1 Complementary Slackness

Another intuition:

- $\min\{yb \mid yA \geq c\}$ (note **flipped sign**)
- suppose b points straight up.
- so goal is to follow gravity.
- put a ball in the polytope, let it fall
- stops at opt y (no local minima)
- stops because in physical equilibrium
- equilibrium exterted by forces normal to "floors"
- that is, aligned with the A_i (columns)
- thus $b = \sum A_i x_i$ for some **nonnegative** force coeffs x_i.
- in other words, x feasible for $\max\{cx \mid Ax = b, x \geq 0\}$
- also, only walls touching ball can exert any force on it
- thus, $x_i = 0$ if $yA_i > c_i$
- that is, $(c_i - yA_i)x_i = 0$
- thus, $cx = \sum(yA_i)x_i = yb$
- so x is dual optimal.

Leads to another idea: *complementary slackness*:

- given feasible solutions x and y, $cx - by \geq 0$ is *duality gap.*
- optimal iff gap 0 (good way to measure "how far off"
- Go back to original primal and dual forms
- rewrite dual: $yA + s = c$ for some $s \geq 0$ (that is, $s = c_j - yA_j$
- The following are equivalent for feasible x, y:
  - x and y are optimal
  - $sx = 0$
  - $x_j s_j = 0$ for all j
  - $s_j > 0$ implies $x_j = 0$
- proof:
  - $cx = by$ iff $(yA + s)x = (Ax)y$, so $sx = 0$

- if $sx = 0$, then since $s, x \geq 0$ have $s_j x_j = 0$ (converse easy)
- so of course $s_j > 0$ forces $x_j = 0$ (converse easy)

- basic idea: opt cannot have a variable x_j and corresponding dual constraint s_j slack at same time: one must be tight.

- Another way to state: in arbitrary form LPs, feasible points optimal if:

$$\begin{aligned} y_i(a_i x - b_i) &= 0 \forall i \\ (c_j - yA_j)x_j &= 0 \forall j \end{aligned}$$

- proof: note in definition of primal/dual, feasiblity means $y_i(a_i x - b_i) \geq 0$ (since $\geq$ constraint corresponds to nonnegative y_i). Also $(c_j - yA_j)x_j \geq 0$. Also,

$$\begin{aligned} \sum y_i(a_i x - b_i) + (c_j - yA_j)x_j &= yAx - yb + cx - yAx \\ &= cx - yb \\ &= 0 \end{aligned}$$

at opt. But since all terms are nonnegative, all must be 0

Let's take some duals.
Max-Flow min-cut theorem:

- primal problem: create infinite capacity (t, s) arc

$$\begin{aligned} P &= \max \sum_w x_{ts} \\ \sum_w x_{vw} - x_{wv} &= 0 \\ x_{vw} &\leq u_{vw} \\ x_{vw} &\geq 0 \end{aligned}$$

- dual problem:

$$\begin{aligned} D &= \min \sum_{vw} y_{vw} u_{vw} \\ y_{vw} &\geq 0 \\ z_v - z_w + y_{vw} &\geq 0 \\ z_t - z_s + y_{ts} &\geq 1 \end{aligned}$$

- note $y_{ts} = 0$ since otherwise dual infinite. so $z_t - z_s \geq 1$.

- rewrite as $z_w \leq z_v + y_{vw}$.

- deduce y_{vw} are edge lengths, z_v are distance upper bounds from source.
- might as well set z to distances from source (doesn't affect constraints)
- sanity check: mincut: assign length 1 to each mincut edge
- unfortunately, might have noninteger dual optimum.
- note z_i are distances, rescale to $z_s = 0$
- let $T = v \mid z_v \geq 1$ (so $s \notin W, t \in W$)
- use complementary slackness:
  - if (v, w) crosses out of T, then $z_v - z_w + y_v w \geq z_v - z_w > 1 - 1 = 0$
  - so $x_{vw} = u_{vw}$
  - on the orher hand, if (v, w) goes *into* T, then $y_{vw} \geq z_w - z_v > 0$, so $x_{vw} = 0$.
  - in other words: all leaving edges saturated, all coming edges empty.
- now just observe that value of flow equal value crossing cut equals value of cut.

Min cost circulation: change the objective function associated with max-flow.

