1 Linear Programming

1.1 Introduction

Problem description:

- motivate by min-cost flow
- bit of history
- everything is LP
- NP and coNP. P breakthrough.
- general form:
  - **variables**
  - **constraints:** linear equalities and inequalities
  - x **feasible** if satisfies all constraints
  - LP feasible if some feasible x
  - x **optimal** if optimizes objective over feasible x
  - LP is **unbounded** if have feasible x of arbitrary good objective value
  - **lemma:** every lp is infeasible, has opt, or is unbounded
  - (by compactness of R^n and fact that polytopes are closed sets).

Problem formulation:

- canonical form: $\min c^T x, Ax \geq b$
- matrix representation, componentwise $\leq$
- rows a_i of A are **constraints**
- c is **objective**
- any LP has transformation to canonical:
  - max/min objectives same
  - move vars to left, consts to right
  - negate to flip $\leq$ for $\geq$
  - replace $=$ by two $\leq$ and $\geq$ constraints
- standard form: $\min c^T x, Ax = b, x \geq 0$
  - slack variables
  - splitting positive and negative parts $x \to x^+ - x^-$
- $Ax \geq b$ often nicer for theory; $Ax = b$ good for implementations.

Some steps towards efficient solution:

- What does answer look like?
- Can it be represented effectively?
- Easy to verify it is correct?
- Is there a small proof of no answer?
- Can answer, nonanswer be found efficiently?

1.2 Linear Equalities

How solve? First review systems of linear equalities.

- $Ax = b$. when have solution?
- baby case: A is squre matrix with unique solution.
- demonstrate solution by giving x
- easy to verify correct!
- construct using, eg, Gaussian elimination.
- Suppose there's a answer. Can we write it down?
- discuss polynomiality, integer arithmetic later
- equivalent statements:
  - A invertible
  - A^T invertible
  - $\det(A) \neq 0$
  - A has linearly independent rows
  - A has linearly independent columns
  - $Ax = b$ has unique solution for every b
  - $Ax = b$ has unique solution for some b.

To talk formally about polynomial size/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)

What if A isn't square? Can we find an answer?

- Note we are asking if columns of A span b
- Find a maximal set of linearly independent columns
- These span b if A does.
- Extend to a basis by adding (independent) columns
- Now previous analysis applies
- Unique x
- It has better zero out all the added columns

Can we show there *isn't* an answer?

- $Ax = b$ has a *witness* for true: give x.
- How about a proof that there is no solution?
- note that "$Ax = b$" means columns of A span b.
- in general, set of points $\{Ax \mid x \in \Re^n\}$ is a **subspace**
- claim: no solution iff for some y, $yA = 0$ but $yb \neq 0$.
- proof: if $Ax = b$, then $yA = 0$ means $yb = yAx = 0$.
- if no $Ax = b$, means columns of A don't span b
- set of points $\{Ax\}$ is subspace not containing b

- find part of b perpendicular to subspace, call it y
- then $yb \neq 0$, but $yA = 0$,
- standard form LP asks for linear combo too, but requires that all coefficients of combo be nonnegative!

Algorithmic?

- Subtract projection of first column onto b
- Remainder of b must be spanned by other columns
- Recurse with other columns
- finish with a solution to an $Ax = b$ problem, so poly size.

1.3 Geometry

Polytopes

- canonical form: $Ax \geq b$ is an intersection of (finitely many) halfspaces, a **polytope**
- standard form: $Ax = b$ is an intersection of hyperplanes (thus a subspace), then $x \geq 0$ intersects in some halfspace. Also a polytope, but not full dimensional.
- polytope is **bounded** if fits inside some box.
- either formulation defines a **convex** set:
  - if $x, y \in P$, so is $\lambda x + (1 - \lambda)y$ for $\lambda \in 0, 1$.
  - that is, line from x to y stays in P.
- halfspaces define convex sets. Converse also true!
- let C be any convex set, $z \notin C$.
- then there is some a, b such that $ax \geq b$ for $x \in C$, but $az < b$.
- proof by picture. also true in higher dimensions (don't bother proving)
- deduce: every convex set is the intersection of the halfspaces containing it.

1.4 Basic Feasible Solutions

Again, let's start by thinking about structure of optimal solution.

- Can optimum be in "middle" of polytope?
- Not really: if can move in all directions, can move to improve opt.

Where can optimum be? At "corners."

