Optimization on Dynamic Graphs

Justin Romberg, Georgia Tech ECE
Bellairs Workshop on ML and SSP for Data on Graphs
Holetown, Barbados
January 23, 2024

Collaborators

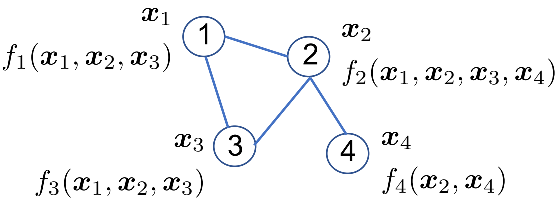
Tomer Hamam
Imagry, Haifa, Israel

Joe Driscoll
Georgia Tech, ECE

Shared variables described by a graph

- Nodes i: variables $\boldsymbol{x}_i$ and function f_i
- Edge (i, j): f_i and f_j *share variables*
- Optimization program

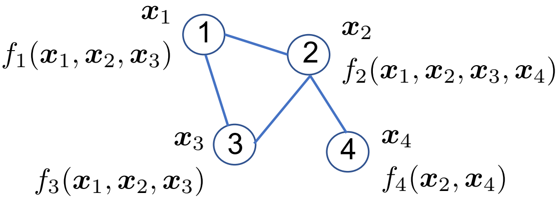
$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \quad \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right)$$



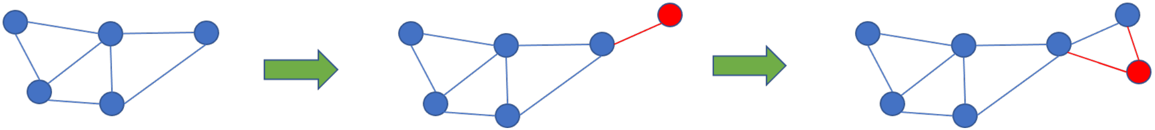
Shared variables described by a graph

- Nodes i: variables $\boldsymbol{x}_i$ and function f_i
- Edge (i, j): f_i and f_j *share variables*
- Optimization program

$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \quad \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right)$$

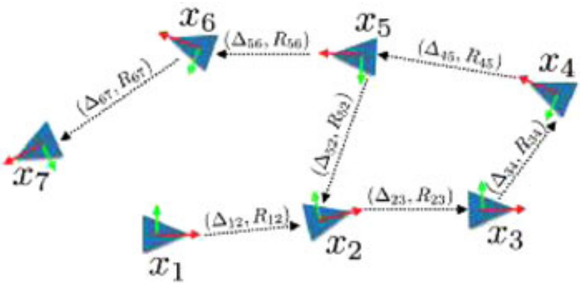


Key question: *how does solution change as the graph evolves?*



Example: Localization and Pose Estimation

- Estimate poses: $x_i = (\text{position}, \text{orientation})$ at time i from relative measurements

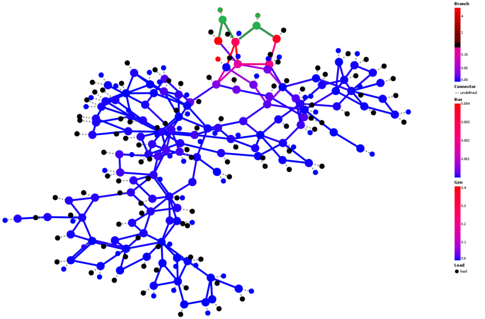


(Carlone et al, '16)

- Naturally posed as a nonconvex least-squares problem on a dynamic graph
  Semidefinite relaxation is a convex problem on a dynamic graph

Example: AC optimal power flow

- Solve for power production that minimizes generation cost while obeying physical constraints

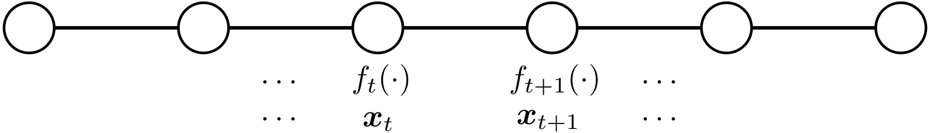


- Naturally posed as a nonconvex problem on a graph
  Also has a semidefinite relaxation

Streaming optimization (chain graph)

One important special case:

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$



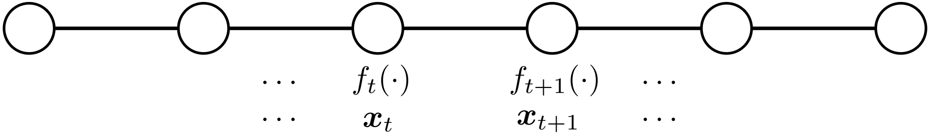
Streaming solution: at time T,

1. observe f_T; initialize $\hat{\boldsymbol{x}}_{T|T}$
2. update solutions $\hat{\boldsymbol{x}}_{t|T}$, $t \leq T$

Streaming optimization (chain graph)

One important special case:

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$



Streaming solution: at time T,

1. observe f_T; initialize $\hat{\boldsymbol{x}}_{T|T}$
2. update solutions $\hat{\boldsymbol{x}}_{t|T}$, $t \leq T$

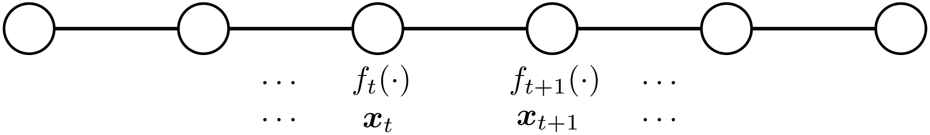
Key questions:

1. does $\hat{\boldsymbol{x}}_{t|T}$ converge as $T \to \infty$?
2. if so, how quickly?

Streaming optimization (chain graph)

Streaming least-squares:

$$\underset{\{\boldsymbol{x}_t\}}{\text{minimize}} \sum_{t=1}^{T} \|\boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{x}_{t-1} - \boldsymbol{y}_t\|_2^2$$



Classical: The Kalman filter

Linear dynamical system for state evolution and measurement:

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-1} + \boldsymbol{d}_t$$
$$\boldsymbol{y}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_t + \boldsymbol{e}_t$$

Observe $\{\boldsymbol{y}_t\}_{t=1}^T$, estimate $\{\boldsymbol{x}_t\}_{t=1}^T$ ...

