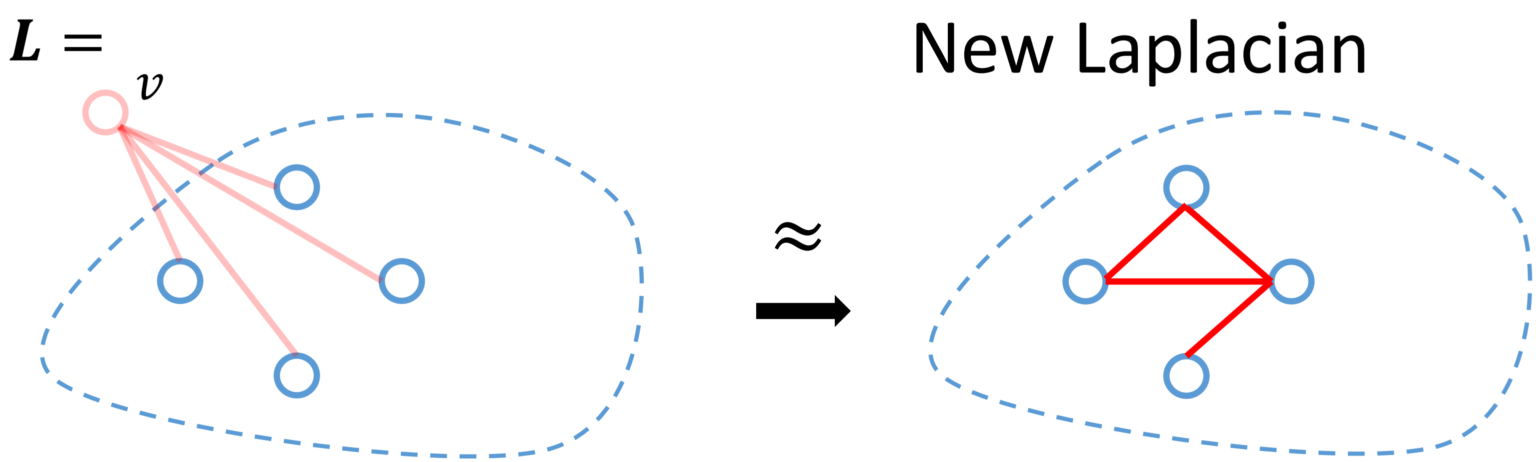


Elimination creates a clique on the neighbors of v

Why is Gaussian Elimination Slow?

Solving $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$ by Gaussian Elimination can take $\Omega(n^3)$ time.

The main issue is **fill**



Laplacian cliques can be sparsified!

Gaussian Elimination

1. Pick a vertex v to eliminate
2. Add the clique created by eliminating v
3. Repeat until done

Approximate Gaussian Elimination

1. Pick a vertex v to eliminate
2. Add the clique created by eliminating v
3. Repeat until done

Approximate Gaussian Elimination

1. Pick a random vertex v to eliminate
2. Add the clique created by eliminating v
3. Repeat until done

Approximate Gaussian Elimination

1. Pick a random vertex v to eliminate
2. Sample the clique created by eliminating v
3. Repeat until done

Resembles **randomized** Incomplete Cholesky

Key Proof Idea: Matrix Martingales

Correct in Expectation + Concentration of Measure

Optimization on Graphs

Variants of this framework
have been used for many problems:

Maximum flow [DS08, CKMST11,KMP12, Mad13]

Minimum cost flow [LS14]

Negative Weight Shortest Paths [CMSV16]

Isotonic Regression [**K**RS15]

Regularized Lipschitz Learning on Graphs [**K**RSS15]

Optimization on Graphs

Laplacian solvers used for many other problems in TCS

Learning on graphs [ZGL03, ZS04, ZBLWS04]

Graph partitioning [OSV12]

Sampling random spanning trees [KM09,MST15,D**K**PRS17,DPR17,S18]

Graph sparsification [SS08,LKP12,**K**PPS17]

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	
Rand 4-regular	17.2	12	13.7	1.7 (CG)

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)
CG = Conjugate Gradient

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	
Rand 4-regular	17.2	12	13.7	1.7 (CG)
Pref attach	9.5	10.8	5.1	3.2 (ICC)

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)
CG = Conjugate Gradient, ICC = Incomplete Cholesky

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	
Rand 4-regular	17.2	12	13.7	1.7 (CG)
Pref attach	9.5	10.8	5.1	3.2 (ICC)
Chimera (2500100,11)	28	350	74	

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)
CG = Conjugate Gradient, ICC = Incomplete Cholesky

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	
Rand 4-regular	17.2	12	13.7	1.7 (CG)
Pref attach	9.5	10.8	5.1	3.2 (ICC)
Chimera (2500100,11)	28	350	74	
Min Cost Flow 1	5.9	4.3	>60	
Min Cost Flow 2	6.8	23	29	

Apx Elim = Approximate Elimination (K & Spielman)
CMG = Combinatorial Multigrid (Koutis)
LAMG = Lean Algebraic Multigrid (Livne & Brandt)
CG = Conjugate Gradient, ICC = Incomplete Cholesky

Julia Package: Laplacians.jl

Graph	Apx Elim	CMG	LAMG	Other
1000x1000 grid	6.3	3.0	15	
100x100x100 grid	8.7	3.4	15	
Rand 4-regular	17.2	12	13.7	1.7 (CG)
Pref attach	9.5	10.8	5.1	3.2 (ICC)
Chimera (2500100,11)	28	350	74	
Min Cost Flow 1	5.9	4.3	>60	
Min Cost Flow 2	6.8	23	29	

Summary: Approximate Elimination processes between 300k and 500k entries per second, for 8 digit accuracy

Others vary widely

Thanks!

https://github.com/danspielman/Laplacians.jl/

Approximate Gaussian Elimination for Laplacians
(R. Kyng, S. Sachdeva)

Faster Approximate Lossy Generalized Flow
via Interior Point Algorithms
(S. Daitch, D. Spielman)

Fast, Provable Algorithms for Isotonic Regression
in all Lp norms
(R. Kyng, A.B. Rao, S. Sachdeva)

Nearly-linear time algorithms for graph partitioning,
graph sparsification, and solving linear systems
(S.-H. Teng, D. Spielman)