- "vertex" is point that is not a convex combination of two others
- "extreme point" is point that is *unique* optimum in some direction

Basic solutions:

- A constraint $ax \leq b$ or $ax = b$ is *tight* or *active* if $ax = b$
- for n-dim LP, point is *basic* if (i) all equality constraints are tight and (ii) n linearly independent constraints are tight.
- in other words, x is at intersection of boundaries of n linearly independent constraints
- note x is therefore the unique intersection of these boundaries.
- a *basic feasible solution* is a solution that is basic and satisfies all constraints.

In fact, vertex, extreme point, bfs are *equivalent.*

- Proof left somewhat to reader.

Any standard form lp $\min cx,\ Ax = b,\ x \geq 0$ with opt has one at a BFS.

- Suppose opt x is not at BFS
- Then less than n tight constraints
- So at least one degree of freedom
- i.e, there is a (linear) subspace on which all those constraints are tight.
- In particular, some line through x for which all these constraints are tight.
- Write as $x + \epsilon d$ for some vector direction d
- Since x is feasible and other constraints *not* tight, $x + \epsilon d$ is feasible for small enough ϵ.
- Consider moving along line. Objective value is $cx + \epsilon cd$.
- So for either positive or negative ϵ, objective is *nonincreasing*, i.e. doesn't get worse.

- Since started at opt, must be no change at all—i.e., $cd = 0$.
- So can move in *either* direction.
- In at least one direction, some x_i is decreasing.
- Keep going till new constraint becomes tight (some $x_i = 0$).
- Argument can be repeated until n tight constraints, i.e. bfs
- Conclude: every standard form LP with an optimum has one at a bfs.
- Note convenience of using standard form: ensures bounded, so can reach bfs
- canonical form has oddities: e.g. $\max y \mid y \leq 1$.
- but any *bounded, feasible* LP has BFS optimum

Corollary:

- Actually showed, if x feasible, exists BFS with no worse objective.
- Note that in canconical form, might not have opt at vertex (optimize x_1 over (x_1, x_2) such that $0 \leq x_1 \leq 1$).
- But this only happens if LP is unbounded
- In particular, if opt is *unique*, it is a bfs.

Other characterizations of corner:

- "vertex" is point that is not a convex combination of two others
- "extreme point" is point that is *unique* optimum in some direction
- Previous proof shows extreme point implies BFS (because if cannot move to any other opt, must have n tight constraints).
- Also shows BFS if-and-only-if vertex:
  - if point is convex combo (not vertex), consider line through it
  - all points on it feasible
  - so don't have n tight constraints
  - conversely, if less than n tight constraints, they define feasible subspace containing line through point
  - so point is convex combo of points on line.
- To show BFS is extreme point, show point is unique opt for objective that is sum of normals to tight constraints.

Yields first algorithm for LP: try all bfs.

- How many are there?
- just choose n tight constraints out of m, check feasibility and objective
- Upper bound $\binom{m}{n}$

Also shows output is polynomial size:

- Let A' and correspoinding b' be n tight constraints (rows) at opt
- Then opt is (unique) solution to $A'x = b'$
- We saw last time that such an inverse is represented in polynomial size in input

(So, at least *weakly* polynomial algorithms seem possible)
OK, this is an exponential method for finding the optimum. Maybe we can do better if we just try to verify the optimum. Let's look for a way to prove that a given solution x is optimal.

2011 Lecture 10 end
2012 Lecture 12 end

Quest for nonexponential algorithm: start at an easier place: how decide if a solution is optimal?

- decision version of LP: is there a solution with opt$> k$?
- this is in NP, since can exhibit a solution (we showed poly size output)
- is it in coNP? Ie, can we prove there is no solution with opt$> k$? (this would give an optimality test)

2 Duality

What about optimality?

- Intro *duality*, strongest result of LP
- give proof of optimality
- gives max-flow mincut, prices for mincost flow, game theory, lots other stuff.

Motivation: find a **lower** bound on $z = \min\{cx \mid Ax = b, x \geq 0\}$.

- Standard approach: try adding up combos of existing equations
- try multiplying $a_i x = b_i$ by some y_i. Get $yAx = yb$
- If find y s.t. $yA = c$, then $yb = yAx = cx$ and we know opt (we inverted $Ax = b$)
- looser: if require $yA \leq c$, then $yb = yAx \leq cx$ is lower bound since $x_j \geq 0$
- so to get best lower bound, want to solve $w = \max\{yb \mid yA \leq c\}$.

