Lecture 7 - Accelerated Dual Descent

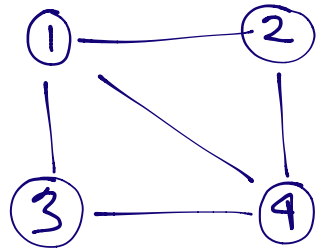
The aim of this last lecture is to expose you to some recent work in the network optimization literature. Specifically, we'll discuss a very recent paper proposing a distributed method to solve the min cost flow problem. First, we need to cover some preliminary concepts which come up in this work.

A Taste of Algebraic Graph Theory

Let $G=(\mathcal{N}, \mathcal{E})$ be a graph with N nodes and E edges.

Example: (To start, assume G is undirected.)

$G =$

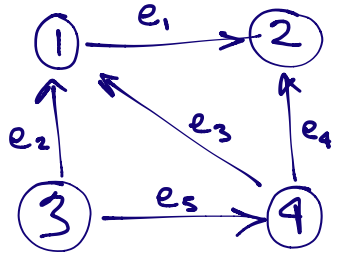


$N=4$

$E=5$

Let us arbitrarily orient and label each edge

Example:



Let A denote the $N \times E$ node-arc incidence matrix.

$$A_{ie} = \begin{cases} +1 & \text{if } e = (i,j) \\ -1 & \text{if } e = (j,i) \\ 0 & \text{otherwise} \end{cases}$$

For our example, we have

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} +1 & -1 & -1 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & +1 & 0 & 0 & +1 \\ 0 & 0 & +1 & +1 & -1 \end{bmatrix} \end{array}$$

Let $x \in \mathbb{R}^E$ be a flow vector.
Then the i^{th} entry of the vector Ax is

$$(Ax)_i = \sum_{j \mid (i,j) \in E} x_{ij} - \sum_{j \mid (j,i) \in E} x_{ji}$$ "surplus at node i"

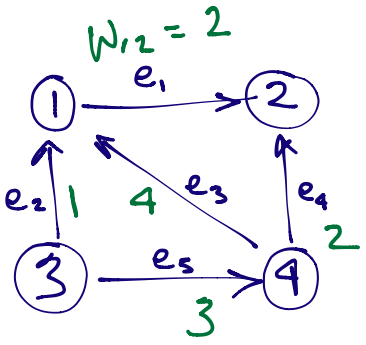
Let $y \in \mathbb{R}^N$ be a vector giving value y_i to node i.
Then for edge $e = (i,j)$,

$$(A^T y)_{e=(i,j)} = y_i - y_j$$ "disagreement on edge $e=(i,j)$"

The Laplacian matrix of G is $L = AA^T$.

If G is a weighted (undirected) graph, with weight $w_{ij} = w_{ji} > 0$ on edge $(i,j) \in \mathcal{E}$, we can form an $E \times E$ diagonal weight matrix W, with diagonal entries $(W)_{ee} = w_{ij}$ for $e = (i,j)$.

Example:



$$W = \begin{bmatrix} 2 & & & \\ & 1 & & 0 \\ & & 4 & \\ & 0 & & 2 \\ & & & & 3 \end{bmatrix}$$

For weighted graphs, the Laplacian is $L = AWA^T$.
(Thus, an unweighted graph is the same as taking $W = I$.)

The Laplacian encodes a lot of useful info about G.
(In class we only touched on properties relevant to the research article we discussed. For more on this very interesting field, I recommend:

- "Spectral Graph Theory" by Dan Spielman (a book chapter)
  http://www.cs.yale.edu/~spielman/PAPERS/SGTChapter.pdf
- Algebraic Graph Theory by Chris Godsil & Gordon Royle (a book)
- Spectral Graph Theory by Fan Chung Graham (a book)

Properties of L:

① Let $y \in \mathbb{R}^N$. $\quad y^T L y = \sum_{e=(i,j)} (y_i - y_j)^2 w_{ij}$

② Let B be the $N \times N$ (symmetric) adjacency matrix of G with

$$B_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \qquad (w_{ij} = 1 \text{ if } G \text{ is unweighted})$$

Let $d_i = \sum_j w_{ij}$ denote the (weighted) degree of node i.
Let $d \in \mathbb{R}^N$ denote the vector with entries d_i $(d = B\mathbb{1})$
Let $D = \text{diag}(d) = \text{diag}(B\mathbb{1})$.
Then $L = D - B$. (Verify this as an exercise.)

