Today

Lagrange Multipliers.

Today

Lagrange Multipliers.

Fast Solution of Laplacian Systems.

Lagrangian Dual.

Find x, subject to

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian:

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x, \lambda)$ is

(A) non-negative in expectation

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

(C) non-positive for any valid λ.

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

(C) non-positive for any valid λ.

If $\exists \lambda \geq 0$, where $L(x,\lambda)$ is positive for all x

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x,\lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

(C) non-positive for any valid λ.

If $\exists \lambda \geq 0$, where $L(x,\lambda)$ is positive for all x

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x, \lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

(C) non-positive for any valid λ.

If $\exists \lambda \geq 0$, where $L(x, \lambda)$ is positive for all x

(A) there is no feasible x.

Lagrangian Dual.

Find x, subject to

$f_i(x) \leq 0, i = 1, \ldots m.$

Remember calculus (constrained optimization.)

Lagrangian: $L(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$

$\lambda_i \geq 0$ - Lagrangian multiplier for inequality i.

For feasible solution x, $L(x, \lambda)$ is

(A) non-negative in expectation

(B) positive for any λ.

(C) non-positive for any valid λ.

If $\exists \lambda \geq 0$, where $L(x, \lambda)$ is positive for all x

(A) there is no feasible x.

(B) there is no x, λ with $L(x, \lambda) < 0$.

Lagrangian:constrained optimization.

$$\begin{aligned} &\min \quad f(x) \\ &\text{subject to } f_i(x) \le 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian:constrained optimization.

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to } f_i(x) \leq 0, & & i = 1, ..., m \end{array}$$

Lagrangian function:

Lagrangian:constrained optimization.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & i = 1, ..., m \end{aligned}$$

Lagrangian function:

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Lagrangian:constrained optimization.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \le 0, \quad & \quad i = 1, ..., m \end{aligned}$$

Lagrangian function:

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v

Lagrangian:constrained optimization.

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to } f_i(x) \leq 0, & & i = 1, ..., m \end{array}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$

Lagrangian:constrained optimization.

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to } f_i(x) \le 0, & & i = 1,\ldots,m \end{array}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \le 0$
For all $\lambda \ge 0$ have $L(x,\lambda) \le v$

Lagrangian:constrained optimization.

$$\begin{array}{ll} \min & f(x) \\ \text{subject to } f_i(x) \leq 0, & i = 1, ..., m \end{array}$$

Lagrangian function:

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$

For all $\lambda \geq 0$ have $L(x, \lambda) \leq v$

Maximizing λ, only positive λ_i when $f_i(x) = 0$

Lagrangian:constrained optimization.

$$\begin{array}{lll} \min & f(x) & \\ \text{subject to } f_i(x) \leq 0, & & i = 1, ..., m \end{array}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$

For all $\lambda \geq 0$ have $L(x,\lambda) \leq v$

Maximizing λ, only positive λ_i when $f_i(x) = 0$

which implies $L(x,\lambda) \geq f(x)$

Lagrangian:constrained optimization.

$$\begin{array}{ll} \min & f(x) \\ \text{subject to } f_i(x) \leq 0, & i = 1, ..., m \end{array}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$

For all $\lambda \geq 0$ have $L(x,\lambda) \leq v$

Maximizing λ, only positive λ_i when $f_i(x) = 0$

which implies $L(x,\lambda) \geq f(x) = v$

Lagrangian:constrained optimization.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & i = 1, ..., m \end{aligned}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$

For all $\lambda \geq 0$ have $L(x,\lambda) \leq v$

Maximizing λ, only positive λ_i when $f_i(x) = 0$

which implies $L(x,\lambda) \geq f(x) = v$

Lagrangian:constrained optimization.

$$\begin{aligned} &\min \quad f(x) \\ &\text{subject to } f_i(x) \le 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \le 0$

For all $\lambda \ge 0$ have $L(x,\lambda) \le v$

Maximizing λ, only positive λ_i when $f_i(x) = 0$

which implies $L(x,\lambda) \ge f(x) = v$

If there is λ with $L(x,\lambda) \ge \alpha$ for all x

Lagrangian:constrained optimization.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \le 0, \quad & i = 1,...,m \end{aligned}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \le 0$
For all $\lambda \ge 0$ have $L(x,\lambda) \le v$
Maximizing λ, only positive λ_i when $f_i(x) = 0$
which implies $L(x,\lambda) \ge f(x) = v$

If there is λ with $L(x,\lambda) \ge \alpha$ for all x
Optimum value of program is at least α

Lagrangian:constrained optimization.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, & \qquad i = 1, ..., m \end{aligned}$$

Lagrangian function:

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$
For all $\lambda \geq 0$ have $L(x,\lambda) \leq v$
Maximizing λ, only positive λ_i when $f_i(x) = 0$
which implies $L(x,\lambda) \geq f(x) = v$

If there is λ with $L(x,\lambda) \geq \alpha$ for all x
Optimum value of program is at least α

Primal problem:
x, that minimizes $L(x,\lambda)$ over all $\lambda \geq 0$.