- this is a new linear program, *dual* of original.
- just saw that dual is less than primal (weak duality)

Note: dual of dual is primal:

$$\begin{aligned}
\max\{yb : yA \le c\} &= \max\{by \mid A^T y \le c\} \\
&= -\min\{-by \mid A^T y + Is = c, s \ge 0\} \\
&= -\min\{-by^+ + by^- \mid A^T y + (-A^T)y^- + Is = c, y^+, y^-, s \ge 0\} \\
&= -\max\{cz \mid zA^T \le -b, z(-A^T) \le -b, Iz \le 0\} \\
&= \min\{cx \mid Ax = b, x \ge 0\} \quad (x = -z)
\end{aligned}$$

Weak duality: if P (min, opt z) and D (max, opt w) feasible, $z \ge w$

- $w = yb$ and $z = cx$ for some primal/dual feasible y, x
- x primal feasible ($Ax = b, x \ge 0$)
- y dual feasible ($yA \le c$)
- then $yb = yAx \le cx$

Note corollary:

- (restatement:) if P, D both feasible, then both bounded.
- if P feasible and unbounded, D not feasible
- if P feasible, D either infeasible or bounded
- in fact, only 4 possibilities. both feasible, both infeasible, or one infeasible and one unbounded.
- **notation:** P unbounded means D infeasible; write solution $-\infty$. D unbounded means P infeasilbe, write solution ∞.

3 Strong Duality

Strong duality: if P or D is feasible then $z = w$

- includes D infeasible via $w = -\infty$)

Proof by picture:

- $\min\{yb \mid yA \ge c\}$ (note: **flipped sign**)
- suppose b points straight up.
- imagine ball that falls down (minimize height)
- 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)
- but those floors need to cancel "gravity" $-b$
- thus $b = \sum A_i x_i$ for some **nonnegative** force coeffs x_i.
- in other words, x feasible for $\min\{cx \mid Ax = b, x \geq 0\}$
- also, only floors 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.

Let's formalize.

- Consider optimum y
- WLOG, ignore all loose constraints (won't need them)
- And if any are redundant, drop them
- So at most n tight constraints remain
- and all linearly independent (A has full rank).
- and since those constraints are tight, $yA = c$

Claim: Exists x, $Ax = b$

- Suppose not? Then "duality" for linear equalities proves exists z, $zA = 0$ but $zb <> 0$.
- WLOG $zb < 0$ (else negate it)
- So consider $y + z$.
- $A(y + z) = Ay + Az = Ay$, so feasible
- $b(y + z) = by + bz < by$, so better than opt! Contra.

Claim: $yb = cx$

- Just said $Ax = b$ in dual
- In primal, all (remaining) constraints are tight, so $yA = c$

- So $yb = yAx = cx$

Claim: $x \geq 0$

- Suppose not.
- Then some $x_i < 0$
- Let $c' = c + e_i$
- In other words, moving i^{th} constraint $yA_i \geq c_i$ "upwards"
- Consider solution to $y'A = c'$
- Exists solution (since A is full rank)
- And $c' \geq c$, so $y'A = c'$ is *feasible* for original constraints $yA \geq c$
- Value of objective is $y'b = y'Ax = c'x$
  - We assumed $x_i < 0$, and *increased* c_i
  - So $c'x < cx$
  - So got better value than opt. Contradiction!
- Intuition: x_i is telling us how much opt will change if we "tighten" i^{th} constraint

Neat corollary: Feasibility or optimality: which harder?

- given optimizer, can check feasiblity by optimizing arbitrary func.
- Given feasibility algorithm, can optimize by combining primal and dual.

Interesting note: knowing dual solution may be useless for finding optimum (more formally: if your alg runs in time T to find primal solution given dual, can adapt to alg that runs in time $O(T)$ to solve primal without dual).
2011 End of Lecture 11

3.1 Rules for duals

General dual formulation:

- primal is

$$\begin{aligned} z &= \min c_1x_1 + c_2x_2 + c_3x_3 \\ A_{11}x_1 + A_{12}x_2 + A_{13}x_3 &= b_1 \\ A_{21}x_1 + A_{22}x_2 + A_{23}x_3 &\geq b_2 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 &\leq b_3 \\ x_1 &\geq 0 \\ x_2 &\leq 0 \\ x_3 &\quad UIS \end{aligned}$$

