# CS540 Introduction to Artificial Intelligence

## Lecture 20

Young Wu

Based on lecture slides by Jerry Zhu and Yingyu Liang

August 8, 2022

# Local Search

## Motivation

- Local search is about searching through a state space by iteratively improving the cost to find an optimal or near-optimal state.
- The successor states are called the neighbors (sometimes move set).
- The assumption is that similar (nearby) solutions have similar costs.

# Local Search Application

## Motivation

- Optimization problems (gradient descent methods are all local search methods)
- Traveling salesman
- Boolean satisfiability (SAT)
- Scheduling

# Hill Climbing (Valley Finding)

## Description

- Start at a random state.
- Move to the best neighbor state (one of the successors).
- Stop when all neighbors are worse than the current state.
- The idea is similar to gradient descent.

# Simulated Annealing

## Description

- Each time, a random neighbor is generated.
- If the neighbor has a lower cost, move to the neighbor.
- If the neighbor has a higher cost, move to the neighbor with a small probability.
- Stop until bored.
- It is a version of Metropolis-Hastings Algorithm.

# Annealing

## Definition

- The annealing process of heated solids.
- Anneal: to subject (glass or metal) to a process of heating and slow cooling to toughen and reduce brittleness.
- Alloys manage to find a near global minimum energy state when heated and then slowly cooled.

# Acceptance Probability

## Definition

- The probability of moving to a state with a higher cost should be small.

1. Constant: $p = 0.1$
2. Decreases with time: $p = \frac{1}{t}$
3. Decreases with time and as the energy difference increases: $p = \exp\left(-\frac{|f(s') - f(s)|}{T(t)}\right)$

- The algorithm corresponding to the third idea is called simulated annealing. The Temperature function $T(t)$ should be a decreasing in time t (iteration number).

# Temperature

## Definition

- $T$ represents temperature which decreases over time. For example, the temperature can change arithmetically or geometrically.

$$T(t+1) = \max\{T(t) - 1, 1\},\ T(0) = \text{ large}$$
$$T(t+1) = 0.9T(t),\ T(0) = \text{ large}$$

- High temperature: almost always accept any $s'$.
- Low temperature: first choice hill climbing.

# Simulated Annealing

## Algorithm

- Input: state space $S$, temperature function $T$, and cost function $f$.
- Output: $s^{\star} \in S$ that minimizes $f(s)$.
- Start at a random state $s_0$.
- At iteration $t$, generate a random neighbor $s'$, and update the state according to the following rule.

$$s_{t+1} = \begin{cases} s' & \text{if } f(s') < f(s_t) \\ s' & \text{with probability } \exp\left(-\dfrac{|f(s') - f(s_t)|}{T(t)}\right) \\ s_t & \text{otherwise} \end{cases}$$

# Simulated Annealing Performance

## Discussion

- Use hill-climbing first.
- Neighborhood design is the most important.
- In theory, with infinitely slow cooling rate, Simulated Annealing finds global minimum with probability 1.

# Genetic Algorithm

## Description

- Start with a fixed population of initial states.
- Find the successors by:

1. Cross over.
2. Mutation.

# Reproduction Probability

## Definition

- Each state in the population has probability of reproduction proportional to the fitness. Fitness is the opposite of the cost: higher cost means lower fitness. Use $F$ to denote the fitness function, for example, $F\left(s\right) = \dfrac{1}{f\left(s\right)}$ is a valid fitness function.

$$p_i = \dfrac{F\left(s_i\right)}{\displaystyle\sum_{j=1}^{N} F\left(s_j\right)}, i = 1, 2, ..., N$$

- A pair of states are selected according to the reproduction probabilities (using CDF inversion).

# Cross Over

## Definition

- The states need to be encoded by strings.
- Cross over means swapping substrings.
- For example, the children of 10101 and 01010 could be the same as the parents or one of the following variations.

$$(11010, 00101), (10010, 01101)$$

$$(10110, 01001), (10100, 01011)$$

# Mutation

## Definition

- The states need to be encoded by strings.
- Mutation means randomly updating substrings. Each character is changed with small probability $q$, called the mutation rate.
- For example, the mutated state from 000 could stay the same or be one of the following.

$$\text{one of } 001, 010, 100, \text{ with probability } q\,(1-q)^2$$
$$\text{one of } 011, 101, 110, \text{ with probability } q^2\,(1-q)$$
$$\text{and } 111, \text{ with probability } q^3$$

# Cross Over, Modifications

## Definition

- The previous cross over method is called 1 point cross over.
- It is also possible to divide the string into $N$ parts. The method is called $N$ point cross over.
- It is also possible to choose each character from one of the parents randomly. The method is called uniform cross over.

# Mutation, Modifications

## Definition

- For specific problems, there are ways other than flipping bits to mutate a state.

1. Two-swap: ABCDE to EBCDA
2. Two-interchange: ABCDE to EDCBA

# Genetic Algorithm, Part I

## Algorithm

- Input: state space $S$ represented by strings $s$ and cost function $f$ or fitness function $F$.
- Output: $s^{\star} \in S$ that minimizes $f(s)$.
- Randomly generate $N$ solutions as the initial population.

$$s_1, s_2, ..., s_N$$

- Compute the reproduction probability.

$$p_i = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)}, i = 1, 2, ..., N$$

# Genetic Algorithm, Part II

## Algorithm

- Randomly pick two states according to $p_i$, say $s_a, s_b$. Randomly select a cross over point $c$, swap the strings.

$$s'_a = s_a\left[0...c\right) s_b\left[c...m\right)$$
$$s'_b = s_b\left[0...c\right) s_a\left[c...m\right)$$

- Randomly mutate each position of each state $s_i$ with a small probability (mutation rate).

$$s'_i\left[k\right] = \begin{cases} s_i\left[k\right] & \text{with probability } 1-q \\ \text{random} & \text{with probability } q \end{cases}, k = 1, 2, ..., m$$

- Repeat with population $s'$.

# Variations

## Discussion

- Parents can survive.
- Use ranking instead of $F(s)$ to compute reproduction probabilities.
- Cross over random bits instead of chunks.

# Genetic Algorithm Performance

## Discussion

- Use hill-climbing first.
- State design is the most important.
- In theory, cross over is much more efficient than mutation.