Designing Fast Distributed Iterations via Semidefinite Programming

Lin Xiao and Stephen Boyd

Stanford University

(Joint work with Persi Diaconis and Jun Sun)

5/14/2004

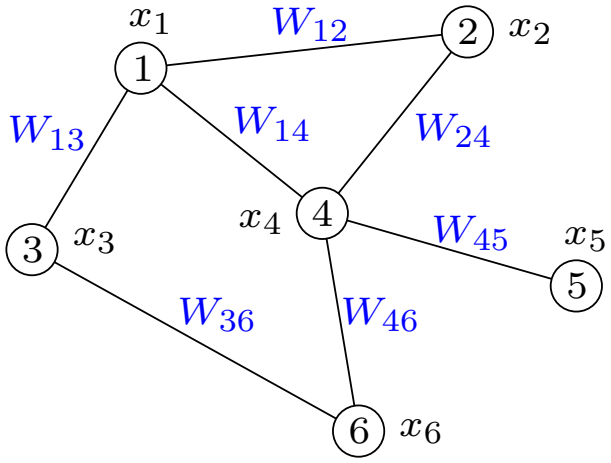
Workshop on Large Scale Nonlinear and Semidefinite Programming, Waterloo

Setting and general problem

- **distributed process:** communication, computation, flow constrained by given graph
- **examples:** distributed consensus, distributed resource allocation, distributed estimation, Markov chains, coordination/control of autonomous agents, iterative solution of equations, . . .
- **weights** on edges affect convergence behavior
- simple results known ($e.g.$, convergence with small, positive weights)

how do we choose weights to yield fastest possible convergence?

Example: distributed average consensus



- compute average $\bar{x} = \frac{1}{n}\sum_i x_i$ (using local communication, iteration)
- each node takes a weighted average of its own and neighbors' values:

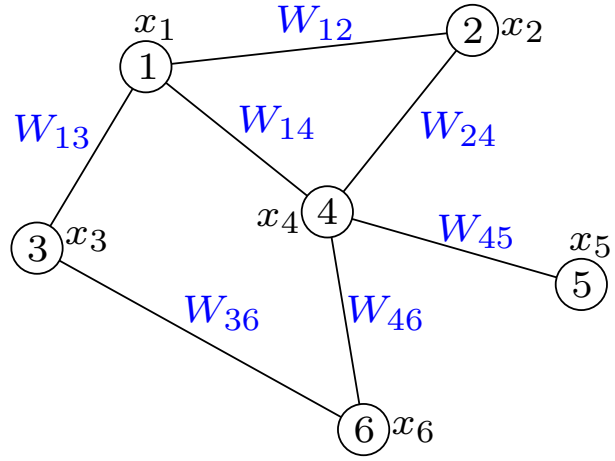
$$x_i(t+1) = W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t)$$

- **how do we choose W to make convergence as fast as possible?**

Example: distributed resource allocation

- resource allocation on a network

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i = c \end{array}$$



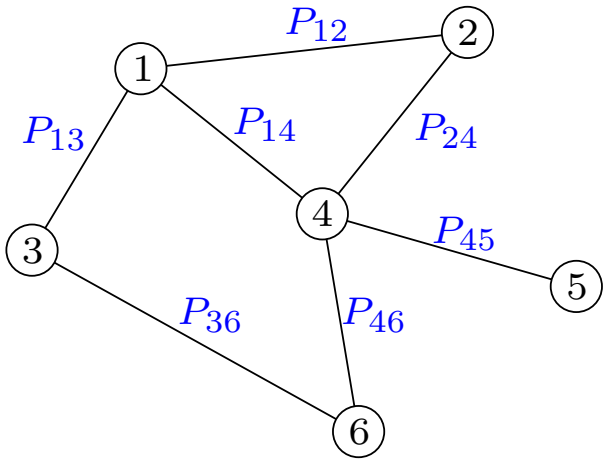
- distributed weighted gradient method:

$$x_i(t+1) = x_i(t) - \sum_{j \in \mathcal{N}_i} W_{ij} \left(f_i'(x_i(t)) - f_j'(x_j(t)) \right)$$

(exchange resources proportional to differences of marginal costs)

- **how do we choose W to make convergence as fast as possible?**

Example: Markov chain on a graph



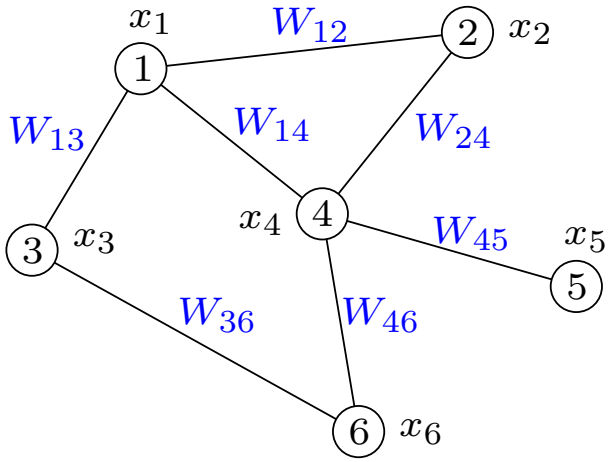
- random walk on graph with symmetric transition probabilities P_{ij}
- (under simple conditions) distribution converges to uniform
- **what edge transition probabilities give fastest mixing?**

Typical results

- using SDP, we can optimize convergence rate (or a bound on it)
- by exploiting structure, associated SDPs can be efficiently solved
- SDP duality yields bounds, insight, . . .

Fast distributed average consensus

Distributed average consensus



- compute average $\bar{x} = \frac{1}{n}\sum_i x_i$ (using local communication, iteration)
- each node takes a weighted average of its own and neighbors' values:

$$x_i(t+1) = W_{ii}x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}x_j(t)$$

- vector form: $x(t+1) = Wx(t)$; W has sparsity pattern constraint given by graph

Convergence conditions and rate

- convergence $\iff \lim_{t\to\infty} W^t = \mathbf{1}\mathbf{1}^T/n \iff$

$$\mathbf{1}^T W = \mathbf{1}^T, \qquad W\mathbf{1} = \mathbf{1}, \qquad \rho(W - \mathbf{1}\mathbf{1}^T/n) < 1$$

  - sum (and therefore average) preserved at each step
  - $\mathbf{1}$ is fixed point of iteration $x(t+1) = Wx(t)$
  - iteration dynamics are stable on $\mathbf{1}^\perp$

- asymptotic convergence rate given by $\rho(W - \mathbf{1}\mathbf{1}^T/n)$

- for symmetric W, same as $\|W - \mathbf{1}\mathbf{1}^T/n\|$

Fastest distributed linear averaging

$$\begin{array}{ll} \text{minimize} & \rho(W - \mathbf{1}\mathbf{1}^T/n) \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^T W = \mathbf{1}^T, \quad W\mathbf{1} = \mathbf{1} \end{array}$$

optimization variable is W; problem data is graph (sparsity pattern $\mathcal{S}$)

- hard problem when W is not symmetric
- can minimize convex upper bound $\|W - \mathbf{1}\mathbf{1}^T/n\|$
- for symmetric W, these two coincide