(UIS emphasizes unrestricted in sign)

- means dual is

$$\begin{aligned} w &= \max y_1 b_1 + y_2 b_2 + y_3 b_3 \\ y_1 A_{11} + y_2 A_{21} + y_3 A_{31} &\leq c_1 \\ y_1 A_{12} + y_2 A_{22} + y_3 A_{32} &\geq c_2 \\ y_1 A_{13} + y_2 A_{23} + y_3 A_{33} &= c_3 \\ y_1 &\quad UIS \\ y_2 &\geq 0 \\ y_3 &\leq 0 \end{aligned}$$

- In general, variable corresponds to constraint (and vice versa):

PRIMAL	minimize	maximize	DUAL
constraints	$\geq b_i$ $\leq b_i$ $= b_i$	≥ 0 ≤ 0 free	variables
variables	≥ 0 ≥ 0 free	$\leq c_j$ $\leq c_j$ $= c_j$	constraints

Derivation:

- remember lower bounding plan: use $yb = yAx \leq cx$ relation.
- If constraint is in "natural" direction, dual variable is positive.
- We saw A_{11} and x_1 case. $x_1 \geq 0$ ensured $yAx_1 \leq c_1 x_1$ for **any** y
- If some $x_2 \leq 0$ constraint, we want $yA_{12} \geq c_2$ to maintain rule that $y_1 A_{12} x_2 \leq c_2 x_2$
- If x_3 unconstrained, we are only safe if $yA_{13} = c_3$.
- if instead have $A_{21} x_1 \geq b_2$, any old y won't do for lower bound via $c_1 x_1 \geq y_2 A_{21} x_1 \geq y_2 b_2$. Only works if $y_2 \geq 0$.
- and so on (good exercise).
- This gives weak duality derivation. Easiest way to derive strong duality is to transform to standard form, take dual and map back to original problem dual (also good exercise).

Note: tighter the primal, looser the dual

- (equality constraint leads to unrestricted var)
- adding primal constraints creates a new dual variable: more dual flexibility

3.2 Shortest Paths

A dual example:

- shortest path is a dual (max) problem:

$$\begin{aligned} w &= \max d_t - d_s \\ d_j - d_i &\leq c_{ij} \end{aligned}$$

- constraints matrix A has ij rows, i columns, ± 1 entries (draw)
- what is primal? unconstrained vars, give equality constraints, dual upper bounds mean vars must be positive.

$$\begin{aligned} z &= \min \sum y_{ij} c_{ij} \\ y_{ij} &\geq 0 \end{aligned}$$

  thus

$$\sum_j y_{ji} - y_{ij} = 1(i = s), -1(i = t), 0 \text{ ow}$$

  It's the minimum cost to send one unit of flow from s to t!

4 Complementary Slackness

Leads to another idea: *complementary slackness*:

- given feasible solutions x and y, $cx - yb \geq 0$ 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, $s_j = c_j - yA_j$)
- 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 $x_j = 0$
- We saw this in duality analysis: only tight constraints "push" on opt, giving nonzero dual variables.
- proof:

- $cx = by$ iff $(yA + s)x = y(Ax)$, iff $sx = 0$
- if $sx = 0$, then since $s, x \geq 0$ have $s_j x_j = 0$ (converse easy)
- so $s_j > 0$ 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.

- Generalize to 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 dual, $\geq$ constraint in primal (min) corresponds to nonnegative y_i.
  - thus, feasiblity means $y_i(a_i x - b_i) \geq 0$.
  - similarly, $\leq$ constraint in dual corresponds to nonnegative x_j in primal
  - so feasibility means $(c_j - yA_j)x_j \geq 0$.
  - 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 just argued all terms are nonnegative, all must be 0
  - conversely, if all are 0, then $cx = by$, so we are optimal

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

- modify to circulation to simplify
- 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: vars z_v dual to balance constraints, y_{vw} dual to capacity constraints.