Classical: The Kalman filter

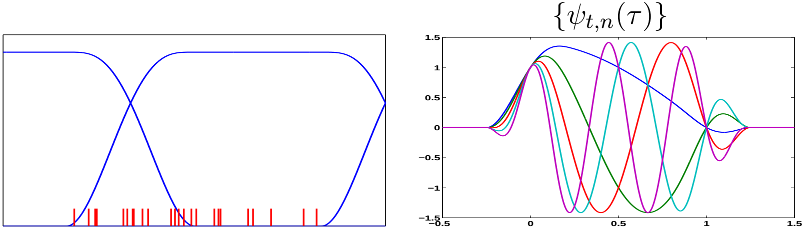
Linear dynamical system for state evolution and measurement:

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-1} + \boldsymbol{d}_t$$
$$\boldsymbol{y}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_t + \boldsymbol{e}_t$$

Observe $\{\boldsymbol{y}_t\}_{t=1}^T$, estimate $\{\boldsymbol{x}_t\}_{t=1}^T$ ...

$$\underset{\{\boldsymbol{x}_t\}}{\text{minimize}} \sum_{t=1}^{T} \|\boldsymbol{\Phi}_t \boldsymbol{x}_t - \boldsymbol{y}_t\|_2^2 + \lambda_t \|\boldsymbol{x}_t - \boldsymbol{F}_{t-1} \boldsymbol{x}_{t-1}\|_2^2$$

Streaming recon. from non-uniform samples



Sample batch t at locations $\tau_1, \ldots, \tau_M$
One batch overlaps frame bundles $t-1$ and t

Single sample at τ_m

$$s(\tau_m) = \sum_n x_{t-1,n}\, \psi_{t-1,n}(\tau_m) + \sum_n x_{t,n} \psi_{t,n}(\tau_m)$$

Collecting all samples into vector $\boldsymbol{y}_t$, we can write

$$\boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{B}_t & \boldsymbol{A}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix}$$

Structured linear system

After collecting batches $t = 0, 1, \ldots, T$, we have the (possibly large) system

$$\Phi_T \underline{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{0} & \cdots & & & & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{0} & \cdots & & & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_2 & \boldsymbol{A}_2 & \boldsymbol{0} & \cdots & & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_3 & \boldsymbol{A}_3 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_4 & \boldsymbol{A}_4 & \cdots & \boldsymbol{0} \\ \vdots & & & & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & & & \cdots & \boldsymbol{B}_T & \boldsymbol{A}_T \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \boldsymbol{x}_4 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \\ \boldsymbol{y}_4 \\ \vdots \\ \boldsymbol{y}_T \end{bmatrix}.$$

Tri-diagonal structure

At every time T, the least-squares system is block tri-diagonal,

$$\boldsymbol{\Phi}_T^{\mathrm{T}}\boldsymbol{\Phi}_T\underline{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{E}_0^{\mathrm{T}} & \mathbf{0} & \cdots & & & \mathbf{0} \\ \boldsymbol{E}_0 & \boldsymbol{D}_1 & \boldsymbol{E}_1^{\mathrm{T}} & \mathbf{0} & \cdots & & \mathbf{0} \\ \mathbf{0} & \boldsymbol{E}_1 & \boldsymbol{D}_2 & \boldsymbol{E}_2^{\mathrm{T}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{E}_2 & \boldsymbol{D}_3 & \boldsymbol{E}_3^{\mathrm{T}} & \cdots & \mathbf{0} \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & & & \boldsymbol{E}_{T-2} & \boldsymbol{D}_{T-1} & \boldsymbol{E}_{T-1}^{\mathrm{T}} \\ \mathbf{0} & \cdots & & & \mathbf{0} & \boldsymbol{E}_{T-1} & \boldsymbol{D}_T \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \vdots \\ \boldsymbol{x}_{T-1} \\ \boldsymbol{x}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \vdots \\ \boldsymbol{g}_{T-1} \\ \boldsymbol{g}_T \end{bmatrix}$$

Factorization: Forward sweep

There is an easy LU factorization,

$$\begin{bmatrix} \boldsymbol{Q}_0 & \mathbf{0} & \cdots & & \mathbf{0} \\ \boldsymbol{E}_0 & \boldsymbol{Q}_1 & \mathbf{0} & & \\ \mathbf{0} & \boldsymbol{E}_1 & \boldsymbol{Q}_2 & \ddots & \\ \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{E}_{T-1} & \boldsymbol{Q}_T \end{bmatrix} \begin{bmatrix} \mathbf{I} & \boldsymbol{U}_0 & \mathbf{0} & & \\ \mathbf{0} & \mathbf{I} & \boldsymbol{U}_1 & \mathbf{0} & \\ \vdots & & \ddots & \ddots & \\ & & & \ddots & \boldsymbol{U}_{T-1} \\ \mathbf{0} & & & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_T \end{bmatrix}$$

where the $\boldsymbol{Q}_t$ and $\boldsymbol{U}_t$ can be computed *recursively*

Factorization: Forward sweep

There is an easy LU factorization,

$$\begin{bmatrix} \boldsymbol{Q}_0 & \mathbf{0} & \cdots & & \mathbf{0} \\ \boldsymbol{E}_0 & \boldsymbol{Q}_1 & \mathbf{0} & & \\ \mathbf{0} & \boldsymbol{E}_1 & \boldsymbol{Q}_2 & \ddots & \\ \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{E}_{T-1} & \boldsymbol{Q}_T \end{bmatrix} \begin{bmatrix} \mathbf{I} & \boldsymbol{U}_0 & \mathbf{0} & & \\ \mathbf{0} & \mathbf{I} & \boldsymbol{U}_1 & \mathbf{0} & \\ \vdots & & \ddots & \ddots & \\ & & & \ddots & \boldsymbol{U}_{T-1} \\ \mathbf{0} & & & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_T \end{bmatrix}$$

where the $\boldsymbol{Q}_t$ and $\boldsymbol{U}_t$ can be computed *recursively*

for $t = 1, 2, \ldots, T-1$

$$\boldsymbol{U}_{t-1} = \boldsymbol{Q}_{t-1}^{-1} \boldsymbol{E}_{t-1}^{\mathrm{T}}$$

$$\boldsymbol{Q}_t = \boldsymbol{D}_t - \boldsymbol{E}_{t-1} \boldsymbol{Q}_{t-1}^{-1} \boldsymbol{E}_{t-1}^{\mathrm{T}}$$

$$\boldsymbol{v}_t = \boldsymbol{Q}_t^{-1}(\boldsymbol{g}_t - \boldsymbol{E}_{t-1} \boldsymbol{v}_{t-1})$$

end

Solution update: Backward sweep

With estimates after T frames in hand

$$\left\{\hat{\boldsymbol{x}}_{0|T},\ \hat{\boldsymbol{x}}_{1|T},\ \ldots,\hat{\boldsymbol{x}}_{T|T}\right\} = \underset{\{\boldsymbol{x}_t\}}{\text{arg min}} \sum_{t=1}^{T} \|\boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{x}_{t-1} - \boldsymbol{y}_t\|^2$$

we introduce a new loss function with $(\boldsymbol{y}_{T+1}, \boldsymbol{A}_{T+1}, \boldsymbol{B}_{T+1})$

$$f_{T+1}(\boldsymbol{x}_T, \boldsymbol{x}_{T+1}) = \left\|\boldsymbol{A}_{T+1}\boldsymbol{x}_{T+1} - \boldsymbol{B}_{T+1}\boldsymbol{x}_T - \boldsymbol{y}_{T+1}\right\|^2$$