Semidefinite programming formulation

(for symmetric weights)

introduce scalar variable s to bound spectral norm

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preceq W - \mathbf{1}\mathbf{1}^T/n \preceq sI \\ & W \in \mathcal{S}, \quad W = W^T, \quad W\mathbf{1} = \mathbf{1} \end{array}$$

an SDP, hence, efficiently solved, duality theory, . . .

can also pose problem of minimizing $\|W - \mathbf{1}\mathbf{1}^T/n\|$, with nonsymmetric W, as SDP

Constant weights

constant weight on all edges:

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i} \alpha(x_j(t) - x_i(t))$$

- maximum-degree weight: $\alpha = 1/\max_i d_i$
  d_i is degree (number of neighbors) of node i

- best constant weight: $\alpha^\star = 2/(\lambda_1(L) + \lambda_{n-1}(L))$
  L is **Laplacian** of graph; $L = \mathbf{diag}(d) - A$, A is adjacency matrix

- sometimes give reasonably fast convergence

Metropolis weights

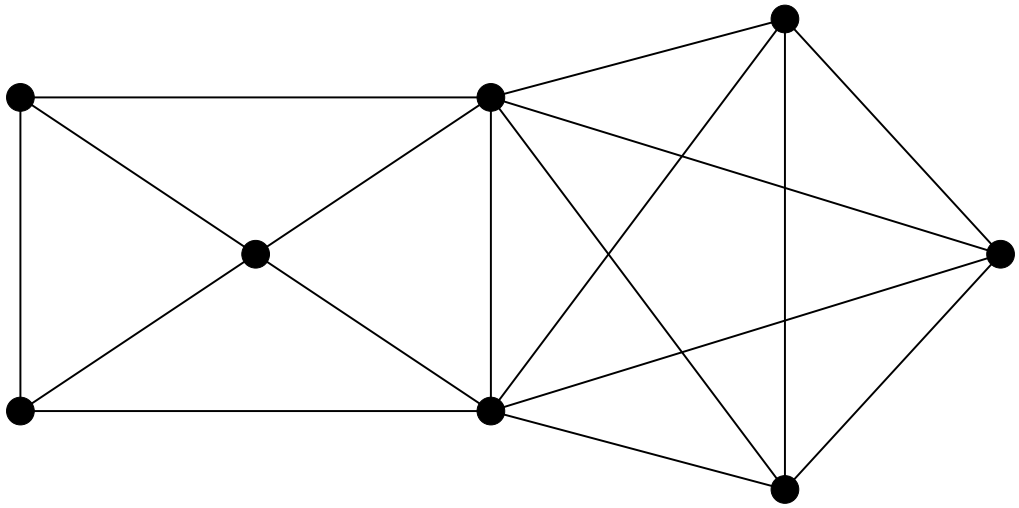
Metropolis-Hastings weights:

$$W_{ij} = \frac{1}{\max\{d_i, d_j\}}, \qquad \{i,j\} \in \mathcal{E}$$

(self-weights given by $W_{ii} = 1 - \sum_{j \in \mathcal{N}_i} W_{ij}$)

- adapted from Metropolis algorithms in Markov chain Monte Carlo
- Metropolis weights based on *local* information
- often gives reasonable convergence

A small example

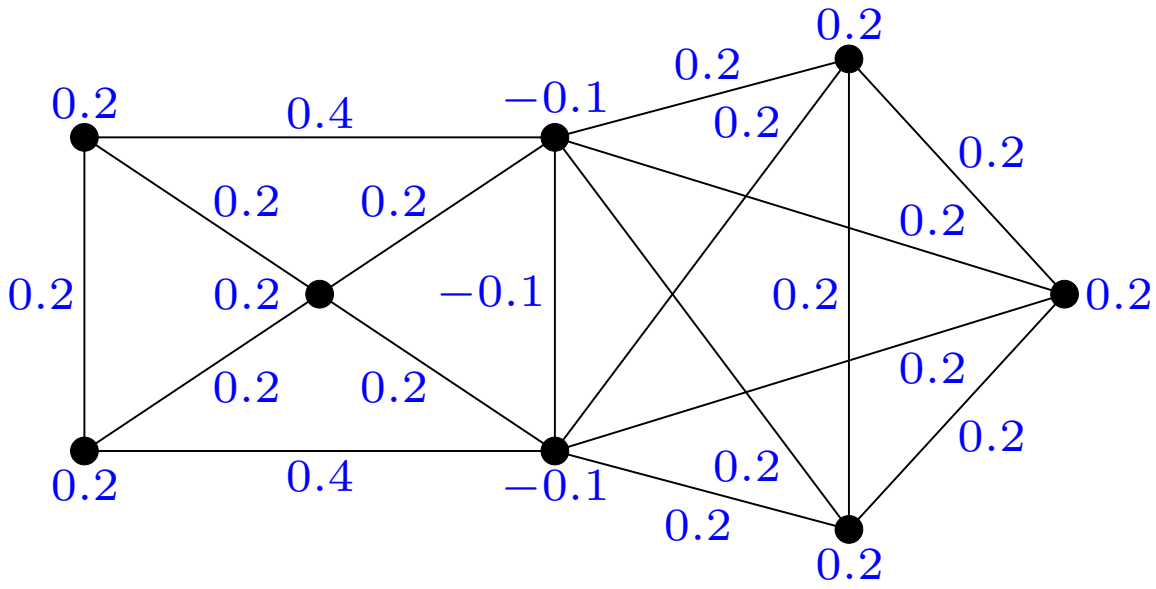


convergence factors and convergence times:

	max degree	Metropolis	optimal symm.
$\rho(W - \mathbf{1}\mathbf{1}^T/n)$	0.746	0.743	0.600
$\tau = 1/\log(1/\rho)$	3.413	3.366	1.958

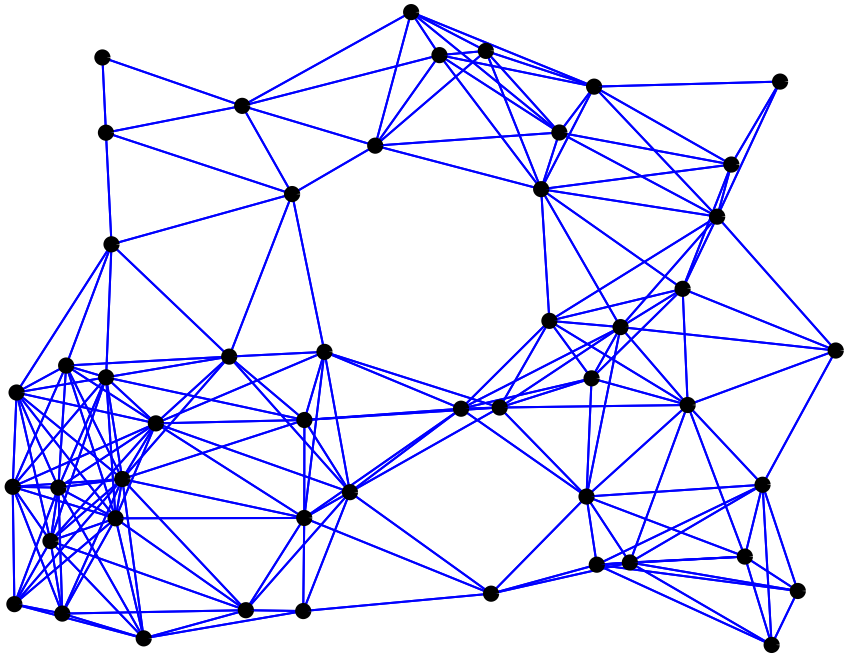
Optimal symmetric weights

(note: some weights are negative!)



