Objectives

- Understand the idea of an approximation algorithm
- Understand local search and its variants, esp. with regard to solving the TSP

NP-Complete

- Recall our Definition: A decision problem \( A \) is NP-complete iff
  - \( A \in \text{NP} \)
  - For all \( B \in \text{NP} \), \( B \leq^f_A \)
- Can you use this definition directly to prove that a problem is in NP-Complete?
- Why not?

Proving a Problem is NP-Complete

- Let \( X \) be an NP-Complete problem
  - e.g., 3CNF SAT
- Consider a decision problem \( Z \) in NP such that \( X \leq^f Z \)
- Then?

Proving a Problem is NP-Complete

- Let \( X \) be an NP-Complete problem
  - e.g., 3CNF SAT
- Consider a decision problem \( Z \) in NP such that \( X \leq^f Z \)
- Thus, \( Z \) is also NP-Complete

Then our strategy to prove \( Z \) is NP-Complete is the following:

1. Show that \( Z \) is in NP
   - Easy
2. Show that a known NP-Complete problem can be reduced to \( Z \)
   - More fun!
Hard Problems

- What can we do if we have found a hard problem?

Options

- Brute force:
  - Enumerate all and check or score
- State-space search:
  - Back-tracking (Brute-force or Intelligent)
- Branch & Bound – for optimization
- Approximation algorithms:
  - $3\leq \text{cost}(*) = \epsilon$, where $*$ is an optimal solution and $\tilde{*}$ is an approximate solution produced by the algorithm

Solution-space search for optimization:

- Local search
- Local search + random restarts
- Deterministic annealing
- Simulated annealing
- Genetic algorithms (LS + fitness & cross-over)

Local Search

TSP: 2-change

2-change Neighborhood

Local Optimality
3-change

Local Search with 3-change

Local Search Landscape

Local Search + Random Restarts

Simulated Annealing

let $s$ be any starting solution
repeat
  randomly choose a solution $s'$ in the neighborhood of $s$
  if $\Delta = \text{cost}(s') - \text{cost}(s)$ is negative:
    replace $s$ by $s'$
  else:
    replace $s$ by $s'$ with probability $e^{-\Delta/T}$.

Plus some logic to lower the temperature and detect convergence.
Simulated Annealing Landscape

Genetic Algorithms
- For some optimization problems it is desirable to search the space more broadly.
- not just focus on a good solution based on an initial start state.
- Genetic/Evolutionary Algorithms work well in many of these situations.
- Performs a more exploratory search of the search space than standard local search variations.

Genetic Algorithm

Procedure GA(t)
1. \( t = 0 \);
2. Initialize Population \( P(t) \);
3. Evaluate \( P(t) \);
4. Do:
   1. \( t = t + 1 \);
   2. Parent_Selection \( P(t) \);
   3. Recombine \( P(t) \);
   4. Mutate \( P(t) \);
   5. Evaluate \( P(t) \);
   6. Survive \( P(t) \);
5. Until 1. Sufficiently “good” individuals are discovered or 2. many iterations with no improvement

End