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

- Think of y_{vw} as "lengths"
- 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 "distances"
- In particular, can substract z_s from everything without changing feasibility (subs cancel)
- Now z_v is upper bound on distance from source to v.
- So, are trying to maximize source-sink distance
  - Good justification for shortest aug path, blocking flows
- sanity check: mincut: assign length 1 to each mincut edge
- unfortunately, might have noninteger dual optimum.
- let $S = v \mid z_v < 1$ (so $s \in S, t \notin S$)
- use complementary slackness:
  - if (v, w) *leaves* S, then $y_{vw} \geq z_w - z_v > 0$, so $x_{vw} = u_{vw}$, (tight) i.e. (v, w) saturated.
  - if (v, w) *enters* S, then $z_v > z_w$. Also know $y_{vw} \geq 0$; add equations and get $z_v + y_{vw} > z_w$ i.e. slack.
  - so $x_{wv} = 0$
  - in other words: all leaving edges saturated, all coming edges empty.
- now just observe that value of flow equals 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!

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

- rewrite dual: $p_v = -z_v$

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

- 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 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.

2011: End of Lecture 12

5 Algorithms

5.1 Simplex

vertices in standard form/bases:

- Without loss of generality make A have full row rank (define):
  - find basis in rows of A, say $a_1, \ldots, a_k$
  - any other a_ℓ is linear combo of those.
  - so $a_\ell x = \sum \lambda_i a_i x$
  - so better have $b_l = \sum \lambda_i a_i$ if any solution.
  - if so, anything feasible for $a_1, \ldots, a_\ell$ feasible for all.
- m constraints $Ax = b$ all tight/active
- given this, need $n - m$ of the $x_i \geq 0$ constraints
- also, need them to form a basis with the a_i.
- **write matrix** of tight constraints, first m rows then identity matrix
- need linearly independent rows
- equiv, need linearly independent columns
- but columns are linearly independent iff m columns of A including all corresp to nonzero x are linearly independent
- gives other way to define a vertex: x is vertex if
  - $Ax = b$
  - m linearly independent columns of A include all $x_j \neq 0$

  This set of m columns is called a *basis*.
- x_j of columns called *basic* set B, others *nonbasic* set N
- given bases, can compute x:
  - A_B is basis columns, $m \times m$ and full rank.
  - solve $A_B x_B = b$, set other $x_N = 0$.
  - note can have many bases for same vertex (choice of 0 x_j)

Summary: x is vertex of P if for some basis B,

- $x_N = 0$
- A_B nonsingular
- $A_B^{-1} b \geq 0$

Simplex method:

- start with a basic feasible soluion
- try to improve it
- rewrite LP: $\min c_B x_B + c_N x_N$, $A_B x_B + A_N x_N = b$, $x \geq 0$
- B is basis for bfs
- since $A_B x_B = b - A_N x_N$, so $x_B = A_B^{-1}(b - A_N x_N)$, know that

$$\begin{aligned} cx &= c_B x_B + c_N x_N \\ &= c_B A_B^{-1}(b - A_N x_N) + c_N x_N \\ &= c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N \end{aligned}$$

- *reduced cost* $\tilde{c}_N = c_N - c_B A_B^{-1} A_N$
- if no $\tilde{c}_j < 0$, then increasing any x_j increases cost (may violate feasiblity for x_B, but who cares?), so are at optimum!
- if some $\tilde{c}_j < 0$, can increase x_j to decrease cost
- but since x_B is func of x_N, will have to stop when x_B hits a constraint.
- this happens when some x_i, $i \in B$ hits 0.
- we bring j into basis, take i out of basis.
- we've moved to an *adjacent* basis.
- called a *pivot*
- **show picture**

Notes:

- Need initial vertex. How find?
- maybe some $x_i \in B$ already 0, so can't increase x_j, just pivot to same obj function.
- could lead to cycle in pivoting, infinite loop.
- can prove exist noncycling pivots (eg, lexicographically first j and i)
- no known pivot better than exponential time
- note traverse path of edges over polytope. Unknown what shortest such path is
- Hirsh conjecture: path of $m - d$ pivots exists.
- even if true, simplex might be bad because path might not be monotone in objective function.
- certain recent work has shown $n^{\log n}$ bound on path length

5.2 Simplex and Duality

- defined *reduced costs* of nonbasic vars N by

$$\tilde{c}_N = c_N - c_B A_B^{-1} A_N$$

  and argued that when all $\tilde{c}_N \geq 0$, had optimum.

- Define $y = c_B A_B^{-1}$ (so of course $c_B = y A_B$)
- nonegative reduced costs means $c_N \geq y A_N$
- put together, see $yA \leq c$ so y is dual feasible
- but, $yb = c_B A_B^{-1} b = c_B x_B = cx$ (since $x_N = 0$)
- so y is dual optimum.
- more generally, y measures duality gap for current solution!
- another way to prove duality theorem: prove there is a terminating (non cycling) simplex algorithm.