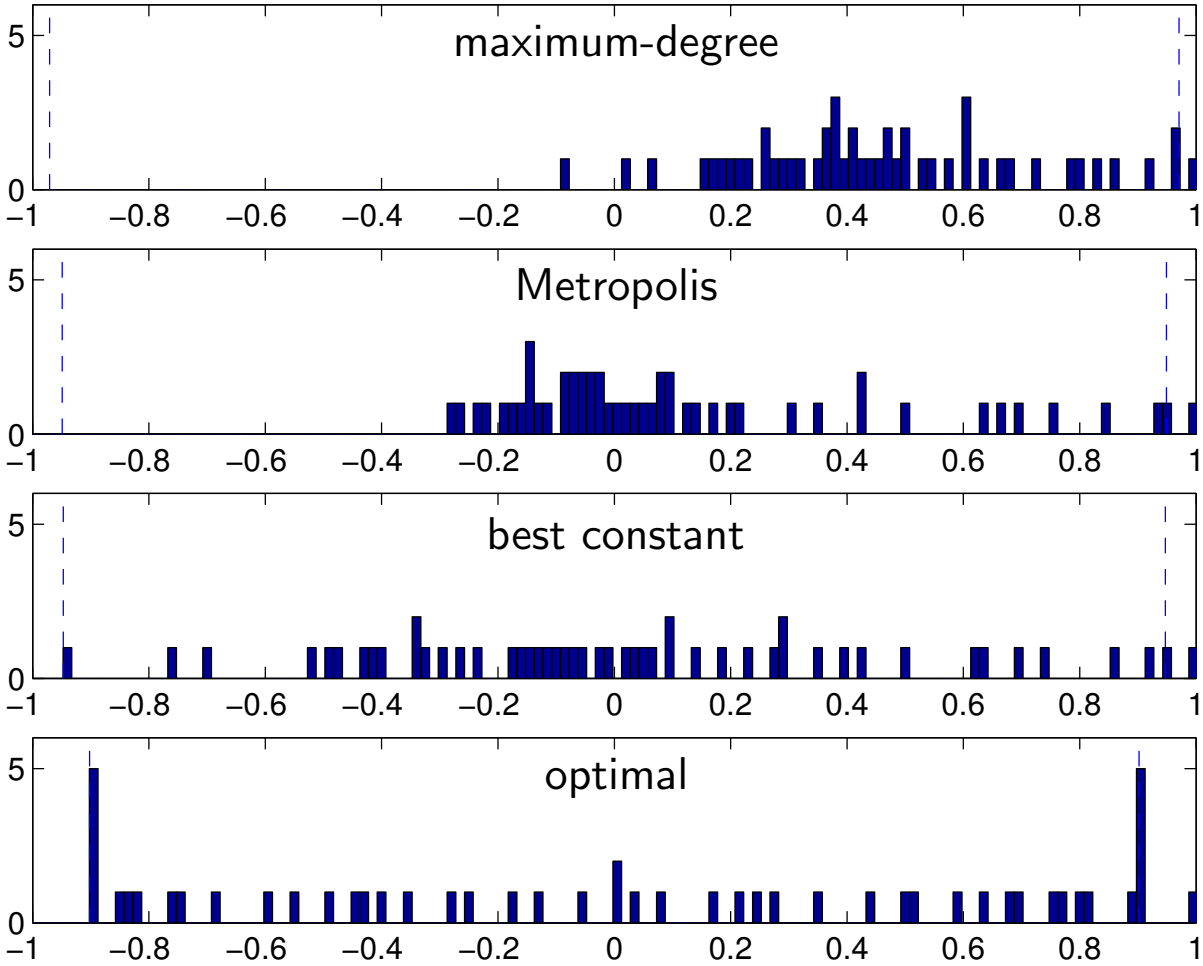
A larger example

50 nodes, 200 edges

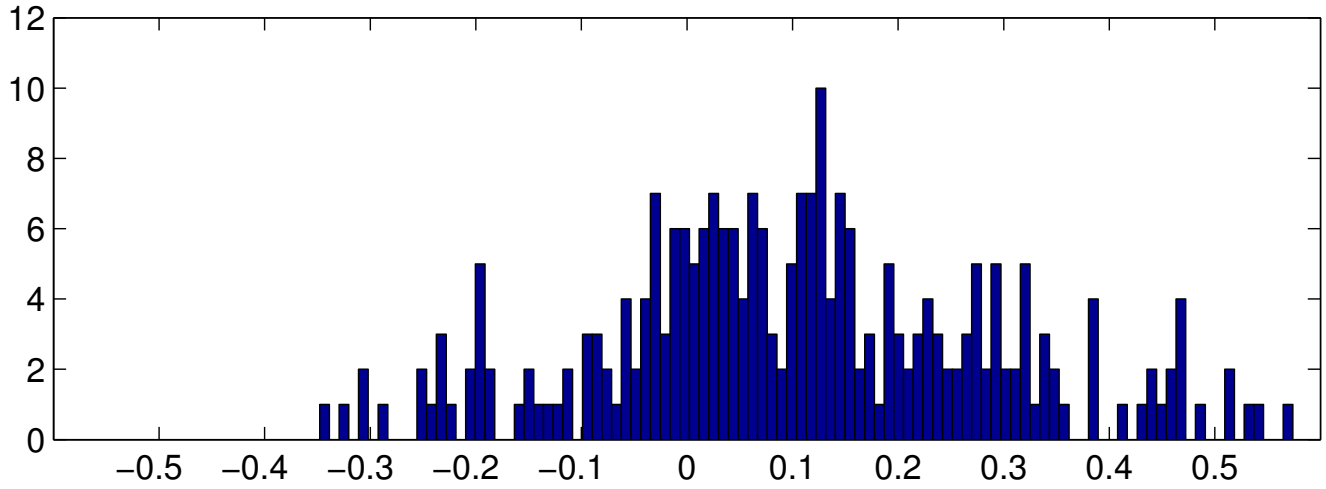


	max degree	Metropolis	best constant	optimal
$\rho(W - \mathbf{1}\mathbf{1}^T/n)$	0.971	0.949	0.947	0.902
$\tau = 1/\log(1/\rho)$	33.980	19.104	18.363	9.696

Eigenvalue distributions



Optimal weights



Application: Data fusion in sensor networks

- estimate a vector of unknown, fixed parameters $x \in \mathbf{R}^m$
- n sensors; each makes noisy measurement $y_i = A_i x + v_i \in \mathbf{R}^{m_i}$
  independent noises v_i have zero mean, covariance Σ_i
- aggregate measurement

$$y = Ax + v = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} x + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

  v has covariance $\Sigma = \mathbf{diag}(\Sigma_1, \ldots, \Sigma_n)$

- maximum likelihood estimate of x is weighted least-squares (WLS) solution

$$\begin{aligned} x_{\mathrm{wls}} &= \left(A^T\Sigma^{-1}A\right)^{-1}A^T\Sigma^{-1}y \\ &= \left(\sum_{i=1}^{n} A_i^T\Sigma_i^{-1}A_i\right)^{-1}\sum_{i=1}^{n} A_i^T\Sigma_i^{-1}y_i \end{aligned}$$

