CS410: Artificial Intelligence 2021 Fall
Homework 3: Local Search & Adversarial Search & MDP
Due date: 23:59:59 (GMT +08:00), November 6 2021

Wendi Chen 519021910071

1. **Local Search**. The traveling salesman problem (TSP) is the problem of finding the shortest route to visit a set of cities exactly once and return to the starting city. Describe how to use genetic algorithm for TSP. Propose a state representation, the corresponding crossover and mutation, and the fitness function.

   Solution. We assume that there are n cities.

   - **State Representation.** Each city is denoted as a distinct number from 1 to n. Then we represent the route by using a sequence $S^{(k)} = \{x_1^{(k)}, \cdots, x_n^{(k)}\}$ where $x_i^{(k)}$ means the the i-th visited city in the k-th sequence. Besides, we have $x_i^{(k)} \in \{1 \cdots n\}$ and $x_i^{(k)} \neq x_j^{(k)} (i \neq j)$.
   - **Crossover.** For one pair of sequences $S^{(k_1)}, S^{(k_2)}$, random choose two crossover points i and j $(i \leq j)$. Then, we exchange the values of the subsequences $\{x_i^{(k_1)}, \cdots, x_j^{(k_1)}\}$ and $\{x_i^{(k_2)}, \cdots, x_j^{(k_2)}\}$. Here, we'll find some repeated values in the sequences and we use the following algorithm to eliminate repeated values. Let's take $S^{(k_1)}$ for example.
     (a) If there exists $x_p^{(k_1)} (1 \leq p \leq i-1$ or $j+1 \leq p \leq n)$ in $S^{(k_1)}$ that satisfies $x_p^{(k_1)} = x_q^{(k_1)} (i \leq q \leq j)$, then we let $x_p^{(k_1)} \leftarrow x_q^{(k_2)}$.
     (b) Back to (a) until there's no such $x_p^{(k_1)}$.
   - **Mutation (with some prob).** For sequence $S^{(k)}$, randomly choose two cities $x_i^{(k)}, x_j^{(k)} (i \neq j)$ and exchange their values.
   - **Fitness Function.** We use the reciprocal of the route length as the fitness function. We denote the distance between $x_i^{(k)}$ and $x_j^{(k)}$ as $d(x_i^{(k)}, x_j^{(k)})$. Then we have

$$f(S^{(k)}) = \frac{1}{d(x_1^{(k)}, x_2^{(k)}) + d(x_2^{(k)}, x_3^{(k)}) + \cdots + d(x_{n-1}^{(k)}, x_n^{(k)}) + d(x_n^{(k)}, x_1^{(k)})}$$

2. **Game Tree**. Prove that with a positive linear transformation of leaf values (i.e., transforming a value x to $ax + b$ where $a > 0$), the choice of move remains unchanged in a game tree, even when there are chance nodes.

   Solution. For a positive linear transformation $T(x) = ax + b (a > 0)$, if we always prefer x_i with smaller i when there is a tie, we have the following properties.

$$\max_i x_i = x_p \Leftrightarrow \max_i T(x_i) = T(x_p)$$

$$\min_i x_i = x_p \Leftrightarrow \min_i T(x_i) = T(x_p)$$

$$argmax_i x_i = p \Leftrightarrow argmax_i T(x_i) = p$$

$$\sum_i \alpha_i T(x_i) = T(\sum_i \alpha_i x_i) \text{ where } \sum_i \alpha_i = 1$$

   Then, we'll finish the proof by induction. We assume that the game tree has n layers and we want to prove that if the original value of the node in i-th layer is v_i, then after the transformation, the value will be $v_i' = T(v_i)$. We prove from bottom to top.

   - For the n-th layer, according to the problem description, we have $v' = T(v)$.
   - We assume that for the $i+1$-th$(2 \leq i < n)$ layer, $v_{i+1}' = T(v_{i+1})$. Then, according to the properties of positive transformation, no matter the node in i-th layer is a max, min or chance node, we have $v_i' = T(v_i)$.
   - For the first layer, according to the properties of positive transformation, if the choice of move is a, then after the transformation, the choice will still be a.