③ Let $y = c\mathbb{1}$ (i.e., $y \in \text{span}(\mathbb{1})$)
Then $Ly = 0$. Why? $Ly = (D-B)(c\mathbb{1})$
$= c(D\mathbb{1} - B\mathbb{1})$
$= c(d - d)$ ✓

④ Let $y = c\mathbb{1}$. Then $y^T L y = 0$.
This is clear since, from ③, $y^T L y = y^T 0 = 0$.
Alternatively, if $y = c\mathbb{1}$ then $y_i = y_j \; \forall \, i,j \in \mathcal{N}$, so $(y_i - y_j)^2 = 0$ and the claim follows from ①.

⑤ For any $y \in \mathbb{R}^N$, $y^T L y \geq 0$. $\Rightarrow L$ is positive semi-definite
This follows directly from ①.

⑥ If G is connected, then for any $y \perp \mathbb{1}$ (i.e., $y^T \mathbb{1} = 0$) it holds that $y^T L y > 0$. $\Rightarrow$ If G is connected, $\text{rank}(L) = n-1$
Why? If $y^T \mathbb{1} = 0$, $\sum_i y_i = 0$, so not all values of y are the same. Then there must be neighboring nodes i & j with $y_i \neq y_j \Rightarrow (y_i - y_j)^2 > 0$

⑦ Let $\mu_1 = 0 < \mu_2 \leq \mu_3 \leq \cdots \leq \mu_N$ denote the eigenvalues of L in ascending order. (Note: Since L is symmetric it has N real eigenvalues.)
The second smallest eigenvalue of L, μ_2, captures information about the connectivity of G, and μ_2 is often called the "algebraic connectivity"
E.g., let Δ denote the diameter of G (the max length of a shortest path in G). Then it can be shown that $\mu_2 \geq \frac{1}{N\Delta}$.

Sometimes it is more convenient to work with the normalized Laplacian,

$$\mathcal{L} \overset{\text{def}}{=} D^{-1/2} L D^{-1/2} = D^{-1/2}(D - B) D^{-1/2}$$
$$= I - D^{-1/2} B D^{-1/2}.$$

This is especially useful for graphs where some nodes have a much higher degree than others.

For $y \in \mathbb{R}^N$, $y^T \bar{L} y = \sum_{e=(i,j)} \left(\frac{y_i}{\sqrt{d_i}} - \frac{y_j}{\sqrt{d_j}} \right)^2 w_{ij}$.

It still holds that $\mu_1(\bar{L}) = 0$. (But instead of $y = c\mathbb{1}$ being the corresponding eigenvector, it is the vector with entries $y_i := \frac{\sqrt{d_i}}{\sum_j d_j}$.)

Also, the random walk matrix $P = D^{-1}B$ has the same eigenvalues as $D^{1/2} P D^{-1/2} = D^{-1/2} B D^{-1/2}$ (since they are related through a similarity transformation), and so

$$\mu_i(\bar{L}) = 1 - \mu_{N-i+1}(P)$$

where $\mu_i(X)$ is the i^{th} smallest eigenvalue of X.

Note that $\mu_2(\bar{L}) \geq \omega \Rightarrow 1 - \mu_{N-1}(P) \geq \omega$.
Recall that $1 - \mu_{N-1}(P)$ is the "spectral gap" of P, and we've previously seen how this is related to the rate of convergence of P to its stationary distribution.

