CSE 589 -- Lecture 4

When you start on a long journey, trees are trees, water is water, and mountains are mountains. After you have gone some distance, trees are no longer trees, water no longer water, mountains no longer mountains. But after you have traveled a great distance, trees are once again trees, water is once again water, mountains are once again mountains.

-- Zen teaching

Announcements

- Gilligan's Island Rule
- Final

Plan for Today

- Linear programming
  - see handout for references
- NP-completeness, NP-complete problems
  - Cormen, Leiserson, Rivest, Chapter 36

Linear Programming

- The process of minimizing a linear objective function subject to a finite number of linear equality and inequality constraints.
- Example applications:
  - airline crew scheduling
  - manufacturing and production planning
  - portfolio selection
  - telecommunications network design
- "Few problems studied in computer science have greater application in the real world."

Feasible Set

- Each linear inequality divides n-dimensional space into two halfspaces, one where the inequality is satisfied, and one where it's not.
- Feasible Set : solutions to a family of linear inequalities.
- Family of linear cost functions, gives family of parallel hyperplanes (lines in 2D, planes in 3D, etc.). Want to find one of minimum cost => must occur at corner of feasible set.

An example: The diet problem

- Trying to decide on lowest cost diet that provides sufficient amount of protein, with two choices:
  - steak: 2 units of protein/pound, $3/pound
  - peanut butter: 1 unit of protein/pound, $2/pound
- In proper diet, need 4 units protein/day.

Goal: minimize $2x + 3y$

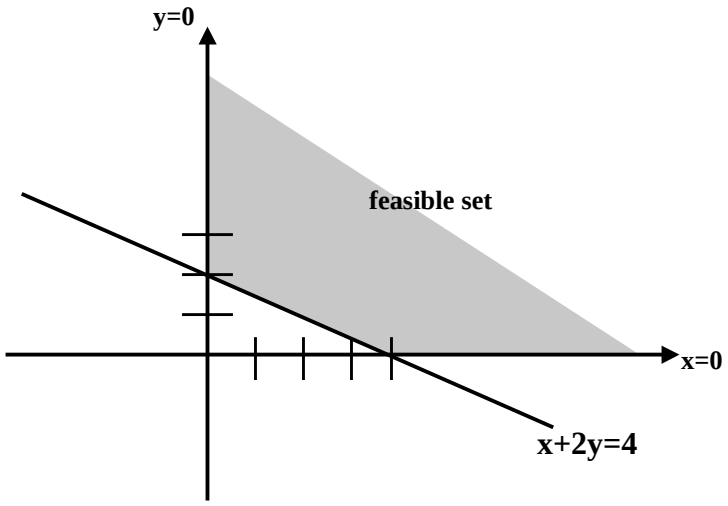
subject to constraints:

$x + 2y >= 4$

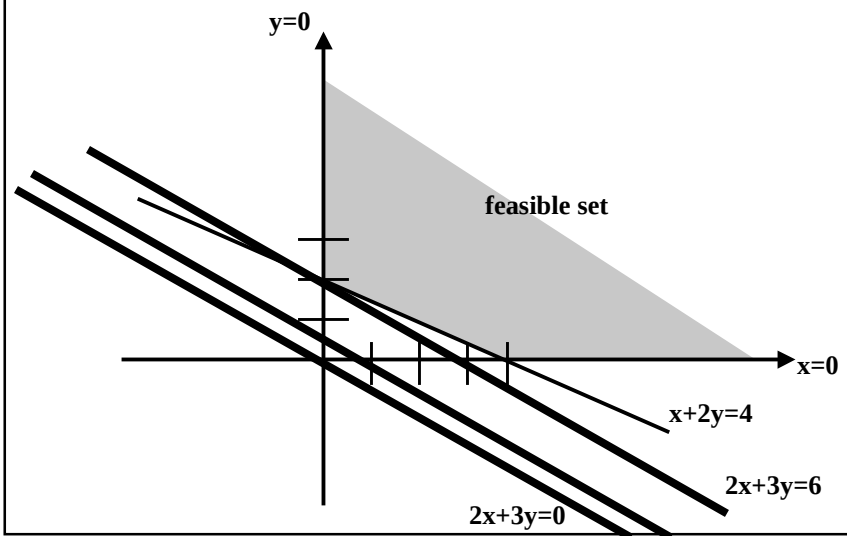
x = # pounds peanut butter/day in optimal diet >=0