- primal:

$$\begin{aligned} z &= \min \sum c_{vw} x_{vw} \\ \sum_w x_{vw} - x_{wv} &= 0 \\ x_{vw} &\leq u_{vw} \\ x_{vw} &\geq 0 \end{aligned}$$

- as before, dual: variable y_{vw} for capacity constraint on f_{vw}, z_v for balance.
- Change to primal min problem flips sign constraint on y_{vw}
- What does change in primal objective mean for dual? Different constraint bounds!

$$\begin{aligned} & \max \sum y_{vw} u_{vw} \\ z_v - z_w + y_{vw} &\leq c_{vw} \\ y_{vw} &\leq 0 \\ z_v & \quad \text{UIS} \end{aligned}$$

- rewrite dual: $p_v = -z_v$

$$\max \sum y_{vw} u_{vw}$$
$$\begin{aligned} y_{vw} &\leq 0 \\ y_{vw} &\leq c_{vw} + p_v - p_w = c_{ve}^{(p)} \end{aligned}$$

- Note: $y_{vw} \leq 0$ says the objective function is the sum of the **negative parts** of the reduced costs (positive ones get truncated to 0)
- Note: optimum ≤ 0 since of course can set $y = 0$. Since since zero circulation is primal feasible.
- complementary slackness.
  - Suppose $f_{vw} < u_{vw}$.
  - Then dual variable $y_{vw} = 0$
  - So $c_{ij}^{(p)} \geq 0$
  - Thus $c_{ij}^{(p)} < 0$ implies $f_{ij} = u_{ij}$
  - that is, all negative reduced cost arcs saturated.
  - on the other hand, suppose $c_{ij}^{(p)} > 0$
  - then constraint on z_{ij} is slack
  - so $f_{ij} = 0$
  - that is, all positive reduced arcs are empty.

2 Ellipsoid

We know a lot about structure. And we've seen how to verify optimality in polynomial time. Now turn to question: can we solve in polynomial time?
Yes, sort of (Khachiyan 1979):

- polynomial algorithms exist
- strongly polynomial do not.

2.1 Size of Problem

To talk formally about polynomial time, need to talk about size of problems.

- number n has size $\log n$
- rational p/q has size size(p)+size(q)
- size(product) is sum(sizes).

- dimension n vector has size n plus size of number
- $m \times n$ matrix similar: mn plus sizeof numbers
- size (matrix product) at most sum of matrix sizes
- our goal: polynomial time in size of input, measured this way

Claim: if A is $n \times n$ matrix, then $\det(A)$ is poly in size of A

- more precisely, twice the size
- proof by writing determinant as sum of permutation products.
- each product has size n times size of numbers
- $n!$ products
- so size at most size of ($n!$ times product) $\leq n \log n + n \cdot$size(largest entry).

Corollary:

- inverse of matrix is poly size (write in terms of cofactors)
- solution to $Ax = b$ is poly size (by inversion)

Claim: all vertices of LP have polynomial size.

- vertex is bfs
- bfs is intersection of n constraints $A_B x = b$
- invert matrix.

Now can prove that feasible alg can optimize a different way:

- use binary search on value z of optimum
- add constraint $cx \leq z$
- know opt vertex has poly number of bits
- so binary search takes poly (not logarithmic!) time
- not as elegant as other way, but one big advantage: feasiblity test over basically same polytope as before. Might have fast feasible test for this case.

2.2 Basic Idea of Ellipsoid

Define an ellipsoid

- generalizes ellipse
- write some $D = BB^T$ "radius"
- *center* z
- point set $\{(x-z)^T D^{-1}(x-z) \leq 1\}$
- note this is just a basis change of the unit sphere $x^2 \leq 1$.
- under transform $x \to Bx + z$

Outline of algorithm:

- goal: find a feasible point for $P = \{Ax \leq b\}$
- start with ellipse containing P, center z
- check if $z \in P$
- if not, use separating hyperplane to get 1/2 of ellipse containing P
- find a smaller ellipse containing this 1/2 of original ellipse
- until center of ellipse is in P.