One last useful property of the normalized Laplacian $\bar{L}$ is that the largest eigenvalue is bounded:

$$\mu_N(\bar{L}) \leq 2, \text{ and } \mu_N(\bar{L}) = 2 \text{ iff } G \text{ is bipartite.}$$

Accelerated Dual Descent

The rest of today's lecture discusses the paper:

"Accelerated Dual Descent for Network Flow Optimization"

by M Zargham, A Ribeiro, A Ozdaglar, and A Jadbabaie,

which has been accepted for publication in IEEE Trans. on Automatic Contrl

We consider a general convex min-cost flow problem.

We're given a graph $G=(\mathcal{N},\mathcal{E})$ and our goal is to

$$\underset{x\in\mathbb{R}^E}{\text{minimize}} \quad \sum_{e\in\mathcal{E}} \phi_e(x_e) \qquad (P)$$

$$\text{subject to} \quad Ax=b$$

where $b\in\mathbb{R}^N$ is given and satisfies $\sum_i b_i=0$ $\left(b^T\mathbb{1}=0\right)$,
and A is the node-arc incidence matrix of G.

Assumptions 1: G has properties:

a) G is connected with algebraic connectivity $\mu_2(L)\geq w$.

b) G is not bipartite. $\Rightarrow 0<\mu_i(L)<2$ for $i=2,\dots,n$

Assumptions 2: The cost functions $\phi_e(\cdot)$ have properties:

a) $\phi_e(\cdot)$ is twice differentiable

b) $\phi_e(\cdot)$ is strongly convex: $\exists\ \gamma, \Gamma > 0$ such that

$$0 < \gamma \leq \phi_e''(\cdot) \leq \Gamma < \infty$$

c) (Lipschitz continuity of $1/\phi_e''$) $\exists\ \xi > 0$ such that for all $x_e, \bar{x}_e$

$$\left| 1/\phi_e'(x_e) - 1/\phi_e(\bar{x}_e) \right| \leq \xi \left| x_e - \bar{x}_e \right|$$

In previous lectures we studied the linear min cost flow problem which is equivalent to taking $\phi_e(x_e) = a_e x_e$.
Note that this cost function doesn't satisfy Assumptions 2 since $\phi_e''(\cdot) = 0$

An example cost function that satisfies Assumptions 2 is

$$\phi_e(x_e) = a_e (x_e - b_e)^2$$

with $a_e > 0$.

Our goal is to solve the problem (P) in a distributed way:

- Node i is responsible for updating x_e if $e = (i,j)$
- We want algorithms where nodes only use "local" information to perform updates (e.g., from neighbors in G).

Dual Problem Formulation

The problem (P) is convex but constrained. To get an unconstrained formulation we work with the dual. Under the assumptions above we know that an optimal solution to the dual coincides with the optimal solution to the primal.

Introduce Lagrange multipliers $\lambda \in \mathbb{R}^N$ (One λ_i for each row of $Ax = b$.)

The Lagrangian function for (P) is

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T (Ax - b)$$

The dual objective function is $\tilde{q}(\lambda) = \min_x \mathcal{L}(x, \lambda)$.
Since $\mathcal{L}(x, \lambda)$ is convex in x for any fixed λ, $\nabla_x \mathcal{L}(x^*, \lambda) = 0$ is necessary and sufficient for x^* to minimize $\mathcal{L}(x, \lambda)$ over x, with λ held fixed. Consider the flow variable x_e for $e = (i, j)$.

$$\frac{\partial}{\partial x_e} \mathcal{L}(x, \lambda) = \phi_e'(x_e) - (\lambda_i - \lambda_j) = 0$$

$$\Rightarrow x_e^*(\lambda) = (\phi_e')^{-1}(\lambda_i - \lambda_j)$$

Let $x(\lambda)$ denote the flow vector with entries $x_e(\lambda)$ given by this last formula.

Then $\tilde{q}(\lambda) = f(x(\lambda)) - \lambda^T (Ax(\lambda) - b)$.