Lagrangian:constrained optimization.

$$\begin{array}{ll} \min & f(x) \\ \text{subject to } f_i(x) \leq 0, & i = 1, ..., m \end{array}$$

Lagrangian function:

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

If (primal) x has value v $f(x) = v$ and all $f_i(x) \leq 0$
For all $\lambda \geq 0$ have $L(x, \lambda) \leq v$
Maximizing λ, only positive λ_i when $f_i(x) = 0$
which implies $L(x, \lambda) \geq f(x) = v$

If there is λ with $L(x, \lambda) \geq \alpha$ for all x
Optimum value of program is at least α

Primal problem:
x, that minimizes $L(x, \lambda)$ over all $\lambda \geq 0$.

Dual problem:
λ, that maximizes $L(x, \lambda)$ over all x.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, & \qquad i = 1, ..., m \end{aligned}$$

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} &\min \quad f(x) \\ &\text{subject to } f_i(x) \leq 0, \qquad i = 1, ..., m \end{aligned}$$

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & i = 1, ..., m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} &\min \quad f(x) \\ &\text{subject to } f_i(x) \leq 0, \qquad i = 1, ..., m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & \quad i = 1, ..., m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} &\min \quad f(x) \\ &\text{subject to } f_i(x) \leq 0, \qquad i = 1, \ldots, m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Feasible primal, $f_i(x^*) \leq 0$, and feasible dual $\lambda_i \geq 0$.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & i = 1, \ldots, m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Feasible primal, $f_i(x^*) \leq 0$, and feasible dual $\lambda_i \geq 0$.

Complementary slackness: $\lambda_i f_i(x^*) = 0$.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & \quad i = 1, ..., m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Feasible primal, $f_i(x^*) \leq 0$, and feasible dual $\lambda_i \geq 0$.

Complementary slackness: $\lambda_i f_i(x^*) = 0$.

Launched nonlinear programming!

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\min \quad f(x)$$
$$\text{subject to } f_i(x) \le 0, \qquad i = 1, \dots, m$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Feasible primal, $f_i(x^*) \le 0$, and feasible dual $\lambda_i \ge 0$.

Complementary slackness: $\lambda_i f_i(x^*) = 0$.

Launched nonlinear programming! See paper.

Why important: KKT.

Karash, Kuhn and Tucker Conditions.

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } f_i(x) \leq 0, \quad & i = 1, \ldots, m \end{aligned}$$

$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$

Local minima for feasible x^*.

There exist multipliers λ, where

$\nabla f(x^*) + \sum_i \lambda_i \nabla f_i(x^*) = 0$

Feasible primal, $f_i(x^*) \leq 0$, and feasible dual $\lambda_i \geq 0$.

Complementary slackness: $\lambda_i f_i(x^*) = 0$.

Launched nonlinear programming! See paper.

Linear Program.

$\min cx, Ax \geq b$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } b_i - a_i \cdot x \leq 0, & \qquad i = 1, ..., m \end{aligned}$$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

or

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda.$

Best λ?

Linear Program.

$\min cx, Ax \geq b$

$$
\begin{aligned}
\min \quad & c \cdot x \\
\text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, \ldots, m
\end{aligned}
$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad & i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda.$

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda$,

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c,$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } b_i - a_i \cdot x \leq 0, \quad & i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda.$

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Duals!

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Duals!

Note:

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x).$

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda.$

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Duals!

Note: Lagrange multipliers for equality constraints.

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \qquad i = 1, ..., m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Duals!

Note: Lagrange multipliers for equality constraints.

Usually: ν, and un-restricted.

Linear Program.

$\min cx, Ax \geq b$

$$\begin{aligned} \min \quad & c \cdot x \\ \text{subject to } & b_i - a_i \cdot x \leq 0, \quad & i = 1, \ldots, m \end{aligned}$$

Lagrangian (Dual):

$L(\lambda, x) = cx + \sum_i \lambda_i (b_i - a_i x)$.

or

$L(\lambda, x) = -(\sum_j x_j (a_j \lambda - c_j)) + b\lambda$.

Best λ?

$\max b \cdot \lambda$ where $a_j \lambda = c_j$.

$\max b\lambda, \lambda^T A = c, \lambda \geq 0$

Duals!