- centralized data fusion: fusion center collects all measurements, computes WLS solution

A simple distributed scheme for sensor fusion

- each sensor initializes

$$P_i(0) = A_i^T \Sigma_i^{-1} A_i, \qquad q_i(0) = A_i^T \Sigma_i^{-1} y_i$$

- use distributed average consensus to compute (entrywise)

$$P = \frac{1}{n} \sum_{i=1}^{n} A_i^T \Sigma_i^{-1} A_i, \qquad q = \frac{1}{n} \sum_{i=1}^{n} A_i^T \Sigma_i^{-1} y_i$$

- then locally compute the WLS estimate $x_{\mathrm{wls}} = P^{-1} q$

- Metropolis weights yield simple, isotropic protocol for sensor nodes

Fastest mixing Markov chain on a graph

Markov chain on a graph

- random walk on connected graph $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$

$$\mathcal{V} = \{1, \ldots, n\}, \qquad \mathcal{E} = \{(i,j) \mid i \text{ and } j \text{ connected}\}$$

  we'll assume each vertex has self-loop, *i.e.*, $(i,i) \in \mathcal{E}$

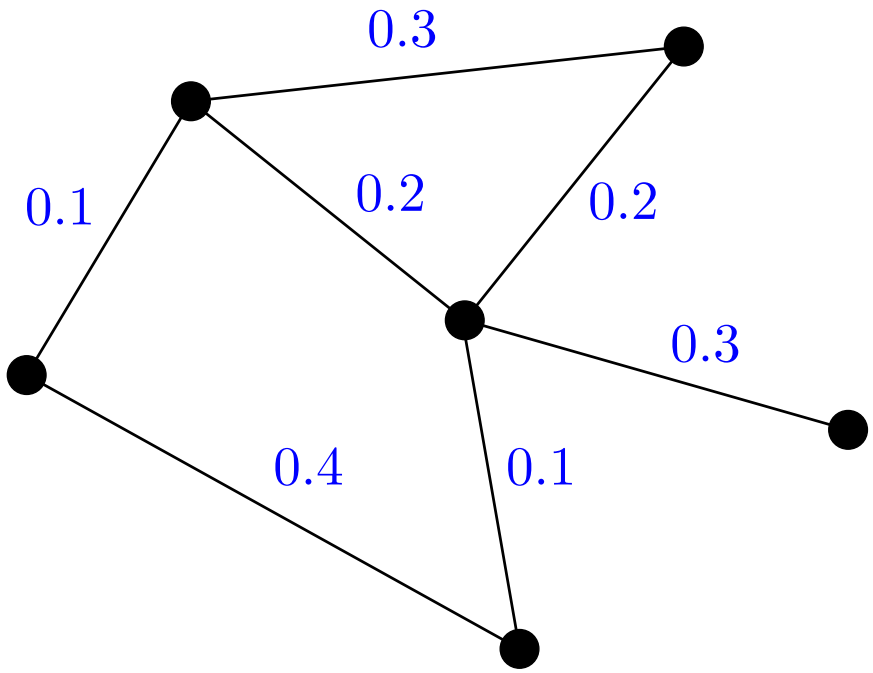
- define Markov chain on vertices $X(t) \in \{1, \ldots, n\}$, with transition probabilities on edges

$$P_{ij} = \mathbf{Prob}\,(X(t+1) = j \mid X(t) = i)$$

  we'll focus on symmetric transition probability matrices P

- all results can be extended to reversible Markov chains

Example



self-loop transition probabilities not shown, given by

$$P_{ii} = 1 - \sum_{j \neq i} P_{ij}$$

Stationary distribution

- probability distribution $\pi_i(t) = \mathbf{Prob}(X(t) = i)$ satisfies $\pi(t+1)^T = \pi(t)^T P$

- since $P = P^T$ and $P\mathbf{1} = \mathbf{1}$, uniform distribution $\pi = \mathbf{1}/n$ is stationary, *i.e.*, $(\mathbf{1}^T/n)P = \mathbf{1}^T/n$

- (assuming irreducible, aperiodic)

$$\lim_{t\to\infty} \|\pi(t) - \mathbf{1}/n\| = 0$$

  i.e., distribution converges to uniform

Second largest eigenvalue modulus (SLEM)

- since $P = P^T$, all eigenvalues are real; can order as

$$1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq -1$$

- second largest eigenvalue modulus (SLEM):

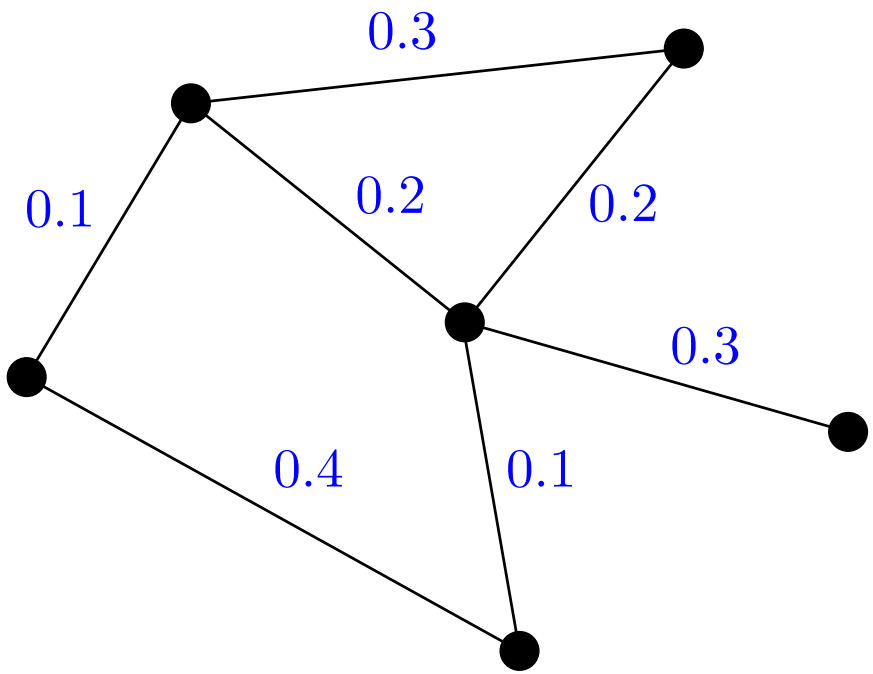
$$\mu(P) = \max_{i=2,\ldots,n} |\lambda_i(P)| = \max\{\lambda_2(P), -\lambda_n(P)\}$$

- asymptotic rate of convergence to $\pi_{\mathrm{st}} = \mathbf{1}/n$ determined by SLEM, $e.g.$,

$$\sup_{\pi(0)} \|\pi(t) - \mathbf{1}/n\|_{\mathrm{tv}} \leq \left(\sqrt{n}/2\right) \mu^t$$