y = # pounds steak/day in optimal diet >=0

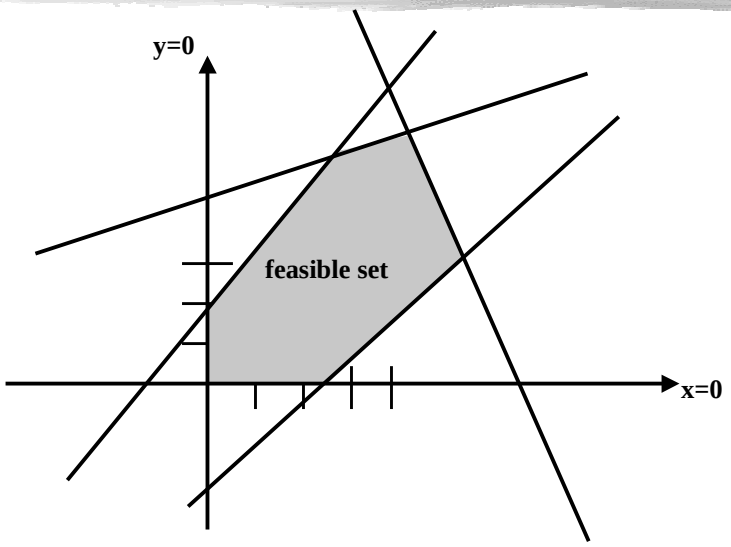
Visually…
x= peanut butter, y = steak



Optimal vector occurs at corner of feasible set!



Optimal vector occurs at corner of feasible set!



The Feasible Set

- Intersection of a set of half-spaces, called a **polyhedron**.
- If it's bounded and nonempty, it's a **polytope**

3 cases:

- feasible set is empty
- cost function is unbounded on feasible set.
- cost has a minimum (or maximum) on feasible set.

First two cases very uncommon for genuine problems in economics and engineering.

General Form of a Linear Program.

Minimize $b_1y_1 + b_2y_2 + \ldots + b_my_m$

subject to $\Sigma_{1<=i<=m} a_{ij}y_i >= c_j$ $\quad j=1..n$

$y_i >= 0$ $\quad i=1..m$

or

Maximize $c_1x_1 + c_2x_2 + \ldots + c_nx_n$

subject to $\Sigma_{1<=j<=n} a_{ij}x_j <= b_j$ $\quad i=1..m$

$x_j >= 0$ $\quad j=1..n$

Easy to deal with any linear inequalities/equalities.

- y not constrained to be positive =>
  replace y by $(y^+ - y^-)$ and $y^+, y^- >= 0$
- $\Sigma_{1<=i<=m} a_{ij}y_i <= c_j$ => multiply by (-1)
- $\Sigma_{1<=i<=m} a_{ij}y_i = c_j$ => replace with
  $\Sigma_{1<=i<=m} a_{ij}y_i >= c_j$ and $\Sigma_{1<=i<=m} -a_{ij}y_i >= -c_j$

The Simplex Method

- Phase I : locate a corner of the feasible set.
  - corner = intersection of n different planes (in n dimensions)
- Phase II: move from corner to corner along the edges of the feasible set -- always go along an edge that is guaranteed to decrease the cost.
  - Edge = intersection of n-1 different planes
- When reach a local minimum (maximum), you've found the optimum.

Simplex Algorithm: An Example

Maximize 5x + 4y + 3z

subject to 2x+ 3y + z <= 5

4x + y + 2z <= 11

3x + 4y + 2z <= 8

x,y,z >=0.

Step 0: convert inequalities into equalities by introducing slack variables a,b,c.

Define: a = 5-2x-3y-z => a >=0

b = 11-4x-y-2z => b >=0

c = 8-3x-4y-2z => c >=0

F = 5x+4y + 3z, objective function

Example of Simplex Method, continued.