Note: Lagrange multipliers for equality constraints.

Usually: ν, and un-restricted.

In this case, x for lagrangian of Dual.

Linear Systems...

Linear Systems...

Linear Systems

$$Ax = b$$

Linear Systems

$Ax = b$

Find x.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$

$O(n^{2.36\ldots})$ with fast matrix multiplication.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$

$O(n^{2.36\ldots})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$
$O(n^{2.36\ldots})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.
For today: where m is sum of nonzeros in matrix.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$
$O(n^{2.36\ldots})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.
For today: where m is sum of nonzeros in matrix.
For positive semidefinite matrix.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$
$O(n^{2.36\ldots})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.
For today: where m is sum of nonzeros in matrix.
For positive semidefinite matrix.

Today: $\tilde{O}(m)$ for Laplacian matrices.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$
$O(n^{2.36...})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.
For today: where m is sum of nonzeros in matrix.
For positive semidefinite matrix.

Today: $\tilde{O}(m)$ for Laplacian matrices.
Laplacian: $dI - A$ where A is adjacency matrix of a graph.

Linear Systems

$Ax = b$

Find x.

Gaussian elimination: $O(n^3)$
$O(n^{2.36\ldots})$ with fast matrix multiplication.

Iterative Methods: $O(nm \log \frac{1}{\varepsilon})$ to ε approximate.
For today: where m is sum of nonzeros in matrix.
For positive semidefinite matrix.

Today: $\tilde{O}(m)$ for Laplacian matrices.
Laplacian: $dI - A$ where A is adjacency matrix of a graph.
$\rightarrow$ symmetric diagonally dominant matrices by reduction.

Electrical Flow: a detour.

A graph $G = (V, E)$.

Electrical Flow: a detour.

A graph $G = (V, E)$.
Circuit: nodes V, resistors E, value 1 (for today.)

Electrical Flow: a detour.

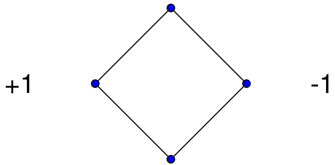
A graph $G = (V, E)$.
  Circuit: nodes V, resistors E, value 1 (for today.)

Given $\chi : V \to \Re$
  Find flow that routes χ and minimizes
    $\sum_e f(e)^2$.

Electrical Flow: a detour.

A graph $G = (V, E)$.
 Circuit: nodes V, resistors E, value 1 (for today.)

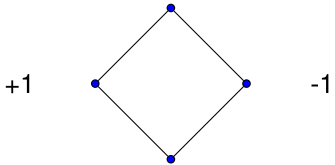
Given $\chi : V \to \Re$
 Find flow that routes χ and minimizes
 $\sum_e f(e)^2$.



Electrical Flow: a detour.

A graph $G = (V, E)$.
Circuit: nodes V, resistors E, value 1 (for today.)

Given $\chi : V \to \Re$
Find flow that routes χ and minimizes
$\sum_e f(e)^2$.

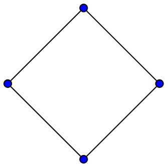


Claim: Minimizer is electrical flow.

Electrical Flow: a detour.

A graph $G = (V, E)$.
Circuit: nodes V, resistors E, value 1 (for today.)

Given $\chi : V \to \Re$
Find flow that routes χ and minimizes
$\sum_e f(e)^2$.

+1  -1

Claim: Minimizer is electrical flow.

Flow corresponds to flow induced by a set of potentials.

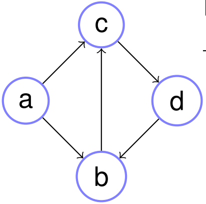
Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e = (u,v) \\ 1 & e = (v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u = v \\ -1 & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

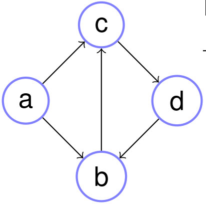
L

	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e = (u,v) \\ 1 & e = (v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u = v \\ -1 & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

L

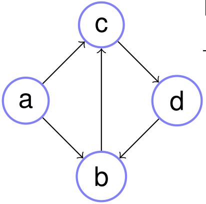
	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

Fun facts: $\mathbf{f} \in \Re^{|E|}$

Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e = (u,v) \\ 1 & e = (v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u = v \\ -1 & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

L

	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

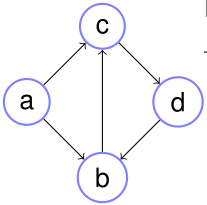
Fun facts: $\mathbf{f} \in \Re^{|E|}$

$[B^T f]_u = \sum_{e=(u,v)} f_e - \sum_{e=(v,u)} f_e$

Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e = (u,v) \\ 1 & e = (v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u = v \\ -1 & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

L

	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

Fun facts: $\mathbf{f} \in \Re^{|E|}$

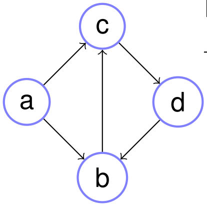
$[B^T f]_u = \sum_{e=(u,v)} f_e - \sum_{e=(v,u)} f_e$

$B^T B = L$

Some Matrices.