- associated mixing time is $\tau = 1/\log(1/\mu)$

Example



$$\mu = 0.86 \qquad \tau = \frac{1}{\log(1/\mu)} = 6.84$$

Convexity of mixing rate

$\mu(P)$ is **convex function** of P

$\mu(P)$ is spectral norm of P on $\mathbf{1}^\perp = \{v \mid \mathbf{1}^T v = 0\}$:

$$\begin{aligned} \mu(P) &= \left\| \left(I - (1/n)\mathbf{1}\mathbf{1}^T\right) P \left(I - (1/n)\mathbf{1}\mathbf{1}^T\right) \right\|_2 \\ &= \left\| P - (1/n)\mathbf{1}\mathbf{1}^T \right\|_2 \end{aligned}$$

$\left(I - (1/n)\mathbf{1}\mathbf{1}^T\right)$ is projection matrix onto subspace $\mathbf{1}^\perp$

another proof:

- for general symmetric X, $\lambda_1(X) + \lambda_2(X)$ and $-\lambda_n(X)$ are convex
- here $\lambda_1 = 1$, so $\max\{\lambda_2(X), -\lambda_n(X)\}$ is convex

Fastest mixing Markov chain (FMMC) problem

$$\begin{array}{ll} \text{minimize} & \mu(P) = \left\| P - (1/n)\mathbf{1}\mathbf{1}^T \right\|_2 \\ \text{subject to} & P \geq 0, \quad P\mathbf{1} = \mathbf{1}, \quad P = P^T \\ & P_{ij} = 0, \quad (i,j) \notin \mathcal{E} \end{array}$$

- variable is matrix P; problem data is graph
- convex optimization problem, hence efficiently solved, duality theory, . . .

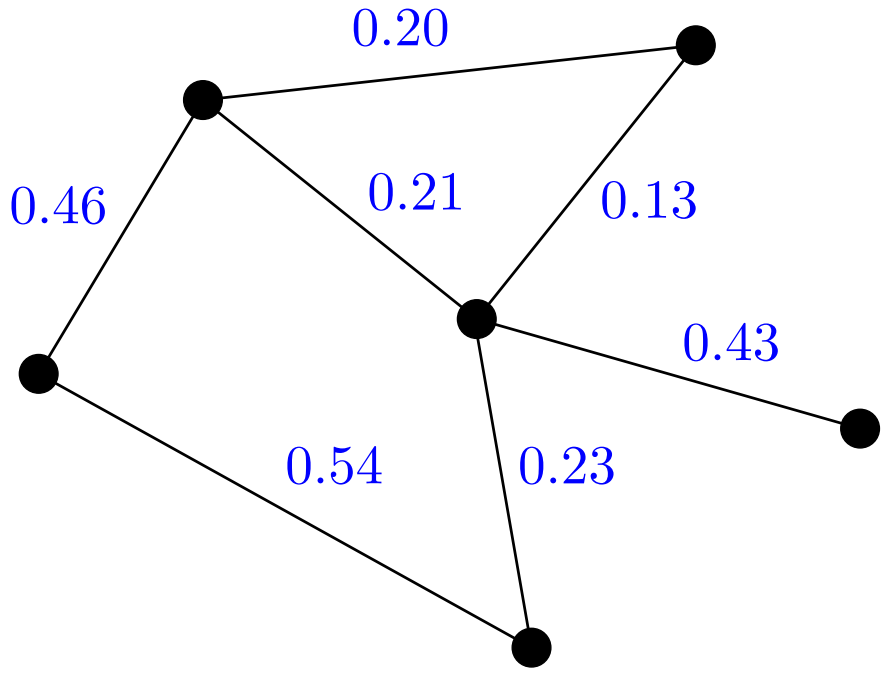
SDP formulation of FMMC

introduce scalar variable s to bound norm of $P - (1/n)\mathbf{1}\mathbf{1}^T$

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preceq P - (1/n)\mathbf{1}\mathbf{1}^T \preceq sI \\ & P \geq 0, \quad P\mathbf{1} = \mathbf{1}, \quad P = P^T \\ & P_{ij} = 0, \quad (i,j) \notin \mathcal{E} \end{array}$$

an SDP in variables P, s

Example



$$\mu^\star = 0.72 \qquad \tau^\star = \frac{1}{\log(1/\mu^\star)} = 3.06$$

Two common suboptimal schemes

let d_i be degree of vertex i (not counting self-loops)

- maximum degree chain: with $d_{\max} = \max_{i \in \mathcal{V}} d_i$

$$P_{ij}^{\mathrm{md}} = \frac{1}{d_{\max}}, \qquad (i,j) \in \mathcal{E}, \quad i \neq j,$$

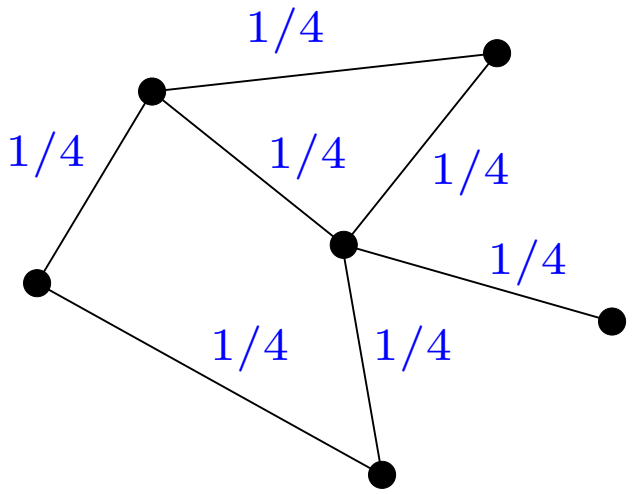
- Metropolis-Hastings chain

$$P_{ij}^{\mathrm{mh}} = \frac{1}{\max\{d_i, d_j\}}, \qquad (i,j) \in \mathcal{E}, \quad i \neq j$$

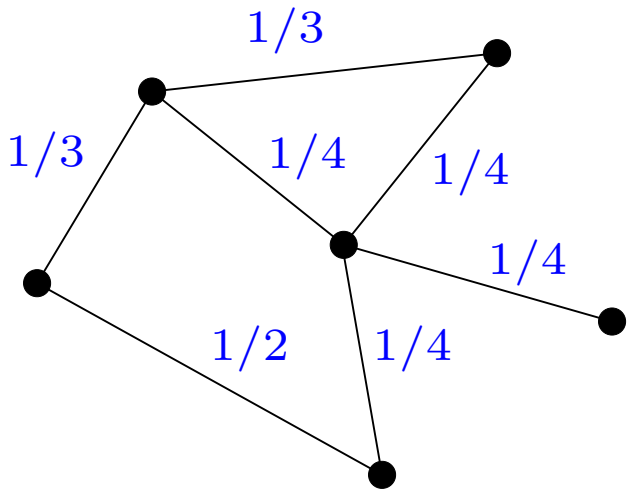
diagonal entries determined by $P_{ii} = 1 - \sum_{j \neq i} P_{ij}$

