Three main directions to solve NP-hard problems:
- Integer Programming Techniques
- Heuristics
- Approximation Algorithms

On time-accuracy tradeoff schedule:
- brute force
- integer programming
- approximation algorithms
- heuristics

Most accuracy --- least accuracy
Worst time --- best time

We'll consider these directions on the example of the Traveling Salesman Problem (TSP)

Given: $G = (V, E)$, distance function $C: E \rightarrow \mathbb{R}$
Goal: Find a tour which visits all nodes such that the sum of traveled distances is minimized.

Most discrete models can be formulated as integer programs.

Integer program for TSP:
Define $x_{ij} = 1$ if arc $i \rightarrow j$ is included in tour

$$\min \sum_{i,j \in E} C_{ij} x_{ij}$$

s.t. $\sum_{j \in V} x_{ij} = 2$ for each node $i$

$$\sum_{i \in V} x_{ij} = 2 \forall SCV$$

$x_{ij} \in \{0, 1\}$ for each $(i,j) \in E$

Integer programs are much harder to solve than linear programs.
CPF solns are not guaranteed to be integer solns → simplex doesn't work.

What's number of solutions for TSP?
If $|V| = n$ then $\exists \frac{(n-1)!}{2}$ different solns.

Suppose a computer can evaluate one soln in $10^3$ sec.
In that case, if $n = 23$ then it will take 178 centuries to solve the problem by brute force (evaluation of all solutions).

Thus, need more efficient algorithms.

First Direction:
Integer Programming Techniques

Characteristics:
- guarantee optimal soln most of the time
- might be very time-inefficient
- use LP formulations and Simplex as subroutine

Integer Program (integer linear program) has
- linear objective function
- linear constraints
- integer variables

Consider LP relaxation: the same program but substitute $x_{ij} \in \{0, 1\}$ by $0 \leq x_{ij} \leq 1$.
LP relaxation doesn't guarantee to return integer solution (the optimal solution might consist of many $x_{ij} = \frac{1}{2}$).

But using LP relaxations can develop techniques to get optimal integer soln:
- branch and bound
- cutting plane

These techniques don't guarantee time efficiency; might take a lot of time to get to optimality.

Next time: Heuristics
Approximation Algorithms