Given $G=(V,E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e=(u,v) \\ 1 & e=(v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u=v \\ -1 & (u,v)\in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

L

	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

Fun facts: $\mathbf{f} \in \Re^{|E|}$

$[B^T f]_u = \sum_{e=(u,v)} f_e - \sum_{e=(v,u)} f_e$

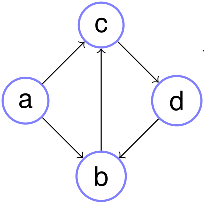
$B^T B = L$

$[Bx]_{e=(u,v)} = x_u - x_v$

Some Matrices.

Given $G = (V, E)$, arbitrarily orient edges.

$$B_{v,e} = \begin{cases} -1 & e = (u,v) \\ 1 & e = (v,u) \\ 0 & \text{otherwise} \end{cases} \qquad L_{u,v} = \begin{cases} d & u = v \\ -1 & (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$



B

	a	b	c	d
(a,b)	1	-1	0	0
(a,c)	1	0	-1	0
(c,d)	0	0	1	-1
(d,b)	0	-1	0	1
(b,c)	0	1	-1	1

L

	a	b	c	d
a	2	-1	-1	0
b	-1	2	0	-1
c	-1	-1	3	-1
d	-1	-1	-1	3

Fun facts: $\mathbf{f} \in \Re^{|E|}$

$[B^T f]_u = \sum_{e=(u,v)} f_e - \sum_{e=(v,u)} f_e$

$B^T B = L$

$[Bx]_{e=(u,v)} = x_u - x_v$

$x^T L x = \sum_{e=(u,v)} (x_u - x_v)^2$

Duality..

Given G, χ, $\chi \perp 1$

Duality..

Given G, χ, $\chi \perp 1$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian:

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual:

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$?

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
  $2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Dual problem:

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Dual problem: Find ϕ that maximizes ...

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Dual problem: Find ϕ that maximizes ...
$\max_\phi 2\phi^T \chi - \phi^T L \phi$

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Dual problem: Find ϕ that maximizes ...
$\max_\phi 2\phi^T \chi - \phi^T L \phi$
Note: want $\phi^T L \phi = \sum_e (\phi_u - \phi_v)^2$ to be small.

Duality..

Given G, χ, $\chi \perp 1$

Minimize $|f|^2$ subject to $B^T f = \chi$.

Lagrangian: $L(\phi, f) = \sum_e f(e)^2 + 2\phi^T(\chi - B^T f)$

Lagrangian Dual: Find ϕ that maximizes $\min_f L(\phi, f)$.

Given ϕ, minimize $L(\phi, f)$? Calculus.

For $e = (u, v)$
$2f(e) + 2(\phi_v - \phi_u) = 0$ (Minimum when partial derivatives = 0.)

$\rightarrow f(e) = (\phi_u - \phi_v)$ Potential differences!!!

Matrix Form: $f = B\phi$ Again, flows should be potential differences.

Dual problem: Find ϕ that maximizes ...
$\max_\phi 2\phi^T \chi - \phi^T L\phi$

Note: want $\phi^T L\phi = \sum_e (\phi_u - \phi_v)^2$ to be small.
Minimize Squared Potential differences!

Why did we take dual?

Dual problem:

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_{\phi} 2\phi\chi - \phi L\phi$

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Also useful for convergence.

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Also useful for convergence.

Algorithm maintains feasible ϕ, f,

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_{\phi} 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Also useful for convergence.

Algorithm maintains feasible ϕ, f,

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi^T L\phi$

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_{\phi} 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Also useful for convergence.

Algorithm maintains feasible ϕ, f,

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi^T L\phi$

Duality gap is “distance” from optimal!

Why did we take dual?

Dual problem:
Find ϕ that maximizes ...
$\max_\phi 2\phi\chi - \phi L\phi$

Take the derivative:
$L\phi - \chi$
$L\phi = \chi$ at optimal point!

Optimal potential is solution to a Laplacian linear system.

Also useful for convergence.

Algorithm maintains feasible ϕ, f,

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi^T L\phi$

Duality gap is “distance” from optimal!

Algorithm: Work on flow and potentials.
To drive gap to 0.

Alg.