The solutions $\hat{\boldsymbol{x}}_{T+1|T+1}, \hat{\boldsymbol{x}}_{T|T+1}, \ldots, \hat{\boldsymbol{x}}_{0|T+1}$ can be computed with a *backward sweep*

Solution update: Backward sweep

With estimates after T frames in hand

$$\left\{\hat{\boldsymbol{x}}_{0|T},\ \hat{\boldsymbol{x}}_{1|T},\ \ldots,\hat{\boldsymbol{x}}_{T|T}\right\} = \underset{\{\boldsymbol{x}_t\}}{\text{arg min}} \sum_{t=1}^{T} \|\boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{x}_{t-1} - \boldsymbol{y}_t\|^2$$

we introduce a new loss function with $(\boldsymbol{y}_{T+1}, \boldsymbol{A}_{T+1}, \boldsymbol{B}_{T+1})$

$$f_{T+1}(\boldsymbol{x}_T, \boldsymbol{x}_{T+1}) = \left\|\boldsymbol{A}_{T+1}\boldsymbol{x}_{T+1} - \boldsymbol{B}_{T+1}\boldsymbol{x}_T - \boldsymbol{y}_{T+1}\right\|^2$$

The solutions $\hat{\boldsymbol{x}}_{T+1|T+1}, \hat{\boldsymbol{x}}_{T|T+1}, \ldots, \hat{\boldsymbol{x}}_{0|T+1}$ can be computed with a *backward sweep*

$$\boldsymbol{v}_{T+1} = \boldsymbol{Q}_{T+1}^{-1}(\boldsymbol{A}_{T+1}^{\mathrm{T}}\boldsymbol{y}_{T+1} + \boldsymbol{B}_{T+1}^{\mathrm{T}}\boldsymbol{y}_{T+1} - \boldsymbol{E}_T\boldsymbol{v}_T)$$
$$\hat{\boldsymbol{x}}_{T+1|T+1} = \boldsymbol{v}_{T+1}$$

for $t = T, T-1, \ldots, 0$

$$\hat{\boldsymbol{x}}_{t|T+1} = \boldsymbol{v}_t - \boldsymbol{U}_t\hat{\boldsymbol{x}}_{t+1|T+1}$$

end

Block diagonal dominance

$$\boldsymbol{\Phi}_T^{\mathrm{T}}\boldsymbol{\Phi}_T = \begin{bmatrix} \boldsymbol{D}_0 & \boldsymbol{E}_0^{\mathrm{T}} & \mathbf{0} & \cdots & & & \mathbf{0} \\ \boldsymbol{E}_0 & \boldsymbol{D}_1 & \boldsymbol{E}_1^{\mathrm{T}} & \mathbf{0} & \cdots & & \mathbf{0} \\ \mathbf{0} & \boldsymbol{E}_1 & \boldsymbol{D}_2 & \boldsymbol{E}_2^{\mathrm{T}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{E}_2 & \boldsymbol{D}_3 & \boldsymbol{E}_3^{\mathrm{T}} & \cdots & \mathbf{0} \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & & & \boldsymbol{E}_{T-2} & \boldsymbol{D}_{T-1} & \boldsymbol{E}_{T-1}^{\mathrm{T}} \\ \mathbf{0} & \cdots & & & \mathbf{0} & \boldsymbol{E}_{T-1} & \boldsymbol{D}_T \end{bmatrix}$$

The estimates will stabilize very quickly when

$$\kappa(1-\delta) \le \lambda_{\min}(\boldsymbol{D}_t) \le \lambda_{\max}(\boldsymbol{D}_t) \le \kappa(1+\delta), \quad \|\boldsymbol{E}_t\| \le \kappa\delta, \quad \text{for all } t$$

are akin to a kind of *block diagonal dominance*

Convergence

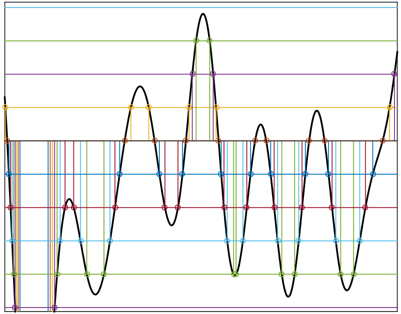
$$\left\{\hat{\boldsymbol{x}}_{0|T},\ \hat{\boldsymbol{x}}_{1|T},\ \ldots,\hat{\boldsymbol{x}}_{T|T}\right\} = \operatorname*{arg\,min}_{\{\boldsymbol{x}_t\}} \sum_{t=1}^{T} \|\boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{x}_{t-1} - \boldsymbol{y}_t\|^2$$

Theorem: For block diagonally dominant $\boldsymbol{D}_t, \boldsymbol{E}_t$, we have

- $\lim_{T\to\infty} \hat{\boldsymbol{x}}_{t|T} =: \hat{\boldsymbol{x}}_t^*$ exists for all t, and
- convergence is fast

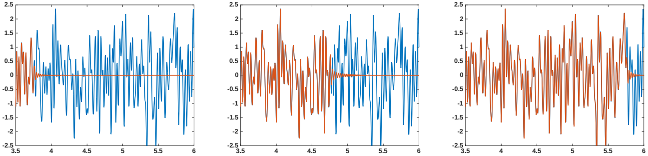
$$\|\hat{\boldsymbol{x}}_{t|T} - \hat{\boldsymbol{x}}_t^*\|_2 \ \leq\ C\left(\frac{\epsilon}{1-\epsilon}\right)^{T-t}, \quad \text{where } \epsilon \approx \delta.$$