Step 1: Find initial feasible solution:

x=0,y=0,z=0 => a=5, b=11, c=8 => F=0.

Step 2: Find feasible solution with higher value of F

For example, can increase x to get F=5x.

How much can we increase x?

a = 5-2x-3y-z >=0 => x <= 5/2 **most stringent**

b = 11-4x-y-2z >=0 => x <= 11/4

c = 8-3x-4y-2z >=0 => x <= 8/3

=> increase x to 5/2 =>**F= 25/2**, a=0, b=1, c=1/2

Example of Simplex Method, continued.

Want to keep doing this, need to get back into state where x,b,c on l.h.s. of equations.

a = 5-2x-3y-z => x= 5/2 -3/2 y -1/2 z - 1/2 a (*)

Substituting (*) into other equations:

b = 11-4x-y-2z >=0 => b = 1 + 5y + 2a

c = 8-3x-4y-2z >=0 => c = 1/2 + 1/2 y -1/2 z + 3/2 a

F = 5x+4y + 3z => F= 25/2 - 7/2 y + 1/2 z - 5/2 a

In order to increase F again, should increase

Example of Simplex Method, continued.

How much can we increase z?

x= 5/2 -3/2 y -1/2 z - 1/2 a => z <= 5

b = 1 + 5y + 2a => no restriction

c = 1/2 + 1/2 y -1/2 z + 3/2 a => z <= 1 **most stringent (^)**

Setting z = 1 yields

x=2, y=0, z=1, a=0, b = 1, c = 0.

F= 25/2 - 7/2 y + 1/2 z - 5/2 a => F= 13.

Again, construct system of equations.

From (^) z = 1 + y + 3a - 2c.

Example of Simplex Method, continued.

Substituting back into other equations:

z = 1 + y + 3a - 2c.

x= 5/2 -3/2 y -1/2 z - 1/2 a => x = 2-2y-2a + c

b = 1 + 5y + 2a => b= 1 + 5y + 2a

F = 25/2 - 7/2 y + 1/2 z - 5/2 a => F = 13 - 3y -a - c

And we're done.

What were we doing?

- Each time we had a feasible solution we were at a corner == a meeting point of 3 different planes. We chose 3 variables and set them to 0, and made sure remaining constraints were satisfied.
- Then we moved along an edge == a meeting point of 2 different planes obtained from corner by removing one equation. Moved along associated edge until arrived at a new corner.

The Simplex Algorithm and beyond....

- In practice, quite fast -- typically only O(m) **pivots** (where m is the number of constraints)
- In worst case, exponential.
- For a long time, it wasn't known if there was a polynomial time algorithm until….
- Khachian's algorithm "The Mathematical Sputnik of 1979"
  - exterior-point method
- And then there was Karmakar (1984)…
  - interior-point method
  - competes viably with simplex algorithm on real-world problems.

A Central Result of LP Theory Duality Theorem

- Every linear program has a dual
- If the original is a minimization, the dual is a maximization and vice versa
- Solution of one leads to solution of other

Primal: Minimize **cx** subject to A**x** >= **b**, **x**>=0

Dual: Maximize **yb** subject to **y**A **<=c**, **y**>=0

If one has optimal solution so does other, and their values are the same.

Primal: Minimize cx subject to Ax >= b, x>=0
Dual: Maximize yb subject to yA <=c, y>=0

- In the primal, **c** in cost function and **b** was in the constraint. In the dual, reversed.
- Inequality sign is changed and minimization turns to maximization.
- Example: minimize 2x + 3y subject to

  x+2y>=4, 2x + 5y >= 1, x - 3y >= 2, x>=0, y>=0
- Dual problem: maximize 4p +q + 2r subject to

  p+2q + r <= 2, 2p+5q -3r <= 3, p,q,r >=0

Simple Example

- Diet problem: minimize 2x + 3y subject to
  x+2y>=4, x>=0, y>=0
- Dual problem: maximize 4p subject to
  p<=2, 2p <=3, p>=0
- Dual: the problem faced by a druggist who sells synthetic protein, trying to compete with peanut butter and steak
- He wants to maximize the price p, subject to constraints:
  - synthetic protein must not cost more than protein
  - price must be non-negative or he won't sell any
  - revenue to druggist will be 4p
- Solution: p<= 3/2 => objective function = 4p = 6
- Not coincidence that it's = minimal cost in original problem.