Given: χ, G

Alg.

Given: χ, G

Take a spanning tree T of G.

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u,v) + 1)$
($l_T(u,v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log \log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u,v) + 1)$
($l_T(u,v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u,v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$!

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u,v)+1)$
($l_T(u,v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u,v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u,v) + 1)$
($l_T(u,v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u,v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log \log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log \log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u, v) + 1)$
($l_T(u, v)$ path length in T)
Route excess on path through tree.

Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u, v)$

Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1 + \varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! ! ! !

Alg.

Given: χ, G

Take a spanning tree T of G. (Which tree?)

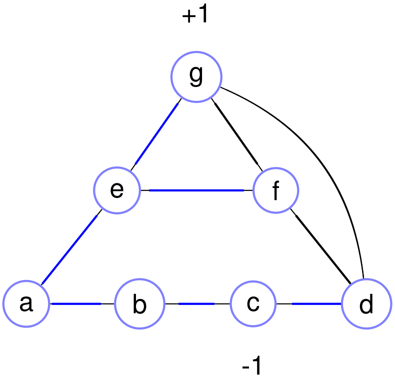
Route flow, f, to satisfy χ through T
Compute, ϕ, using tree ; $\phi_s = 0$, add f_e through T

Repeat:
Choose non-tree edge $e = (u, v)$ (Which non-tree edge?)
$f(e) = (\phi_u - \phi_v)/(l_T(u,v) + 1)$
($l_T(u,v)$ path length in T)
Route excess on path through tree.

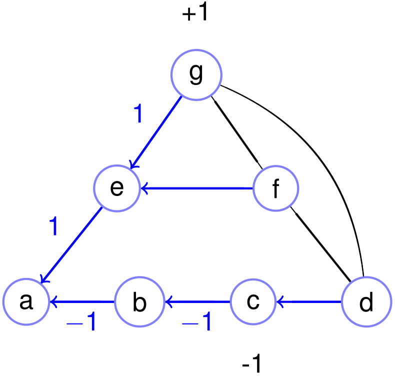
Which Tree?
Claim: Linear time algorithm for T w/ stretch $O(m \log n \log\log n)$!
Stretch: $\sum_{e=(u,v)} l_T(u,v)$

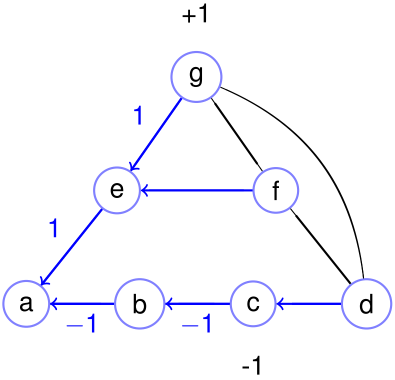
Which non-tree edge?
Choose an edge w/prob. proportional to $l_T(e)$.

Finds $(1+\varepsilon)$ approximation in $O(m \log n \log\log n \log(\frac{n}{\varepsilon}))$! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! ! ! ! !