Example: reconstruction from level crossings



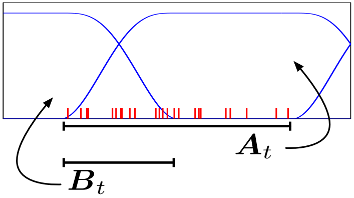
$$\log_{10}\left(\frac{\|\hat{\boldsymbol{x}}_{j|k}-\hat{\boldsymbol{x}}_j^*\|_2}{\|\hat{\boldsymbol{x}}_j^*\|_2}\right)$$

	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$
$k=4$	-0.31	—	—	—	—	—	—
$k=5$	-3.39	-0.32	—	—	—	—	—
$k=6$	-5.12	-3.24	-0.32	—	—	—	—
$k=7$	-7.28	-5.08	-3.46	-0.27	—	—	—
$k=8$	-9.27	-7.08	-5.60	-3.44	-0.34	—	—
$k=9$	-10.84	-8.65	-7.17	-5.19	-2.48	-0.22	—
$k=10$	-13.27	-11.08	-9.60	-7.62	-4.90	-3.44	-0.36



Moral: You can just update 3 frames in the past and still be very accurate ...

Random samples



$$\boldsymbol{D}_t = \boldsymbol{A}_t^{\mathrm{T}} \boldsymbol{A}_t + \boldsymbol{B}_{t+1}^{\mathrm{T}} \boldsymbol{B}_{t+1},$$
$$\boldsymbol{E}_{t-1} = \boldsymbol{B}_t^{\mathrm{T}} \boldsymbol{A}_t$$

N = number of basis functions per frame bundle
M = number of samples per batch
For samples selected uniformly at random, we have with probability $1 - \epsilon$

$$1 - \delta \leq \lambda_{\min}(\boldsymbol{D}_t) \leq \lambda_{\min}(\boldsymbol{D}_t) \leq 1 + \delta, \quad \|\boldsymbol{E}_t\| \leq \delta, \quad \text{for fixed } t$$

with

$$\delta \leq C \sqrt{\frac{N}{M} \log(N/\epsilon)}$$

so we can take

$$M \gtrsim N \log(N/\epsilon).$$

Streaming optimization: convex case

We want to solve

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$

where f_t are smooth and strongly convex

Streaming optimization: convex case

We want to solve

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$

where f_t are smooth and strongly convex

Streaming solution: at time T,

1. observe f_T; initialize $\hat{\boldsymbol{x}}_{T|T}$
2. update solutions $\hat{\boldsymbol{x}}_{t|T}$

Key questions:

1. does $\hat{\boldsymbol{x}}_{t|T}$ converge as $T \to \infty$?
2. if so, how quickly?

Streaming optimization: convex case

We want to solve

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

where f_t are smooth and strongly convex

Key piece of structure: gradient in frame t involves only f_t and f_{t+1}

$$\nabla J_T(\underline{\boldsymbol{x}}) = \begin{bmatrix} \nabla_0 f_1(\boldsymbol{x}_0, \boldsymbol{x}_1) \\ \nabla_1 f_1(\boldsymbol{x}_0, \boldsymbol{x}_1) + \nabla_1 f_2(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ \vdots \\ \nabla_{T-1} f_{T-1}(\boldsymbol{x}_{T-2}, \boldsymbol{x}_{T-1}) + \nabla_{T-1} f_T(\boldsymbol{x}_{T-1}, \boldsymbol{x}_T) \\ \nabla_T f_T(\boldsymbol{x}_{T-1}, \boldsymbol{x}_T) \end{bmatrix}$$

Streaming optimization: convex case

We want to solve

$$\underset{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T}{\text{minimize}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1},\boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

where f_t are smooth and strongly convex

Key piece of structure: Hessian is block tri-diagonal

$$\nabla^2 J_T(\underline{\boldsymbol{x}}) = \begin{bmatrix} \boldsymbol{H}_0 & \boldsymbol{E}_0^{\mathrm{T}} & \boldsymbol{0} & \cdots & & & \boldsymbol{0} \\ \boldsymbol{E}_0 & \boldsymbol{H}_1 & \boldsymbol{E}_1^{\mathrm{T}} & \boldsymbol{0} & \cdots & & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{E}_1 & \boldsymbol{H}_2 & \boldsymbol{E}_2^{\mathrm{T}} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{E}_2 & \boldsymbol{H}_3 & \boldsymbol{E}_3^{\mathrm{T}} & \cdots & \boldsymbol{0} \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & & & \boldsymbol{E}_{T-2} & \boldsymbol{H}_{T-1} & \boldsymbol{E}_{T-1}^{\mathrm{T}} \\ \boldsymbol{0} & \cdots & & & \boldsymbol{0} & \boldsymbol{E}_{T-1} & \boldsymbol{H}_T \end{bmatrix},$$

Convergence: convex case

Let

$$\{\hat{\boldsymbol{x}}_{0|T}, \ldots, \hat{\boldsymbol{x}}_{T|T}\} = \operatorname*{arg\,min}_{\{\boldsymbol{x}_t\}} \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

Theorem: If there are $\{\boldsymbol{w}_T\}$ such that

$$\|\nabla f_T(\hat{\boldsymbol{x}}_{T-1|T-1}, \boldsymbol{w}_T)\| \leq \text{Const} \quad \text{for all} \quad T,$$

then

- $\lim_{T\to\infty} \hat{\boldsymbol{x}}_{t|T} =: \hat{\boldsymbol{x}}_t^*$ exists for all t, and

Convergence: convex case

Let

$$\{\hat{\boldsymbol{x}}_{0|T}, \ldots, \hat{\boldsymbol{x}}_{T|T}\} = \operatorname*{arg\,min}_{\{\boldsymbol{x}_t\}} \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

Theorem: If there are $\{\boldsymbol{w}_T\}$ such that

$$\|\nabla f_T(\hat{\boldsymbol{x}}_{T-1|T-1}, \boldsymbol{w}_T)\| \leq \mathrm{Const} \quad \text{for all} \quad T,$$

then

- $\lim_{T\to\infty} \hat{\boldsymbol{x}}_{t|T} =: \hat{\boldsymbol{x}}_t^*$ exists for all t, and
- convergence is fast

$$\|\hat{\boldsymbol{x}}_{t|T} - \boldsymbol{x}_t^*\| \ \leq \ C \left(\frac{2L-\mu}{2L+\mu}\right)^{T-t}$$

(L = smoothness parameter, μ = strong convexity parameter)

