CS540 Introduction to Artificial Intelligence
Lecture 20

Young Wu
Based on lecture slides by Jerry Zhu and Yingyu Liang

July 5, 2021

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, $F(s) = \frac{1}{f(s)}$ is a valid fitness function.

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

- 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 TSP Example

Discussion

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.

$$s'_a = s_a\,[0...c)\; s_b\,[c...m)$$
$$s'_b = s_b\,[0...c)\; s_a\,[c...m)$$

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

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

- 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.

Game Tree

Motivation

- The initial state is the beginning of the game.
- There are no goal states, but there are multiple terminal states in which the game ends.
- Each successor of a state represents a feasible action (or a move) in the game.
- The search problem is to find the terminal state with the lowest cost (or usually the highest reward).

Adversarial Search

Motivation

- The main difference between finding solutions of games and standard search problems or local search problems is that part of the search is performed by an opponent adversarially.
- Usually, the opponent wants to maximize the cost or minimize the reward from the search. This type of search problems is called adversarial search.
- In game theory, the solution of a game is called an equilibrium. It is a path in which both players do not want to change actions.

Backward Induction

Motivation

- Games are usually solved backward starting from the terminal states.
- Each player chooses the best action (successor) given the (already solved) optimal actions of all players in the subtrees (called subgames).