Example

max-degree



Metropolis-Hastings



$$\mu^{\mathrm{md}} = 0.78$$
$$\tau^{\mathrm{md}} = 4.02$$

$$\mu^{\mathrm{mh}} = 0.77$$
$$\tau^{\mathrm{mh}} = 3.91$$

Fastest mixing to nonuniform distribution

- given desired equilibrium distribution $\pi = (\pi_1, \ldots, \pi_n)$
- we consider P with same sparsity pattern as graph, but not symmetric
- we require **reversible** chain: $\pi_i P_{ij} = \pi_j P_{ji}$, the *detailed balance equation*
- detailed balance is equivalent to $\Pi P = P^T \Pi$, where $\Pi = \mathbf{diag}(\pi)$
- the matrix $\Pi^{1/2} P \Pi^{-1/2}$ is symmetric, with same eigenvalues as P
- eigenvector of $\Pi^{1/2} P \Pi^{-1/2}$ associated with eigenvalue one is

$$q = (\sqrt{\pi_1}, \ldots, \sqrt{\pi_n})$$

- asymptotic rate of convergence of distribution to π determined by

$$\mu(P) = \left\| \Pi^{1/2} P \Pi^{-1/2} - q q^T \right\|_2$$

  which is convex in P

- SDP formulation of fastest mixing reversible Markov chain:

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preceq \Pi^{1/2} P \Pi^{-1/2} - q q^T \preceq sI \\ & P \geq 0, \quad P\mathbf{1} = \mathbf{1}, \quad \Pi P = P^T \Pi \\ & P_{ij} = 0, \quad (i,j) \notin \mathcal{E} \end{array}$$

  variables are s, P; problem data are π and graph

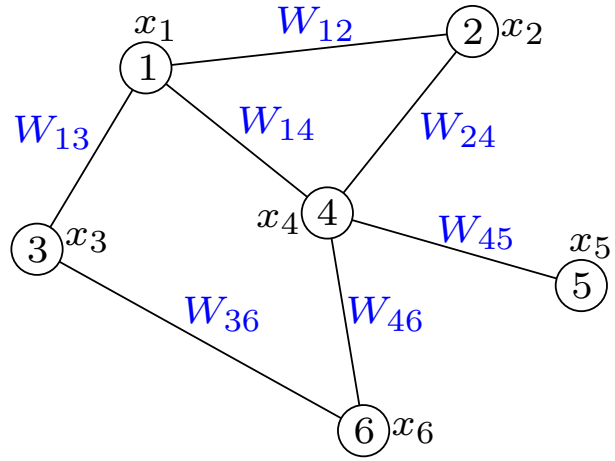
Fast distributed resource allocation

Distributed resource allocation

- resource allocation on a network

  minimize $\sum_{i=1}^n f_i(x_i)$

  subject to $\sum_{i=1}^n x_i = c$



- distributed weighted gradient method:

$$x_i(t+1) = x_i(t) - \sum_{j \in \mathcal{N}_i} W_{ij} \left(f_i'(x_i(t)) - f_j'(x_j(t)) \right)$$

  exchange resources proportional to differences of marginal costs

- **how do we choose W to make convergence as fast as possible?**

Guaranteed convergence rate

- weighted gradient update: $x(t+1) = x(t) - W\nabla f(x)$
- must have $\mathbf{1}^T W = 0$, $W\mathbf{1} = 0$
- can show

$$f(x(t)) - f^\star \leq \eta^t (f(x(0)) - f^\star)$$

where

$$\eta = 1 - \lambda_{n-1}\left(L^{1/2}(W + W^T - W^T U W)L^{1/2}\right)$$

$L = \mathbf{diag}(l_1, \ldots, l_n)$, $U = \mathbf{diag}(u_1, \ldots, u_n)$, $l_i \leq f_i''(x_i) \leq u_i$

- hence, η gives guaranteed convergence rate

Optimal guaranteed convergence rate

optimize guaranteed convergence rate:

$$\begin{array}{ll} \text{maximize} & \lambda_{n-1}\left(L^{1/2}(W+W^T-W^TUW)L^{1/2}\right) \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^TW = 0, \quad W\mathbf{1} = 0 \end{array}$$

(can impose $W = W^T$ or not)

. . . can show this is convex problem; can formulate as SDP

$$\begin{array}{ll} \text{maximize} & s \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^TW = 0, \quad W\mathbf{1} = 0 \\ & \left[\begin{array}{cc} W+W^T - s\left(L^{-1} - (1/\mathbf{1}^TL^{-1}\mathbf{1})L^{-1}\mathbf{1}\mathbf{1}^TL^{-1}\right) & W^T \\ W & U^{-1} \end{array}\right] \succeq 0 \end{array}$$

Simple weight selection methods

- constant weight on all edges: $W_{ij} = \alpha$ for $(i,j) \in \mathcal{E}$, $i \neq j$

  self-weights given by $W_{ii} = -\sum_{j \in \mathcal{N}_i} W_{ij}$

- **max-degree weights:** $\alpha = -1/(\max_{i \in \mathcal{N}} d_i u_i)$

- **Metropolis weights:**

$$W_{ij} = -\min\left\{\frac{1}{d_i u_i}, \frac{1}{d_j u_j}\right\}, \quad i \neq j, \quad (i,j) \in \mathcal{E}$$

  self-weights given by $W_{ii} = -\sum_{j \in \mathcal{N}_i} W_{ij}$

Example

- $f_i(x_i) = \frac{1}{2}a_i(x_i - c_i)^2 + \log\left(1 + e^{b_i(x_i - d_i)}\right)$

  coefficients $a_i \geq 0,\ b_i,\ c_i,\ d_i$ generated randomly

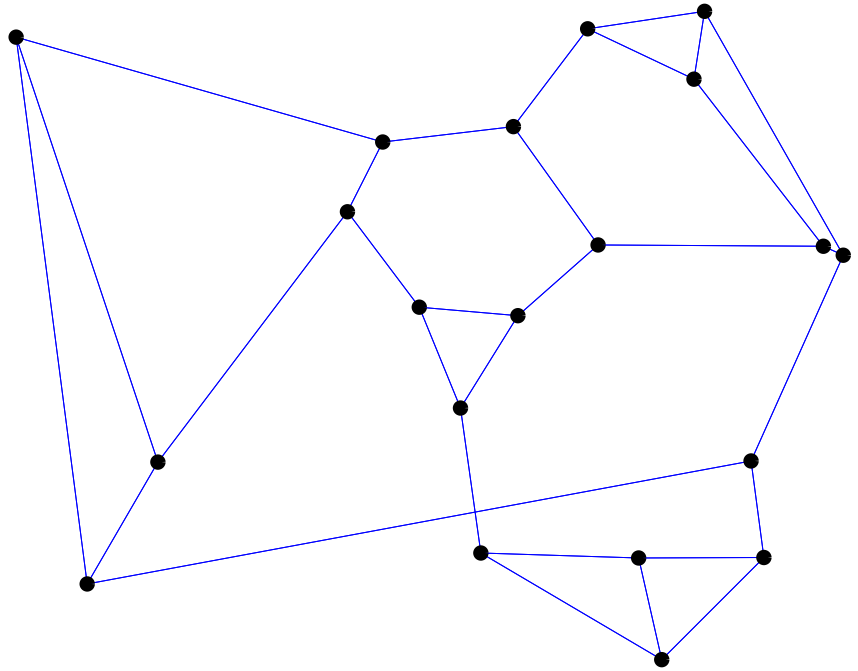
- bounds on second derivatives:

$$l_i = a_i \leq f_i''(x_i) \leq a_i + \frac{1}{4}b_i^2 = u_i$$

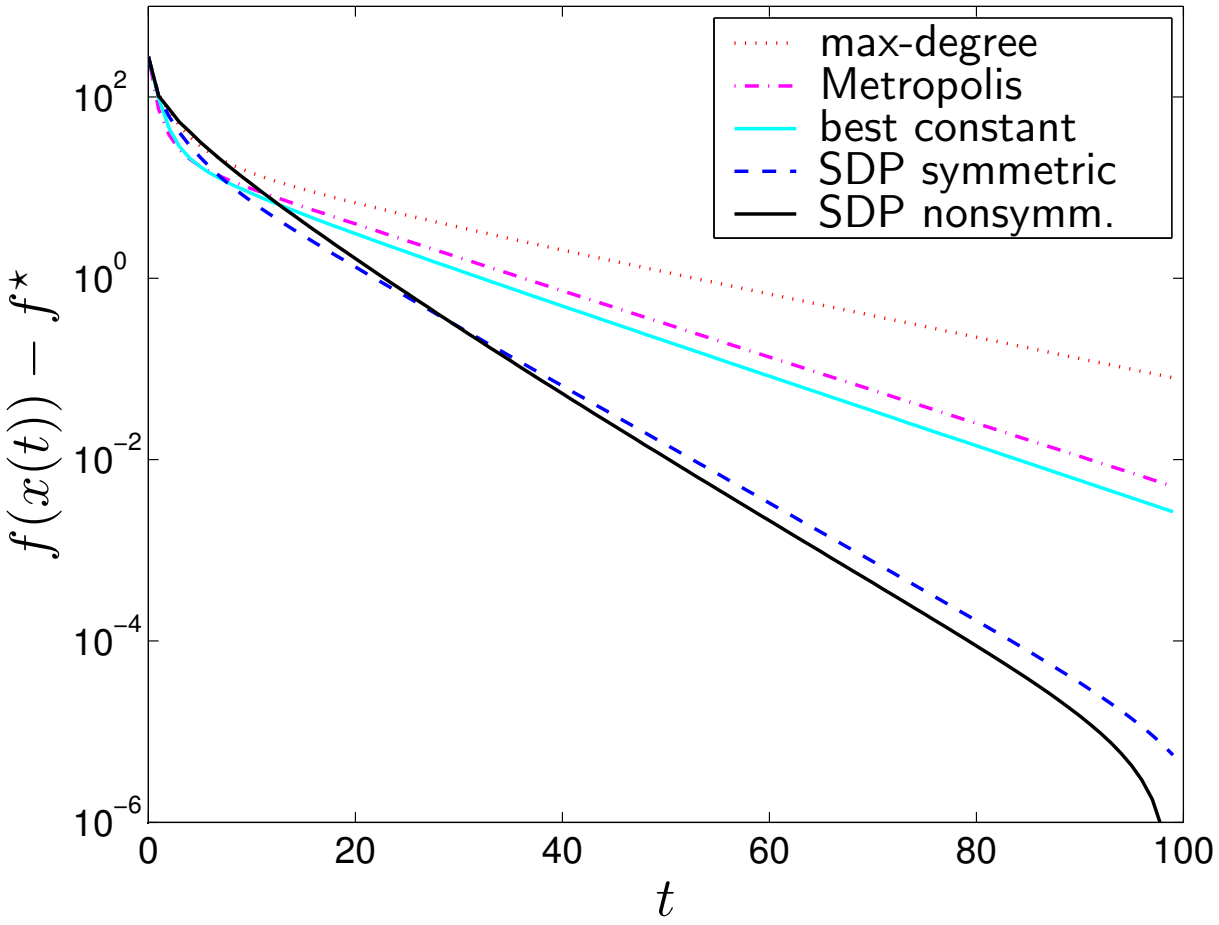
- resource constraint $\mathbf{1}^T x = 0$

- randomly generated regular graph with 20 nodes, degree 3

Example



	max-degree	Metropolis	best constant	SDP symm.	SDP nonsymm.
η	0.950	0.924	0.922	0.875	0.873



- (in this case) η predicts the convergence rate well
- this is frequently, but not always, the case

Computational methods

Computational methods

- **interior-point methods:**
  - exploit sparsity and graph structure
  - can solve problems with a few thousand edges
- **subgradient methods:**
  - compute subgradient efficiently with Lanczos method
  - can solve problems with 10^6 edges

Exploiting structure in interior-point methods

- consider interior-point method for solving SDP

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preceq W - (1/n)\mathbf{1}\mathbf{1}^T \preceq sI \\ & W\mathbf{1} = \mathbf{1}, \quad W = W^T, \quad W \in \mathcal{S} \end{array}$$

- forming search direction equations involves frequent computing of

$$(sI - W + \mathbf{1}\mathbf{1}^T/n)^{-1}, \qquad (sI + W - \mathbf{1}\mathbf{1}^T/n)^{-1}$$

- can efficiently evaluate, exploiting sparse + rank-one structure
- still have to solve dense $m \times m$ system to find search direction
- order m^3

Weighted Laplacian formulation

we'll consider averaging problem with symmetric weights

$$\begin{array}{ll} \text{minimize} & \|W - \mathbf{1}\mathbf{1}^T/n\| \\ \text{subject to} & W \in \mathcal{S}, \quad W = W^T, \quad W\mathbf{1} = \mathbf{1} \end{array}$$

other problems a little more complicated, but similar

we'll use weighted Laplacian formulation

$$\text{minimize} \quad \phi(w) = \|I - A\,\mathbf{diag}(w)A^T - \mathbf{1}\mathbf{1}^T/n\|$$

with $w \in \mathbf{R}^m$ (vector of edge weights); A is (node-edge) incidence matrix

Subgradient method

for $k = 1, 2, \ldots$

compute a subgradient $g \in \partial\phi(w)$

update weights: $w := w - \alpha_k g$

- subgradient (by definition) satisfies

$$\phi(z) \geq \phi(w) + g^T(z - w) \quad \text{for all } z$$

- step lengths satisfy *diminishing rule*:

$$\alpha_k \geq 0, \qquad \lim_{k\to\infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

Subgradient of ϕ

- $Z = W - \mathbf{1}\mathbf{1}^T/n = I - A\,\mathbf{diag}(w)A^T - \mathbf{1}\mathbf{1}^T/n$
- if $\|Z\| = \lambda_1(Z)$, $Zu = \lambda_1(Z)u$, $\|u\| = 1$, then a subgradient is