Convergence: convex case

Proof sketch: Start from

$$\{\hat{\boldsymbol{x}}_{0|T}, \dots, \hat{\boldsymbol{x}}_{T|T}\} = \underset{\{\boldsymbol{x}_t\}}{\text{arg min}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

Add f_{T+1}, initialize

$$\boldsymbol{w}_t^{(0)} = \begin{cases} \hat{\boldsymbol{x}}_{t|T}, & t \leq T, \\ \text{(something)}, & t = T+1 \end{cases}$$

Use gradient descent to move to the new solution, trace the steps

Tracking the steps of gradient descent

Gradient descent:

$$\underline{\boldsymbol{w}}^{(k+1)} = \underline{\boldsymbol{w}}^{(k)} - \alpha \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(k)})$$

(we know this converges linearly)

Tracking the steps of gradient descent

Gradient descent:

$$\underline{\boldsymbol{w}}^{(k+1)} = \underline{\boldsymbol{w}}^{(k)} - \alpha \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(k)})$$

(we know this converges linearly)

Notice that

$$\nabla J_{T+1}(\underline{\boldsymbol{w}}^{(0)}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ * \\ * \end{bmatrix}$$

Tracking the steps of gradient descent

Gradient descent:

$$\underline{\boldsymbol{w}}^{(k+1)} = \underline{\boldsymbol{w}}^{(k)} - \alpha \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(k)})$$

(we know this converges linearly)

Notice that

$$\nabla J_{T+1}(\underline{\boldsymbol{w}}^{(0)}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ * \\ * \end{bmatrix}, \quad \underline{\boldsymbol{w}}^{(1)} = \underline{\boldsymbol{w}}^{(0)} - \alpha \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(0)}) \quad \Rightarrow \quad \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(1)}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ * \\ * \\ * \end{bmatrix}$$

Tracking the steps of gradient descent

Gradient descent:

$$\underline{\boldsymbol{w}}^{(k+1)} = \underline{\boldsymbol{w}}^{(k)} - \alpha \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(k)})$$

(we know this converges linearly)

Notice that

$$\nabla J_{T+1}(\underline{\boldsymbol{w}}^{(0)}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ * \\ * \end{bmatrix}, \quad \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(1)}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ * \\ * \\ * \end{bmatrix}, \quad \nabla J_{T+1}(\underline{\boldsymbol{w}}^{(2)}) = \begin{bmatrix} 0 \\ \vdots \\ * \\ * \\ * \\ * \end{bmatrix}, \quad \cdots$$

frame t is not touched until iteration $k = T - t$ ...

Convergence: convex case

Let

$$\{\hat{\boldsymbol{x}}_{0|T}, \ldots, \hat{\boldsymbol{x}}_{T|T}\} = \underset{\{\boldsymbol{x}_t\}}{\text{arg min}} \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

Theorem: If there are $\boldsymbol{w}_T$ such that

$$\|\nabla f_T(\hat{\boldsymbol{x}}_{T-1|T-1}, \boldsymbol{w}_T)\| \leq \text{Const} \quad \text{for all} \quad T,$$

then

- $\lim_{T\to\infty} \hat{\boldsymbol{x}}_{t|T} =: \hat{\boldsymbol{x}}_t^*$ exists for all t, and
- convergence is fast

$$\|\hat{\boldsymbol{x}}_{t|T} - \boldsymbol{x}_t^*\| \leq C \left(\frac{2L-\mu}{2L+\mu}\right)^{T-t}$$

Convergence: convex case

Let

$$\{\hat{\boldsymbol{x}}_{0|T}, \dots, \hat{\boldsymbol{x}}_{T|T}\} = \operatorname*{arg\,min}_{\{\boldsymbol{x}_t\}} \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = J_T(\underline{\boldsymbol{x}})$$

Theorem: If there are $\boldsymbol{w}_T$ such that

$$\|\nabla f_T(\hat{\boldsymbol{x}}_{T-1|T-1}, \boldsymbol{w}_T)\| \leq \mathrm{Const} \quad \text{for all} \quad T, \qquad ??$$

then

- $\lim_{T\to\infty} \hat{\boldsymbol{x}}_{t|T} =: \hat{\boldsymbol{x}}_t^*$ exists for all t, and
- convergence is fast

$$\|\hat{\boldsymbol{x}}_{t|T} - \boldsymbol{x}_t^*\| \;\leq\; C \left(\frac{2L-\mu}{2L+\mu}\right)^{T-t}$$

Convergence: convex case

Theorem: If the local minimizers

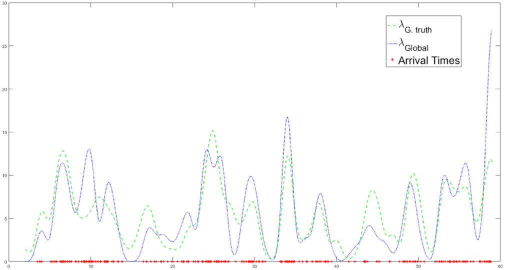
$$(\tilde{\boldsymbol{x}}_{t-1|t}, \tilde{\boldsymbol{x}}_{t|t}) = \text{arg min}\, f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$

are bounded and the Hessian is diagonally dominant, then there are $\{\boldsymbol{w}_T\}$ such that

$$\|\nabla f_T(\hat{\boldsymbol{x}}_{T-1|T-1}, \boldsymbol{w}_T)\| \leq \text{Const} \quad \text{for all} \quad T.$$

Example: Non-homogenous Poisson process

Given "spike" observations at $\tau_1, \ldots, \tau_M$, estimate the background intensity $\lambda(t)$

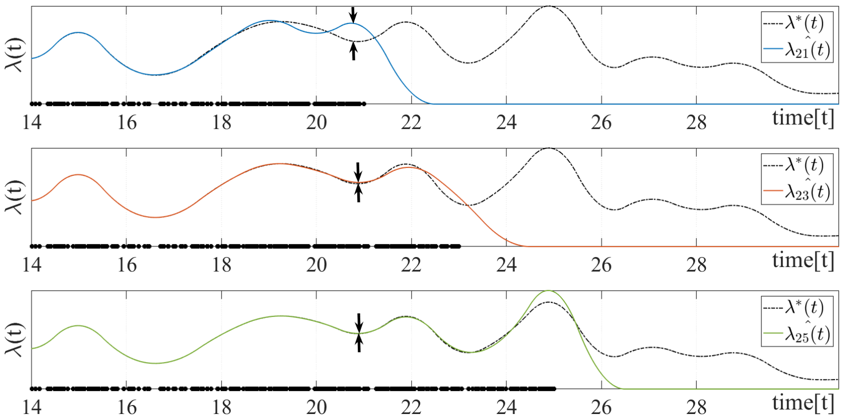


Maximum likelihood, discretized, divided into frames

$$\underset{\{\boldsymbol{x}_t\}}{\text{minimize}} \sum_t f(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t),$$

$$f(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = \langle \boldsymbol{x}_t, \boldsymbol{a}_t \rangle - \langle \boldsymbol{x}_{t-1}, \boldsymbol{b}_t \rangle + \sum_m \log(\langle \boldsymbol{x}_t, \boldsymbol{c}_{m,t} \rangle) + \log(\langle \boldsymbol{x}_{t-1}, \boldsymbol{d}_{m,t} \rangle)$$