More general diet problem

Minimum problem has n unknowns, n foods to be eaten in amounts $x_1, \ldots, x_n$

m constraints represent m required vitamins

- entry a_{ij} is amount of i-th vitamin in j-th food.
- i-th row of Ax >=b forces the diet to include that vitamin in at least the amount b_i.
- $c_1x_1 + \ldots + c_nx_n$ = cost of diet (c_j is cost of j-th food.)

Dual -- druggist selling vitamin pills rather than food.

- Prices adjustable as long as nonnegative.
- Key constraint -- on each food can't charge more than grocer.
- Since food j contains vitamins in amount a_{ij} , the druggist's price for the equivalent in vitamins can't exceed c_j => yA <= c.
- Can then sell amount b_i of each vitamin for a total income of $y_1b_1+..+ y_mb_m$

Example

Minimize c(st) x(st) + c(pb) x(pb) subject to

- prot(st) x(st) + prot(pb) x(pb) >= q(prot)
- carbo(st) x(st) + carbo(pb) x(pb) >= q(carbo)
- x(st), x(pb) >= 0
- c(n) = per unit cost of food n, x(n) quantity of food n to purchase per day, prot(n) = units protein per unit of food n, carbo (n) = units carbo per unit n q(prot) = protein units per day needed, q(carbo) = carbo units per day

Dual -- druggist selling synthetic protein and carbohydrate pills

maximize q(prot) y(s-prot) + q(carbo) y(s-carbo) subject to

- y(s-prot) prot(st) + y(s-carbo) carbo(st) <= c(st)
- y(s-prot) prot(pb) + y(s-carbo) carbo(pb) <= c(pb)
- y(s-prot), y(s-carbo) >= 0
- Can then sell amount q(prot) of protein per day and q(carbo) of carbo per day.

What's going on?

- Notice: feasible sets completely different for primal and dual, but nonetheless an important relation between them.
- Duality theorem says that in the competition between the grocer and the druggist the result is always a tie.
- Optimal solution to primal tells purchaser what to do.
- Optimal solution to dual fixes the natural prices at which economy should run.
- The diet x and vitamin prices y are optimal when
  - grocer sells zero of any food that is priced above its vitamin equivalent.
  - druggist charges 0 for any vitamin that is oversupplied in the diet.

Duality Theorem

- Druggist's max revenue = Purchasers min cost

One direction of duality easy , for any feasible x, y:

q(prot) y(s-prot) + q(carbo) y(s-carbo) <= x(st) c(st) + x(pb) c(pb)

Since each food can be replaced by its vitamin equivalent, with no increase in cost, all adequate diets must be at least as expensive as any price the druggist would charge.

For [x(st), x(pb)] and [y(s-prot), y(s-carbo)] feasible =>

[prot(st) x(st) + prot(pb) x(pb)] y(s-prot) >= q(prot) y(s-prot)
[carbo(st) x(st) + carbo(pb) x(pb)] y(s-carb) >= q(carbo) y(s-carb)

Sum two inequalities =>

q(prot) y(s-prot) + q(carbo) y(s-carbo) <=
[prot(st) x(st) + prot(pb) x(pb)] y(s-prot) +
[carbo(st) x(st) + carbo(pb) x(pb)] y(s-carbo)
= x(st) [y(s-prot) prot(st) + y(s-carbo) carbo(st)] +
x(pb) [y(s-prot) prot(pb) + y(s-carbo) carbo(pb)]
<= x(st) c(st) + x(pb) c(pb)

When they are equal, they both must be optimal.

Practical Use of Duality

- Sometimes simplex algorithm (or other algorithms) will run faster on the dual than on the primal.
- Can be used to bound how far you are from optimal solution.
- Important implications for economists.

Application: Optimal Pipeline

- A piece of data of size D goes through a pipeline of n stages.
- Each stage has associated
  - o_i -- overhead of i-th stage
  - b_i -- bandwidth of i-th stage (bits/sec)
- How should data be broken up into k pieces, not necessarily of equal size, so as to minimize time through the pipeline?