The dual problem is to

$$\underset{\lambda}{\text{maximize}}\ \tilde{q}(\lambda)$$

which is unconstrained (so we're happy).
Of course, it's equivalent to

$$\underset{\lambda}{\text{minimize}}\ q(\lambda) = \lambda^T(Ax(\lambda)-b) - f(x(\lambda)) \qquad (D)$$

which we'll focus on.
In addition to being unconstrained, $q(\lambda)$ has interesting properties.

Lemma 1: a) $\nabla^2 q(\lambda) = A\left[\nabla^2 f(x(\lambda))\right]^{-1} A^T$ ← a weighted Laplacian!

b) $q(\lambda)$ is strongly convex on $\{v \mid v^T\mathbb{1}=0\} \overset{\text{def}}{\equiv} \mathbb{1}^\perp$

This means that $\exists\ m, M>0$ s.t. for $v \in \mathbb{1}^\perp$,

$$\frac{1}{M} v^T v \le v^T \nabla^2 q(\lambda) v \le \frac{1}{m} v^T v.$$

c) $\nabla^2 q(\lambda)$ is Lipschitz continuous: $\exists\ L = \frac{N\beta}{\gamma} > 0$ s.t. for all $\lambda, \bar{\lambda}$

$$\|\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})\| \le L\|\lambda - \bar{\lambda}\|$$

Property (a) follows, effectively, from the problem structure.
Properties (b) & (c) follow from Assumptions 2 above.

Gradient Descent

This is one way to solve the dual problem (D).

- Given an arbitrary initial condition $\lambda_0 \in \mathbb{R}^N$
- Repeat for $k \geq 0$

$$\lambda_{k+1} = \lambda_k - \alpha_k g_k$$

where $g_k = \nabla q(\lambda_k) = A x(\lambda_k) - b$

Observe that the i^{th} entry of g_k is

$$(g_k)_i = [\nabla q(\lambda_k)]_i = \sum_{j|(i,j)\in\mathcal{E}} x_{ij}(\lambda) - \sum_{j|(j,i)\in\mathcal{E}} x_{ji}(\lambda) - b_i$$

We can exploit the problem structure to implement this in a distributed way. Assume:

- Node i knows b_i
- Node i is responsible for updating $(\lambda_k)_i$

Then to compute $(g_k)_i$, node i needs to get λ_j from neighbors j in G. This suggests the following distributed algorithm, where each node updates its λ_i in parallel.

Repeat for $k \geq 0$, at node i

- Send $(\lambda_k)_i$ to neighbors
- Receive $(\lambda_k)_j$ from neighbors
- Compute $x_e(\lambda_k)$ for edges (i,j) and (j,i)
- Compute $(g_k)_i$
- Update $(\lambda_{k+1})_i = (\lambda_k)_i - \alpha_k (g_k)_i$

This is great because it gives us a distributed algorithm to solve our problem.

The reason we're not satisfied is that gradient descent converges slowly. (linear convergence)

$$q(\lambda_k) - q^* \leq c^k \left(q(\lambda_0) - q^*\right)$$

where $q^* = \min_{\lambda} q(\lambda)$ and $c = 1 - \frac{m}{M}$.

(See Ch 9 of Boyd & Vandenberghe for more.)

To get faster convergence we should exploit the fact that $q(\lambda)$ is strongly convex and twice differentiable.

Newton's Method

Instead of gradient descent, use iterates, for $k \geq 0$

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k$$

where the Newton direction d_k satisfies

$$\nabla^2 q(\lambda_k)\, d_k = -g_k.$$

Define $H_k = \nabla^2 q(\lambda_k)$ "Hessian matrix at iteration k"

To get d_k, we would like to set $d_k = -H_k^{-1} g_k$, but we need to be careful. From Lemma 1, $H_k = A\left[\nabla^2 f(x(\lambda_k))\right]^{-1} A^T$

$\Rightarrow H_k$ is a weighted Laplacian, so it has $\operatorname{rank}(H_k) = n-1$ and thus is not invertible.