Example: Non-homogenous Poisson process



Online Newton algorithm

$$\{\hat{\boldsymbol{x}}_{0|T}, \ldots, \hat{\boldsymbol{x}}_{T|T}\} = \underset{\{\boldsymbol{x}_t\}}{\arg\min} \sum_{t=1}^{T} f_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$$

$$\nabla^2 J_T(\underline{\boldsymbol{x}}) = \begin{bmatrix} \boldsymbol{H}_0 & \boldsymbol{E}_0^{\mathrm{T}} & \mathbf{0} & \cdots & & & \mathbf{0} \\ \boldsymbol{E}_0 & \boldsymbol{H}_1 & \boldsymbol{E}_1^{\mathrm{T}} & \mathbf{0} & \cdots & & \mathbf{0} \\ \mathbf{0} & \boldsymbol{E}_1 & \boldsymbol{H}_2 & \boldsymbol{E}_2^{\mathrm{T}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{E}_2 & \boldsymbol{H}_3 & \boldsymbol{E}_3^{\mathrm{T}} & \cdots & \mathbf{0} \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & & & \boldsymbol{E}_{T-2} & \boldsymbol{H}_{T-1} & \boldsymbol{E}_{T-1}^{\mathrm{T}} \\ \mathbf{0} & \cdots & & & \mathbf{0} & \boldsymbol{E}_{T-1} & \boldsymbol{H}_T \end{bmatrix}$$

General approach: solve with Newton method

- $\boldsymbol{s}_k = -\left(\nabla^2 J_T(\underline{\boldsymbol{x}}_T)\right)^{-1} \nabla J_T(\underline{\boldsymbol{x}}_T)$
- $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k$

The Hessian $\nabla^2 J_T(\underline{\boldsymbol{x}}_T)$ is again *tri-diagonal* ...
... so each Newton step looks like a forward-backward least-squares solve

Finite buffering

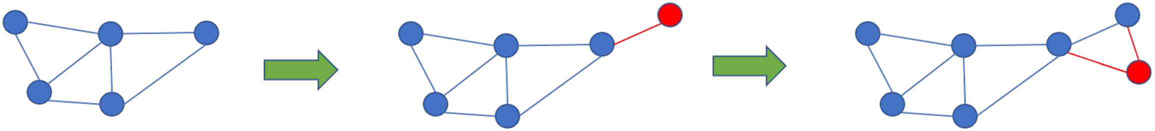
Theorem: If we only update B frames in the past, we have

$$\|\boldsymbol{x}_t^* - \tilde{\boldsymbol{x}}_t^*\| \;\leq\; C\left(\frac{2L-\mu}{2L+\mu}\right)^B$$

where $\tilde{\boldsymbol{x}}_t^*$ are the *buffered solutions* coming from

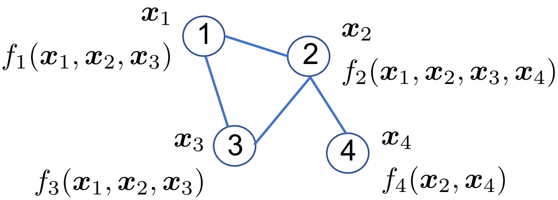
$$\underset{\{\boldsymbol{x}_t,\ldots,\boldsymbol{x}_{t+B+1}\}}{\text{minimize}} \sum_{\tau=t}^{t+B} f_t(\boldsymbol{x}_\tau, \boldsymbol{x}_{\tau+1})$$

Dynamic graph topologies

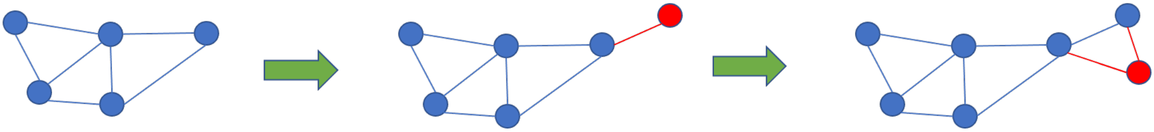


- Nodes i: variables $\boldsymbol{x}_i$ and function f_i
- Edge (i, j): f_i and f_j *share variables*
- Optimization program

$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right)$$



Dynamic graph topologies

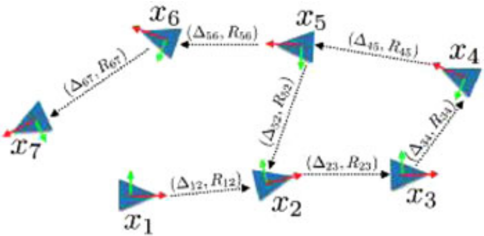


$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right) = \sum_i f_i(\boldsymbol{x}_{[i]})$$

Key question: when we add the red node, do we have to update all other nodes?

Example: Pose graph optimization

- Estimate poses: $x_i = (\text{position}, \text{orientation})$ at time i from relative measurements



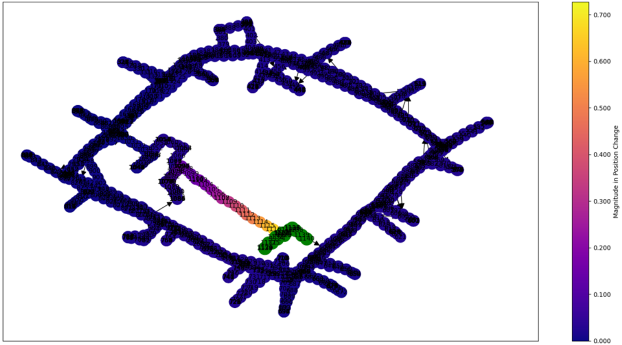
Carlone et al, '16

- Naturally posed as a nonconvex least-squares problem on a dynamic graph
  Semidefinite relaxation is a convex problem on a dynamic graph

Dynamic graph topologies

$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right) = \sum_i f_i(\boldsymbol{x}_{[i]})$$

Key question: when we add a node, do we have to update all other nodes?