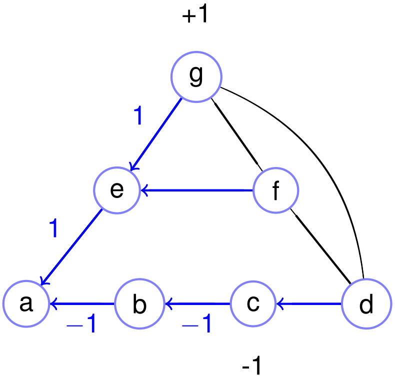


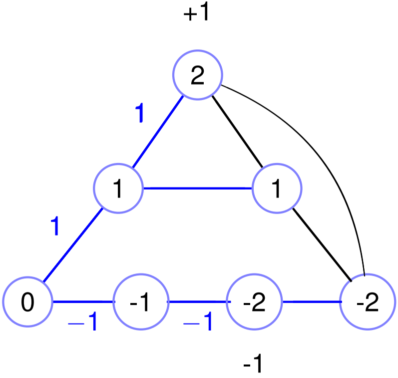
+1
g
e
f
a
b
c
d
-1


+1
g
1
e
f
1
a
b
c
d
−1
−1
-1

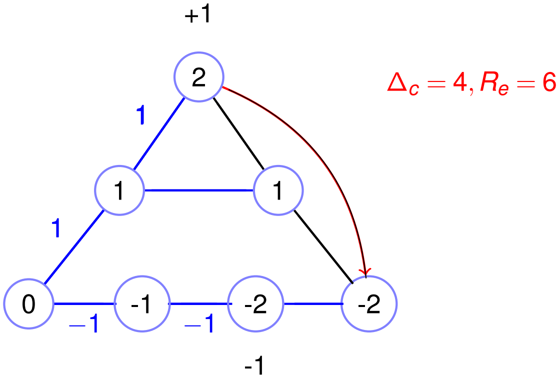


+1
g
1
e
f
1
a
b
c
d
−1
−1
-1

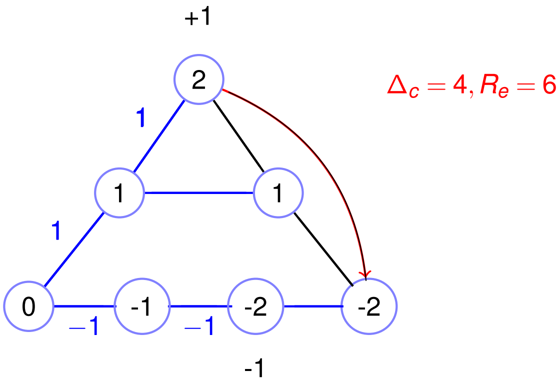

+1
g
1
e
f
1
a
b
c
d
−1
−1
-1

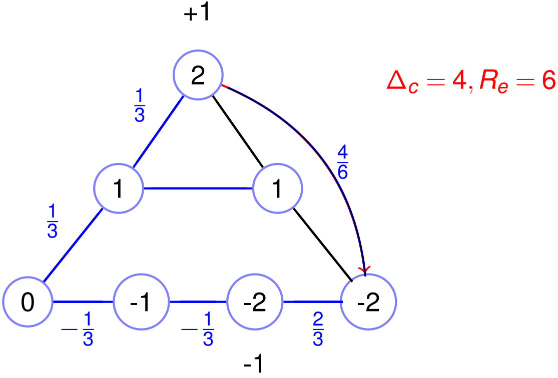


+1
2
1
1
1
1
0
-1
-2
-2
−1
−1
-1

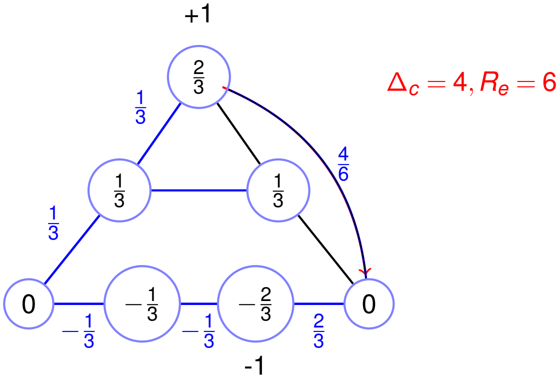


+1
2
1
1
1
1
$\Delta_c = 4, R_e = 6$
0
-1
-2
-2
-1
-1
-1


+1
2
1
1
1
1
0
-1
-2
-2
−1
−1
-1
$\Delta_c = 4, R_e = 6$



+1
2
1
1
0
-1
-2
-2
-1
$\frac{1}{3}$
$\frac{1}{3}$
$\frac{4}{6}$
$-\frac{1}{3}$
$-\frac{1}{3}$
$\frac{2}{3}$
$\Delta_c = 4, R_e = 6$



+1
$\frac{2}{3}$
$\frac{1}{3}$
$\frac{1}{3}$
$\frac{1}{3}$
$\frac{1}{3}$
$\frac{4}{6}$
0
$-\frac{1}{3}$
$-\frac{2}{3}$
0
$-\frac{1}{3}$
$-\frac{1}{3}$
$\frac{2}{3}$
-1
$\Delta_c = 4, R_e = 6$

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Algorithm:

Energy reduction.

Given $T, e = (u, v)$, let $R_e = l_T(u, v) + 1$.

Algorithm:

Energy reduction.

Given T,$e = (u, v)$, let $R_e = l_T(u, v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u, v)$.

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

Energy reduction.

Given T,$e=(u,v)$, let $R_e = l_T(u,v)+1$.

Algorithm:

Repeatedly "Fix" edge $e=(u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e')-\delta)^2 - (f(e'))^2$

Energy reduction.

Given T,$e=(u,v)$, let $R_e = l_T(u,v)+1$.

Algorithm:

Repeatedly “Fix” edge $e=(u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e
(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e')-\delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v)+1$.

Algorithm:

Repeatedly “Fix” edge $e = (u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e') - \delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Energy reduction.

Given T,$e=(u,v)$, let $R_e = l_T(u,v)+1$.

Algorithm:

Repeatedly "Fix" edge $e=(u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e}(f(e')-\delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Note: $\sum_{e' \in C_e} f(e') = R_e \delta$

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e}(f(e') - \delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Note: $\sum_{e' \in C_e} f(e') = R_e \delta$

$\rightarrow -\Delta_{C_e}^2 / R_e$ where $\Delta_{C_e} = \sum_{e' \in C_e} f_(e')$.