5.3 Polynomial Time Bounds

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 unknown.

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.

6 Ellipsoid

Lion hunting in the desert.

- bolzano wierstrauss theorem—proves certain sequence has a subsequence with a limit by repeated subdividing of intervals to get a point in the subinterval.
- The Bolzano-Weierstrass method: Divide the desert by a line running from north to south. The lion is then either in the eastern or in the western part. Let's assume it is in the eastern part. Divide this part by a line running from east to west. The lion is either in the northern or in the southern part. Let's assume it is in the northern part. We can continue this process arbitrarily and thereby constructing with each step an increasingly narrow fence around the selected area. The diameter of the chosen partitions converges to zero so that the lion is caged into a fence of arbitrarily small diameter.

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.

Consider sphere case, separating hyperplane $x_1 = 0$

- try center at $(a, 0, 0, \ldots)$
- Draw picture to see constraints
- requirements:

- $d_1^{-1}(x_1 - a)^2 + \sum_{i>1} d_i^{-1} x_i^2 \leq 1$
- constraint at $(1, 0, 0)$: $d_1^{-1}(x - a)^2 = 1$ so $d_1 = (1 - a)^2$
- constraint at $(0, 1, 0)$: $a^2/(1-a)^2 + d_2^{-1} = 1$ so $d_2^{-1} = 1 - a^2/(1-a)^2 \approx 1 - a^2$

- What is volume? about $(1 - a)/(1 - a^2)^{n/2}$
- set a about $1/n$, get $(1 - 1/n)$ volume ratio.

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

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

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

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

  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.

6.1 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.
2011 Lecture 13 end

7 Interior Point

Ellipsoid has problems in practice ($O(n^6)$ for one). So people developed a different approach that has been extremely successful.
What goes wrong with simplex?

- follows edges of polytope
- complex stucture there, run into walls, etc
- interior point algorithms stay away from the walls, where structure simpler.
- Karmarkar did the first one (1984); we'll descuss one by Ye

7.1 Potential Reduction

Potential function:

- Idea: use a (nonlinear) potential function that is minimized at opt but also enforces feasibility
- use gradient descent to optimize the potential function.
- Recall standard primal $\{Ax = b, x \geq 0\}$ and dual $yA + s = c, s \geq 0$.
- duality gap sx

- Use *logarithmic barrier function*

$$G(x,s) = q \ln xs - \sum \ln x_j - \sum \ln s_j$$

  and try to minimize it (pick q in a minute)

- first term forces duality gap to get small
- second and third enforce positivity
- note barrier prevents from ever hitting optimum, but as discussed above ok to just get close.

Choose q so first term dominates, guarantees good G is good xs

- $G(x,s)$ small should mean xs small
- xs large should mean $G(x,s)$ large
- write $G = \ln(xs)^q / \prod x_j s_j$
- $xs > x_j s_j$, so $(xs)^n > \prod x_j s_j$. So taking $q > n$ makes top term dominate, $G > \ln xs$

How minimize potential function? Gradient descent.

- have current (x,s) point.
- take linear approx to potential function around (x,s)
- move to where linear approx smaller $(-\nabla_x G)$
- deduce potential also went down.
- crucial: can only move as far as linear approximation accurate

First wants big q, second small q. Compromise at $n + \sqrt{n}$, gives $O(L\sqrt{n})$ iterations.
Must stay feasible:

- Have gradient $g = \nabla_x G$
- since potential not minimized, have reasonably large gradient, so a small step will improve potential a lot. **picture**
- want to move in direction of G, but want to stay feasilbe
- project G onto nullspace(A) to get d
- then $A(x+d) = Ax = b$
- also, for sufficiently small step, $x \geq 0$
- potential reduction proportional to length of d

- problem if d too small
- In that case, move s (actually y) by $g - d$ which will be big.
- so can either take big primal or big dual step
- why works? Well, d (perpendicular to A) has $Ad = 0$, so good primal move.
- converseley, part spanned by A has $g - d = wA$,
- so can choose $y' = y + w$ and get $s' = c - Ay' = c - Ay - (g - d) = s - (g - d)$.
- note $dG/dx_j = s_j/(xs) - 1/x_j$
- and $dG/ds_j = x_j/(xs) - 1/s_j = (x_j/s_j)dG/dx_j \approx dG/dx_j$