Summary of Linear Programming

- Of great practical importance to solve linear programs:
  - they model important practical problems
    - production, approximating the solution of inconsistent equations, manufacturing, network design, flow control, resource allocation.
  - solving an LP is often an important component of solving or approximating the solution to an **integer linear programming problem.**
- The simplex algorithm works very well in practice.
- One problem where you really do not want to roll your own code.

New Topic: Easy vs. Hard Problems

- The standard definition of a tractable problem is one that can be solved in time that is polynomial in the size of the input. Why?
  - Very few practical problems require time which is high-degree polynomial
  - equivalent on different models of computation
  - nice closure properties
- This class of problems called P.
- NP -- class of problems whose solution can be verified in polynomial time. Framed as "decision problems".
- **Many many many many many many many many many many important problems are in NP (in fact, NP-complete.)**

Hardest problems in NP are NP-complete

- Provably NP-complete problems --

  If any one can be solved in polynomial time, then all of them can.
- Right now, no known efficient algorithm known. Biggest open problem in CS:

  P = NP?
- Heuristics/approximation algorithms typically used. Sometimes solved exactly too -- depends on the application area.

Most notorious hard graph problem

- **Traveling Salesman Problem**
  - Input description: A weighted graph G
  - Output description: Find the cycle of minimum cost that visits each of the vertices exactly once.
- Example: optimization of tool path for manufacturing equipment. E.g., robot arm assigned to solder all connections on printed circuit board.

Clique

- Input description: A graph G=(V,E)
- Problem description: What is the largest subset S of V such that for all x,y in S, (x,y) in E.
- Example:

  IRS detection of organized tax fraud. Graph has vertices corresponding to submitted tax forms, and edges between pairs that are suspiciously similar.

Independent Set

- Input description: A graph G=(V,E)
- Problem description: What is the largest subset S of V such that no pair of vertices in S has an edge between them.
- Example:
  - Identifying location for a new franchise service such that no two locations are close enough to compete with each other.
  - Highest capacity code for given communication channel.

Hamiltonian Cycle

- Input description: A graph G=(V,E)
- Problem description: Find an ordering of the vertices such that adjacent vertices are connected by an edge and each vertex is visited exactly once.

- Example:

  Pattern recognition -- longest path in graph.

  Triangle strip problem.

Graph Partition

- Input description: A weighted graph G=(V,E) and integers j,k
- Problem description: Partition vertices into two subsets such that each subset has size at most j, and the weight of edges connecting the two subsets is at most k.

- Example:

  VLSI layout

Vertex Coloring

- Input description: A graph G=(V,E)
- Problem description: Color vertices using the minimum number of colors such that for each edge (i,j) in E, vertices i and j have different colors
- Example:

  Register allocation for compilers.

Steiner Tree

- Input description: A graph G=(V,E) and a subset T of the vertices V
- Problem description: Find smallest tree connecting all the vertices of T
- Example:

  Network design and wiring layout.

Integer Linear Programming

- Input description: A linear functional **cx**, a set of linear constraints **Ax >= b** and a set of non-negative variables **x** that can take on only integer values, say 0 or 1.
- Problem description: Find integer values for the variables **x** that minimizes the linear functional **cx** subject to the linear constraints.
- Example: absolutely everything

Satisfiability

- Input description: Given a boolean formula in conjunctive normal form
- Problem description: Find a truth assignment for the variables that causes the formula to evaluate to 1.
- Example: digital design, hardware testing,….

How you prove a problem P is NP-complete.

1. Prove it's in NP
2. Select a known NP-complete problem P'.
3. Describe a polynomial time computable algorithm that computes a function f mapping every instance of P' to an instance of P.
4. Prove that for every yes-instance of P' maps to a yes-instance of P, and every no-instance of P' maps to a no-instance of P.

Called a **reduction.**

Example: Showing Independent Set is NP-complete.

Your turn! Prove that TSP is NP-complete.

Solving NP-complete Problems

- Exactly
  - branch and bound
- Approximately
  - using LP (e.g., randomized rounding)
  - local search
  - simulated annealing
  - ….