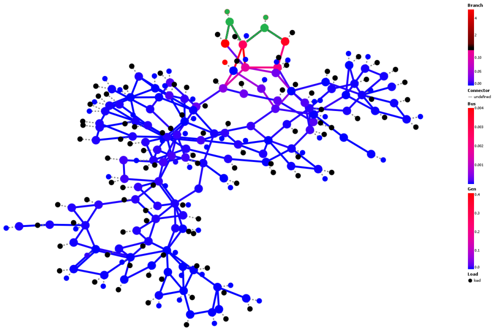


(data from Carlone et al '16)

Dynamic graph topologies

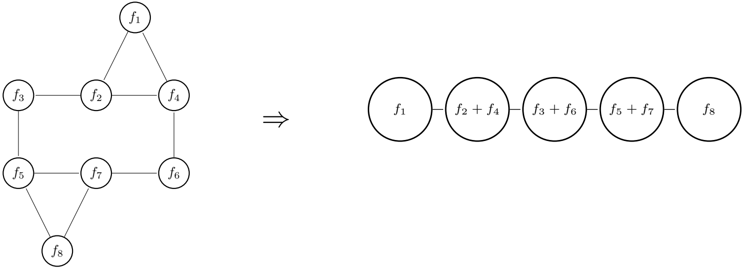
$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right) = \sum_i f_i(\boldsymbol{x}_{[i]})$$

Key question: when we add a node, do we have to update all other nodes?



Collapsing the graph

Key idea: *collapse* the graph between two nodes



Theorem: Difference between solutions at node i before and after node $N+1$ is added

$$\|\hat{\boldsymbol{x}}_{[i]|N} - \hat{\boldsymbol{x}}_{[i]|N+1}\|_2 \ \leq \ \frac{C}{\mu}\left(\frac{L-\mu}{L+\mu}\right)^{d(i,N+1)}$$

where $d(i, N+1)$ = distance between nodes i and $N+1$,
L, μ are Lipschitz and strong convexity constants ...

Collapsing the graph

Theorem: Difference between solutions at node i before and after node $N+1$ is added

$$\|\hat{\boldsymbol{x}}_{[i]|N} - \hat{\boldsymbol{x}}_{[i]|N-1}\|_2 \;\leq\; \frac{C}{\mu}\left(\frac{L-\mu}{L+\mu}\right)^{d(i,N+1)}$$

where $d(i, N+1) =$ distance between nodes i and $N+1$,
L, μ are Lipschitz and strong convexity constants ...

The f_i have Lipschitz gradient parameter L_i, strong convexity parameter μ_i.
We can take

$$\mu = \min_i \mu_i,$$

$$L = K \cdot \max_i L_i, \quad K = \text{ chromatic number of graph}$$

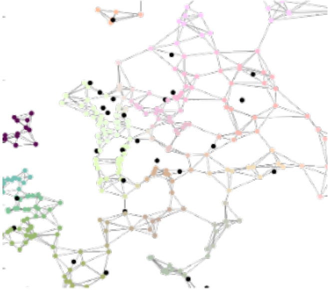
Example: multi-task learning

Solutions of multiple optimization programs are encouraged to be close:

$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \quad \sum_i f_i(\boldsymbol{x}_i) + \lambda \sum_{(j,k)\in\mathcal{E}} w_{jk}\, d(\boldsymbol{x}_j, \boldsymbol{x}_k)$$

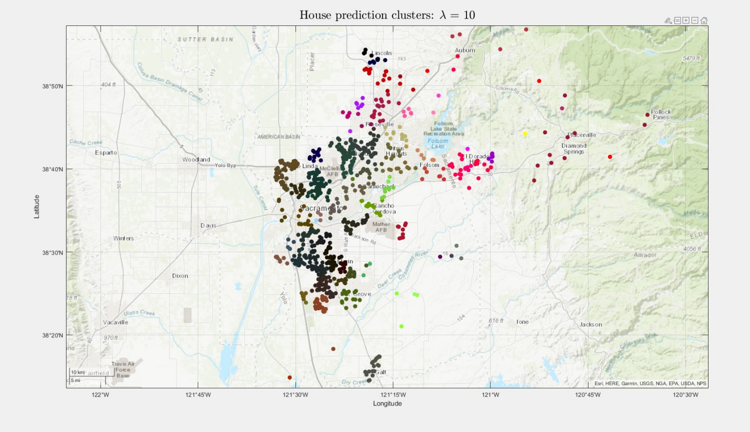
Examples:

- $d(\boldsymbol{x}_j, \boldsymbol{x}_k) = \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2^2$ (diffusion)
- $d(\boldsymbol{x}_j, \boldsymbol{x}_k) = \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2$ (network lasso)
- $\vdots$



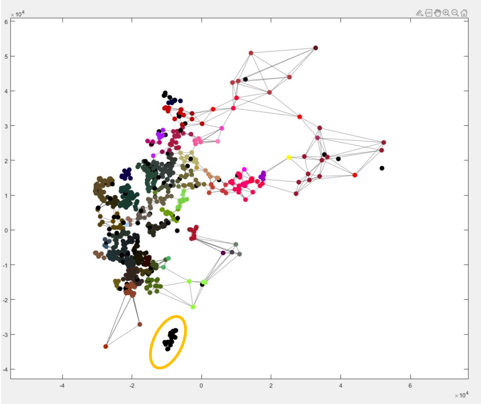
Example: multi-task learning

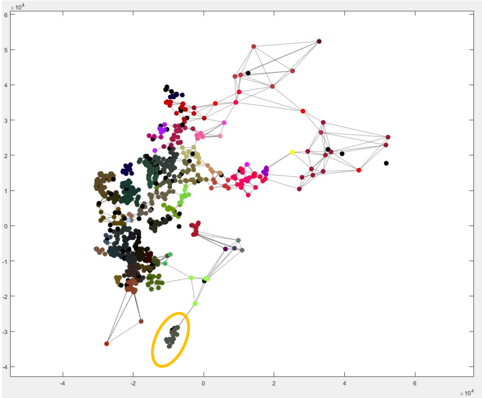
House prices example (Hallac et al. '15)



Example: multi-task learning

House prices example (Hallac et al. '15)

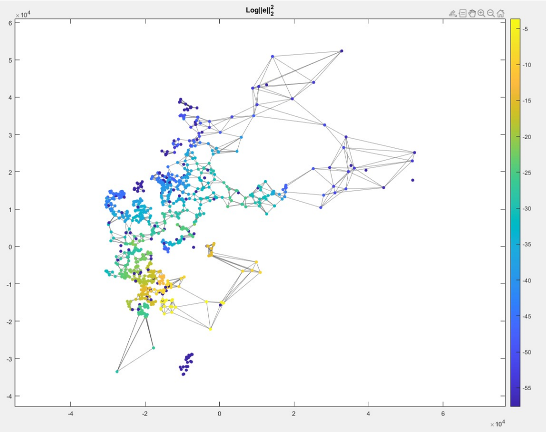




What happens to the solution when the cluster on bottom is added?