Energy reduction.

Given T,$e = (u,v)$, let $R_e = l_T(u,v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e') - \delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Note: $\sum_{e' \in C_e} f(e') = R_e \delta$

$\rightarrow -\Delta_{C_e}^2 / R_e$ where $\Delta_{C_e} = \sum_{e' \in C_e} f_(e')$.

Fix a part of the potential difference, Δ_{c_e} around cyle!!

Energy reduction.

Given T, $e = (u, v)$, let $R_e = l_T(u, v) + 1$.

Algorithm:

Repeatedly "Fix" edge $e = (u, v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e') - \delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Note: $\sum_{e' \in C_e} f(e') = R_e \delta$

$\rightarrow -\Delta_{C_e}^2 / R_e$ where $\Delta_{C_e} = \sum_{e' \in C_e} f_(e')$.

Fix a part of the potential difference, Δ_{c_e} around cyle!!

$\rightarrow$ reduction of $\Delta_{C_e}^2 / R_e$ in energy!

Energy reduction.

Given T,$e=(u,v)$, let $R_e = l_T(u,v)+1$.

Algorithm:

Repeatedly "Fix" edge $e=(u,v)$.

Route $-\delta = -\frac{\sum_{e' \in C_e} f(e')}{R_e}$ flow around cycle induced in T: C_e

(assume e' are oriented around cycle.)

Difference in energy from f and f'.

$\sum_{e' \in C_e} (f(e') - \delta)^2 - (f(e'))^2 = \sum_{e' \in C_e} -2f(e')\delta + \delta^2$

$= -(2\delta \sum_{e' \in C_e} f(e')) + R_E \delta^2$

Note: $\sum_{e' \in C_e} f(e') = R_e \delta$

$\rightarrow -\Delta_{C_e}^2 / R_e$ where $\Delta_{C_e} = \sum_{e' \in C_e} f_(e')$.

Fix a part of the potential difference, Δ_{c_e} around cyle!!

$\rightarrow$ reduction of $\Delta_{C_e}^2 / R_e$ in energy!

Fix $1/R_e$ of a cycle violation!

Duality Gap?

Algorithm maintains feasible ϕ, f, $(B^T f = \chi)$

Duality Gap?

Algorithm maintains feasible ϕ, f, $(B^T f = \chi)$

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$$

Duality Gap?

Algorithm maintains feasible ϕ, f, $(B^T f = \chi)$

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T(f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L \phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

$\Delta_\phi(u,v) = \sum_{e \in P_{u,v}} -f(e)$. assume $f(e)$ is oriented around cycle.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

$\Delta_\phi(u,v) = \sum_{e \in P_{u,v}} -f(e)$. assume $f(e)$ is oriented around cycle.

For $e \in T$, $\Delta_\phi(e) = 0$.

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L\phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

$\Delta_\phi(u,v) = \sum_{e \in P_{u,v}} -f(e)$. assume $f(e)$ is oriented around cycle.

For $e \in T$, $\Delta_\phi(e) = 0$. For $e \notin T$ $f(e) + \sum_{e \in P_e} f(e) = \Delta_{c_e}(f)$

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L \phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

$\Delta_\phi(u,v) = \sum_{e \in P_{u,v}} -f(e)$. assume $f(e)$ is oriented around cycle.

For $e \in T$, $\Delta_\phi(e) = 0$. For $e \notin T$ $f(e) + \sum_{e \in P_e} f(e) = \Delta_{c_e}(f)$

Duality Gap: $\sum_{e \notin T} \sum_e \Delta_{c_e}(f)^2$

Duality Gap?

Algorithm maintains feasible ϕ, f, ($B^T f = \chi$)

Primal value: $|f|^2$.
Dual value: $2\phi\chi - \phi L \phi$

ϕ is tree induced voltages.

Total Duality Gap?

Gap: $|f|^2 - (2\phi^T\chi - \phi^T L\phi)$.

$= |f|^2 - 2\phi^T B^T f + \phi^T B^T B\phi$ where $B^T f = \chi$ and $L = B^T B$.
$= (f - B\phi)^T (f - B\phi)$.

Gap $= \sum_e (f(e) - \Delta_\phi(e))^2$ Difference between ϕ flow and f.

$\Delta_\phi(u,v) = \sum_{e \in P_{u,v}} -f(e)$. assume $f(e)$ is oriented around cycle.