Shrinking Lemma:

- Let $E = (z, D)$ define an n-dimensional ellipsoid
- consider separating hyperplane $ax \leq az$
- Define $E' = (z', D')$ ellipsoid:

$$\begin{aligned} z' &= z - \frac{1}{n+1}\frac{Da^T}{\sqrt{aDa^T}} \\ D' &= \frac{n^2}{n^2-1}(D - \frac{2}{n+1}\frac{Da^T aD}{aDa^T}) \end{aligned}$$

- then

$$\begin{aligned} E \cap \{x \mid ax \leq ez\} &\subseteq E' \\ \text{vol}(E') \leq e^{1/(2n+1)}\text{vol}(E) \end{aligned}$$

- for proof, first show works with $D = I$ and $z = 0$. new ellipse:

$$\begin{aligned} z' &= -1/n+1 \\ D' &= \frac{n^2}{n^2-1}(I - \frac{2}{n+1}I_{11} \end{aligned}$$

  and volume ratio easy to compute directly.

- for general case, transform to coordinates where $D = I$ (using new basis B), get new ellipse, transform back to old coordinates, get (z', D') (note transformation don't affect volume *ratios*.

So ellipsoid shrinks. Now prove 2 things:

- needn't start infinitely large
- can't get infinitely small

Starting size:

- recall bounds on size of vertices (polynomial)
- so coords of vertices are exponential but no larger
- so can start with sphere with radius exceeding this exponential bound
- this only uses polynomial values in D matrix.
- if unbounded, no vertices of P, will get vertex of box.

Ending size:

- convenient to assume that polytope full dimensional
- if so, it has $n + 1$ affinely indpendent vertices
- all the vertices have poly size coordinates
- so they contain a box whose volume is a poly-size number (computable as determinant of vertex coordinates)

Put together:

- starting volume $2^{n^{O(1)}}$
- ending volume $2^{-n^{O(1)}}$
- each iteration reduces volume by $e^{1/(2n+1)}$ factor
- so $2n + 1$ iters reduce by e
- so $n^O(1)$ reduce by $e^{n^{O(1)}}$
- at which point, ellipse doesn't contain P, contra
- must have hit a point in P before.

Justifying full dimensional:

- take $\{Ax \leq b\}$, replace with $P' = \{Ax \leq b + \epsilon\}$ for tiny ϵ
- any point of P is an interior of P', so P' full dimensional (only have interior for full dimensional objects)

- P empty iff P' is (because ϵ so small)
- can "round" a point of P' to P.

Infinite precision:

- built a new ellipsoid each time.
- maybe its bits got big?
- no.

2.3 Separation vs Optimization

Notice in ellipsoid, were only using one constraint at a time.

- didn't matter how many there were.
- didn't need to see all of them at once.
- just needed each to be represented in polynomial size.
- so ellipsoid works, even if huge number of constraints, so long as have *separation oracle:* given point not in P, find separating hyperplane.
- of course, feasibility is same as optimize, so can optimize with sep oracle too.
- this is on a polytope by polytope basis. If can separate a particular polytope, can optimize over that polytope.

This is very useful in many applications. e.g. network design.
Can also show that optimization implies separation:

- suppose can optimize over P
- then of course can find a point in P
- suppose $0 \in P$ (saves notation mess—just shift P)
- define $P^* = \{z \mid zx \leq 1 \ \forall x \in P\}$
- can separate over P^*:
  - given w, run OPT(p) with w objective
  - get x^* maximizing wx
  - if $wx^* \leq 1$ then $w \in P^*$
  - else $wx^* > 1 \geq x^* z \ \forall z \in P^*$ so x^* is separating hyperplane
  - since can separate P^*, can optimize it
- suppose want to separate y from P

- let $z = \text{OPT}(P^*, y)$.
- if $yz > 1$ then (since $z \in P^*$) we have $yz > 1$ but $xz \leq 1 \; \forall x \in P$ (separating hyperplane)
- if $y \leq 1$ then suppose $y \notin P$.
- then $ax \leq \beta$ for $x \in P$ but $ay > \beta$
- since $0 \in P$, $\beta \geq 0$
- if $\beta > 0$ then $\frac{a}{\beta}x \leq 1 \; \forall x \in P$ so its in P^* but $\frac{a}{\beta}y > 1$ so it is a better opt for y contra
- if $\beta = 0$ then $\lambda ax \leq 0 \leq 1 \forall \lambda > 0$ so $\lambda a \in P^*$ but $\lambda ay > 1$ for some $\lambda > 0$ so is better opt for y contra.