However, since $g_k = A x(\lambda_k) - b$ and $\mathbb{1}^T g_k = \underbrace{\mathbb{1}^T A}_{=0} x(\lambda) - \underbrace{\mathbb{1}^T b}_{=0} = 0$

we have $g_k \perp \mathbb{1}$.

Then we can set $d_k = -H_k^+ g_k$, where H_k^+ is the pseudoinverse

This gives us a Newton method which is very fast.
Since $q(\lambda)$ is strongly convex, it holds that

$$q(\lambda) - q^* \leq \frac{M}{2} \|\nabla q(\lambda)\|_2^2$$

Newton's algorithm converges (typically) in two phases.
In the first phase (when $\|\nabla q(\lambda_k)\|_2 > \eta$ for η depending on the problem)

$$q(\lambda_{k+1}) - q^* \leq q(\lambda_k) - q^* - \beta$$

I.e., we make constant progress, where the constant β depends on the problem (strong convexity constants, and Lipschitz constants).
In the second phase (when $\|\nabla q(\lambda_k)\| < \eta$), the convergence is quadratic.
$\Rightarrow$ If k is the first iteration where $\|\nabla q(\lambda_k)\|_2 < \eta$, then

$$q(\lambda_\ell) - q^* \leq \frac{2}{M^3 L} \left(\frac{1}{2}\right)^{2^{\ell-k-1}}$$

Once we reach the second phase, we're done after a few steps.
For more, see Ch 9 in Boyd & Vandenberghe.

The issue with Newton's method is that the pseudoinverse H_k^+ is not easily computed in a distributed manner, using only local information.

The authors of this paper propose a method to approximate H_k^+ in a distributed way, and they analyze the resulting algorithm.

Accelerated Dual Descent

We know H_k is a weighted Laplacian.

Consider "splitting" H_k by writing $H_k = D_k - B_k$,

where D_k is diagonal, $(D_k)_{ii} = 2(H_k)_{ii}$

and B_k has entries

$$(B_k)_{ij} = \begin{cases} (H_k)_{ii} & \text{if } i=j \\ -(H_k)_{ij} & \text{if } i \neq j \end{cases}$$

$\Rightarrow$ We view H_k as the Laplacian of a graph with weighted adjacency matrix B_k.

The corresponding random walk would be $D_k^{-1} B_k$.

We can also write $H_k = D_k^{1/2} \underbrace{\left(I - D_k^{-1/2} B_k D_k^{-1/2}\right)}_{\text{Normalized Laplacian}} D_k^{1/2}$

Then the pseudo inverse is

$$H_k^{+} = D_k^{-1/2} \left(I - D_k^{-1/2} B_k D_k^{-1/2}\right)^{+} D_k^{-1/2}.$$

To get an approximation, use the Taylor expansion for the pseudoinverse:

Suppose $v \perp \mathbb{1}$.

Then $(I-X)^{+} v = \left(\sum_{r \geq 0} X^r\right) v$.

We'll apply this with $X = D_k^{-1/2} B_k D_k^{-1/2}$.

Note that the eigenvalues $\mu_i\left(D_k^{-1/2} B_k D_k^{-1/2}\right) \in (-1, +1)$ for $i \geq 2$, since we assumed that G is not bipartite. Hence its powers converge.

We can get an approximate Newton direction by using the first m terms of the Taylor approximation:

$$d_k^{(m)} = -\sum_{r=0}^{m} D_k^{-1/2}\left(D_k^{-1/2} B_k D_k^{-1/2}\right)^r D_k^{-1/2} g_k$$

This also gives us a distributed algorithm. We can show that to compute $(d_k^{(m)})_i$, node i only needs information from its m-hop neighbors.

The rest of the paper analyzes convergence properties and rates of convergence of the approximate algorithm. In particular, the approximate Newton update keeps the nice convergence properties of Newton's method, up to a point. Once we're very close to the optimum, errors from the truncated Taylor approximation kick in, and so f_k only converges to a ball around the optimal variables f^*. The results are extremely interesting and I encourage you to read the paper for more details.