$$g_{(i,j)} = -(u_i - u_j)^2, \qquad (i,j) \in \mathcal{E}$$

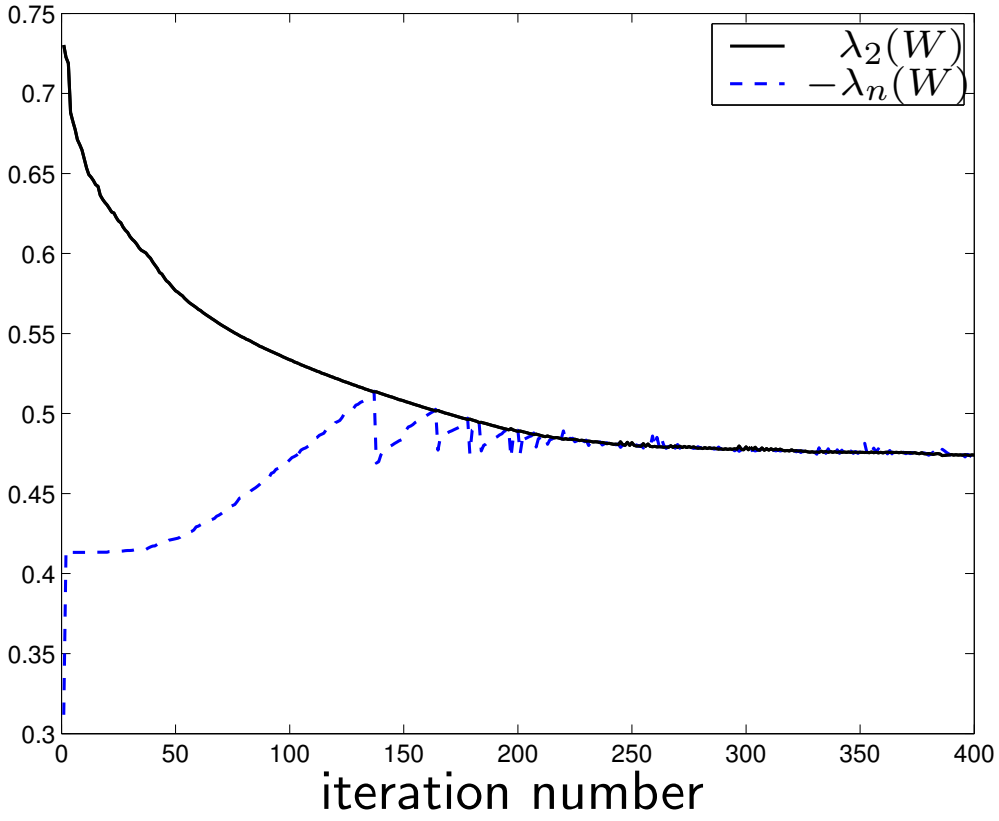
- if $\|Z\| = -\lambda_n(Z)$, $Zu = \lambda_n(Z)u$, $\|u\| = 1$, then a subgradient is

$$g_{(i,j)} = (u_i - u_j)^2, \qquad (i,j) \in \mathcal{E}$$

- can compute $\lambda_1(Z)$, $\lambda_n(Z)$, associated eigenvectors very efficiently by Lanczos method

Example

- random graph with 10^4 vertices, 10^5 edges
- step size $\alpha_k = 1/(4\sqrt{k})$; started at Metropolis weights



Fastest Mixing Continuous-time Markov Chain

and

Maximum Variance Unfolding

Continuous-time Markov chain

- continuous-time Markov chain with rate matrix $Q = Q^T$
- eigenvalues of Q ordered as

$$0 = \lambda_1(Q) > \lambda_2(Q) \geq \cdots \geq \lambda_n(Q)$$

- distribution $\pi(t)$ converges to uniform with rate determined by λ_2

$$\|\pi(t) - \mathbf{1}/n\|_{\mathrm{tv}} \leq (\sqrt{n}/2)e^{\lambda_2(Q)t}$$

- λ_2 is positive homogenous function of Q

Continuous-time FMMC

primal CT-FMMC

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{\{i,j\}\in\mathcal{E}} d_{ij}^2 Q_{ij} \\ \text{subject to} & Q = Q^T, \quad Q\mathbf{1} = 0, \quad Q_{ij} \geq 0 \text{ for } i \neq j, \quad Q \in \mathcal{S} \\ & \lambda_2(Q) \geq 1 \end{array}$$

dual CT-FMMC

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}\, X \\ \text{subject to} & X_{ii} + X_{jj} - X_{ij} - X_{ji} \leq d_{ij}^2, \quad \{i,j\} \in \mathcal{E} \\ & X\mathbf{1} = 0, \quad X \succeq 0 \end{array}$$

Maximum-variance unfolding

geometric interpretation of dual CT-FMMC problem
(Sun, Boyd, Xiao, Diaconis 2004)

- use variables $x_1, \ldots, x_n$, with $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} [x_1 \cdots x_n]$

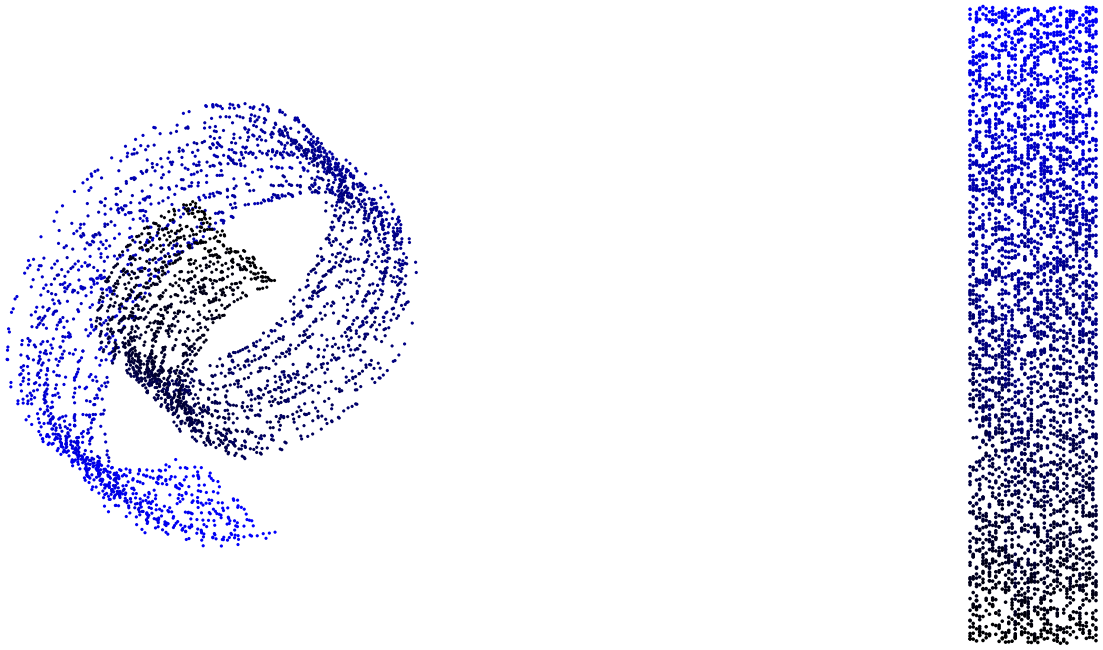
- dual problem becomes **maximum-variance unfolding problem**

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n \|x_i\|^2 \\ \text{subject to} & \sum_i x_i = 0, \quad \|x_i - x_j\| \leq d_{ij}, \quad \{i,j\} \in \mathcal{E} \end{array}$$

- position n points in $\mathbf{R}^n$ to maximize variance, respecting local distance constraints

Maximum-variance unfolding

- similar to **semidefinite embedding** for unsupervised learning of manifolds (L. Saul et al 2003)



- **surprise:** duality between CT-FMMC and max-variance unfolding

References

- Fast linear iterations for distributed averaging
  to appear *Systems and Control Letters*
- Fastest mixing Markov chain on a graph
  to appear *SIAM Review*
- Fast distributed algorithms for optimal redistribution
  submitted to *JOTA*
- Fastest mixing Markov chain on a path
  to appear *American Math. Monthly*
- Symmetry analysis of reversible Markov chains
  submitted to *Internet Mathematics*

available at `www.stanford.edu/~boyd`