Example: multi-task learning

House prices example (Hallac et al. '15)



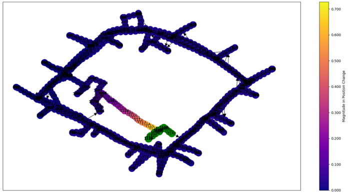
relative change: yellow $= .01$, orange $= 0.001$, blue $= 10^{-9}$

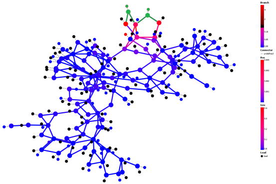
Extension: Constraints

We can accommodate local constraints

$$\underset{\{\boldsymbol{x}_i\}}{\text{minimize}} \quad \sum_i f_i\left(\{\boldsymbol{x}_j : j \in \mathcal{N}(i)\}\right) \quad \text{subject to} \quad \{\boldsymbol{x}_j : j \in \mathcal{N}(i)\} \in \mathcal{C}_i$$

This actually gives us a way to decompose huge SDPs...

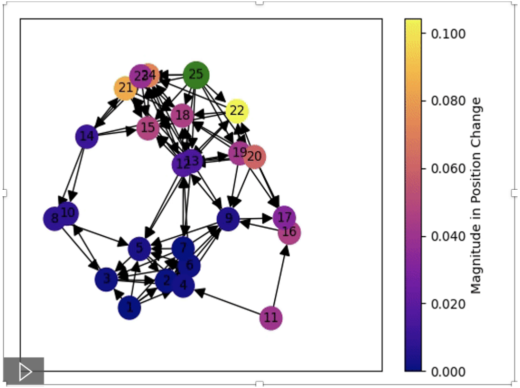




... with small PSD constraints (but have to solve a phase-sync problem)

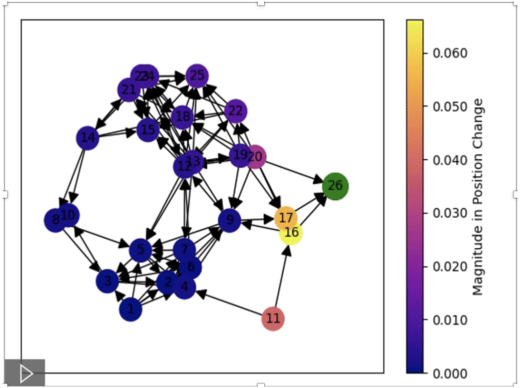
Extension: Growth model

We can get geometric convergence in time if we have a growth model for the graph ...



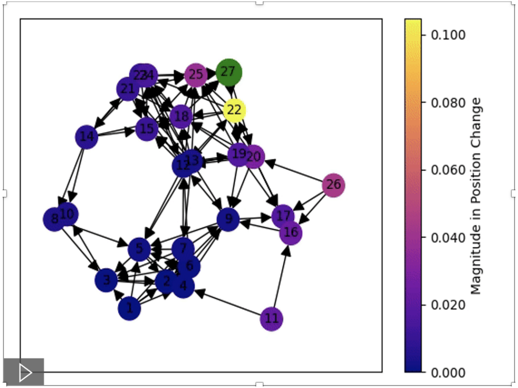
Extension: Growth model

We can get geometric convergence in time if we have a growth model for the graph ...



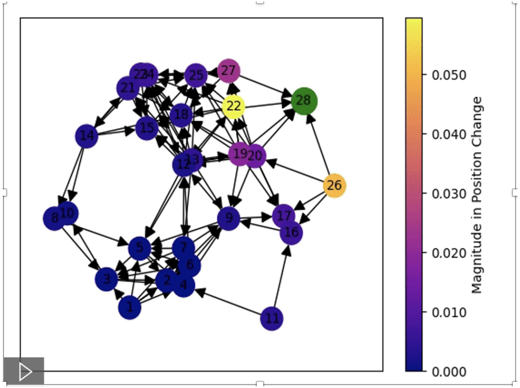
Extension: Growth model

We can get geometric convergence in time if we have a growth model for the graph ...



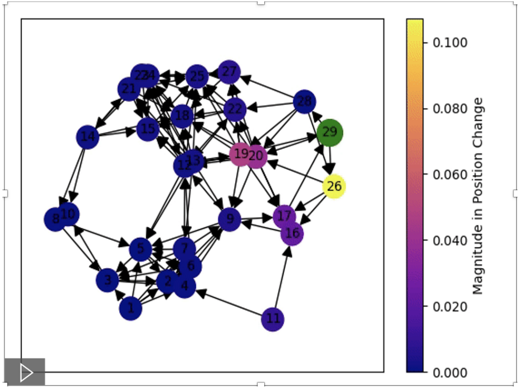
Extension: Growth model

We can get geometric convergence in time if we have a growth model for the graph ...



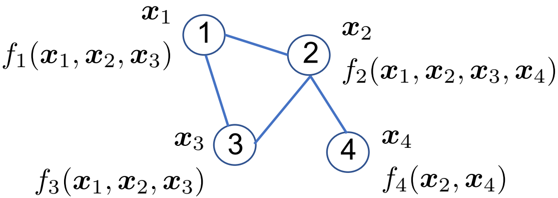
Extension: Growth model

We can get geometric convergence in time if we have a growth model for the graph ...



Closing thoughts

We looked at a very particular type of structured multi-objective optimization problem



Question:
Is there some type of *statistical leverage* we can achieve?

Thank you!

References:

T. Hamam and J. Romberg, "Streaming solutions for time-varying optimization problems," *IEEE Transaction on Signal Processing*, July 2022.

J. Driscoll, T. Hamam and J. Romberg, "Optimization on dynamic graphs," manuscript under preparation.

K. Lee, R. S. Srinivasa, M. Junge, and J. Romberg, "Approximately low-rank recovery from noisy and local measurements by convex programming," *Information and Inference*, 12(3):1612–1654, 2023.