For $e \in T$, $\Delta_\phi(e) = 0$. For $e \notin T$ $f(e) + \sum_{e \in P_e} f(e) = \Delta_{c_e}(f)$

Duality Gap: $\sum_{e \notin T} \sum_e \Delta_{c_e}(f)^2$

Total distance from optimal is cycle violations!

Claim: $E[\text{change in energy}|\mathit{Gap}] = \frac{\mathit{Gap}}{\tau}$

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Claim: E[change in energy$|$*Gap*$] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e}$

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1-1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1-1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time!

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau}\frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta^2_{C_e}}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta^2_{c_e}}{R_e} = -\sum_e \frac{\Delta^2_{c_e}}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! ! ! ! ! !

Claim: $E[\text{change in energy}|Gap] = \frac{Gap}{\tau}$ (τ is stretch of E in T.)

Duality Gap: $\sum_{e \notin T} \Delta_{C_e}(f)^2$

Choose edge e reduce energy by $-\frac{\Delta_{C_e}^2}{R_e}$.

Choose edge with probability $\frac{R_e}{\tau}$.

Expected reduction $-\sum_e \frac{R_e}{\tau} \frac{\Delta_{c_e}^2}{R_e} = -\sum_e \frac{\Delta_{c_e}^2}{\tau}$ $\square$

Duality Gap reduces by $(1 - 1/\tau)$ every iteration on expectation.

$O(\tau \log(n/\varepsilon))$ iterations gives $(1+\varepsilon)$ approximation.

$\tau = O(m \log n \log\log n)$ ...

$\tilde{O}(m)$ iterations

Iteration in $O(\log^2 n)$ time using balanced binary trees.

$\rightarrow \tilde{O}(m)$ time! ! ! ! ! ! ! ! ! ! ! !

Wrapup...

How to get low stretch tree?

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

Wrapup...

How to get low stretch tree?
  Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
    Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Wrapup...

How to get low stretch tree?
  Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
    Open: Get $O(m \log n)$ stretch?

How to do update along cycle?
  Answer: Data Structures.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Wrapup...

How to get low stretch tree?
Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
Open: Get $O(m \log n)$ stretch?

How to do update along cycle?
Answer: Data Structures.
Idea: Use a binary tree on paths.
Decompose tree into paths.

Geometric View.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Geometric View.

Cycles are constraints.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Geometric View.

Cycles are constraints.

Flow around cycle $= 0$.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Geometric View.

Cycles are constraints.

Flow around cycle $= 0$.

Each cycle update is projection into subspace defined by constraint.

Wrapup...

How to get low stretch tree?
Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
Open: Get $O(m \log n)$ stretch?

How to do update along cycle?
Answer: Data Structures.
Idea: Use a binary tree on paths.
Decompose tree into paths.

Geometric View.
Cycles are constraints.
Flow around cycle $= 0$.
Each cycle update is projection into subspace defined by constraint.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Geometric View.

Cycles are constraints.

Flow around cycle $= 0$.

Each cycle update is projection into subspace defined by constraint.

Better Algorithm:

Wrapup...

How to get low stretch tree?
  Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
    Open: Get $O(m \log n)$ stretch?

How to do update along cycle?
  Answer: Data Structures.
    Idea: Use a binary tree on paths.
      Decompose tree into paths.

Geometric View.
  Cycles are constraints.
  Flow around cycle $= 0$.
  Each cycle update is projection into subspace defined by constraint.

Better Algorithm:
  Recursive algorithm give $O(m\sqrt{\log n})$ iterations to halve error.

Wrapup...

How to get low stretch tree?
Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.
Open: Get $O(m \log n)$ stretch?

How to do update along cycle?
Answer: Data Structures.
Idea: Use a binary tree on paths.
Decompose tree into paths.

Geometric View.
Cycles are constraints.
Flow around cycle $= 0$.
Each cycle update is projection into subspace defined by constraint.

Better Algorithm:
Recursive algorithm give $O(m\sqrt{\log n})$ iterations to halve error.

Correspondence to Practice: Random sparsification of Cholesky factorization.

Wrapup...

How to get low stretch tree?

Answer: Elkin-Spielman-Teng, ...,Abraham, Newman.

Open: Get $O(m \log n)$ stretch?

How to do update along cycle?

Answer: Data Structures.

Idea: Use a binary tree on paths.

Decompose tree into paths.

Geometric View.

Cycles are constraints.

Flow around cycle $= 0$.

Each cycle update is projection into subspace defined by constraint.

Better Algorithm:

Recursive algorithm give $O(m\sqrt{\log n})$ iterations to halve error.

Correspondence to Practice: Random sparsification of Cholesky factorization.

Laplacian Systems are quite general: Climate, physics, SDD-matrices.

See you ...

Tuesday.