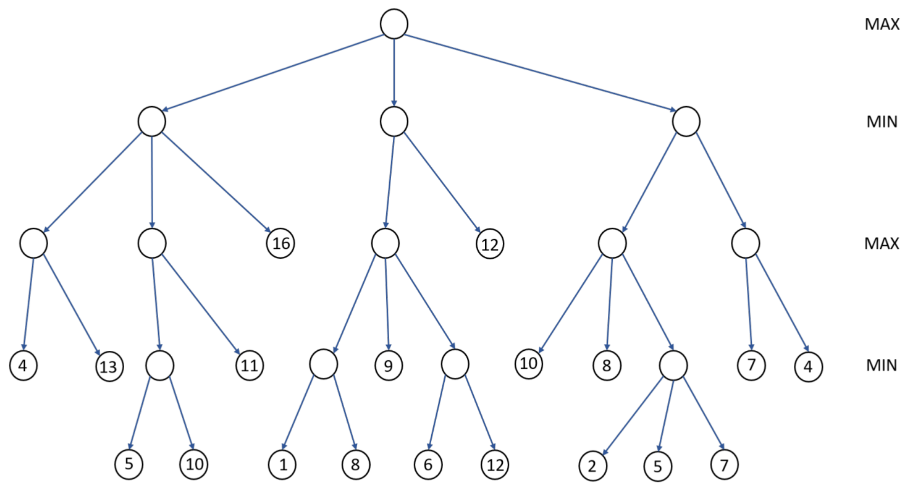

Figure 1: Problem 3.

3. **Alpha-Beta Pruning**. Consider the above game tree.

   (a) Compute the minimax value for each node using Minimax algorithm.

   (b) Prune the game tree using Alpha-Beta pruning algorithm. Provide the final alpha and beta values computed at the root, each internal node visited, and at the top of pruned branches. Provide the pruned branches. Assume child nodes are visited from left to right.

   Solution.

   (a) Please refer to Figure 2.

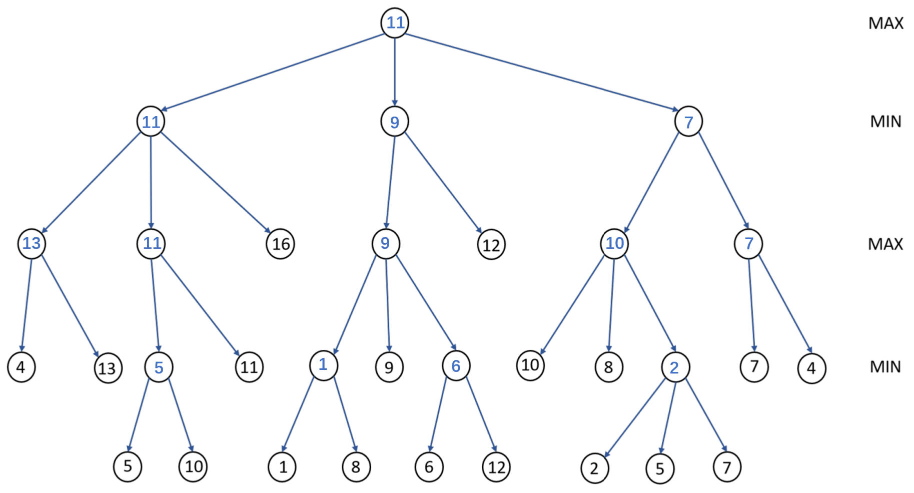

Figure 2: Minimax value for each node.

   (b) I mainly follow the algorithm on Page 46, Lecture 5. The result is shown in Figure 3.

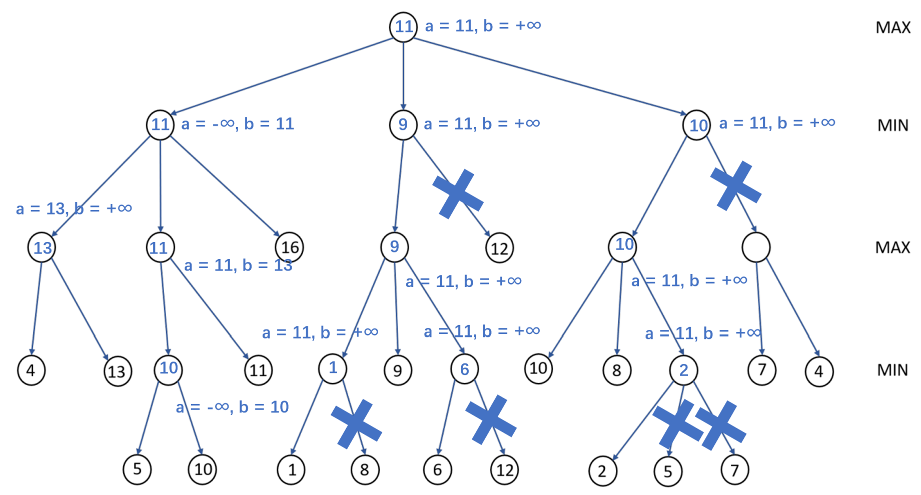

Figure 3: Result of Alpha-Beta pruning algorithm.

4. **Racing Problem**. Consider the racing problem in Page 17, Lecture 6.

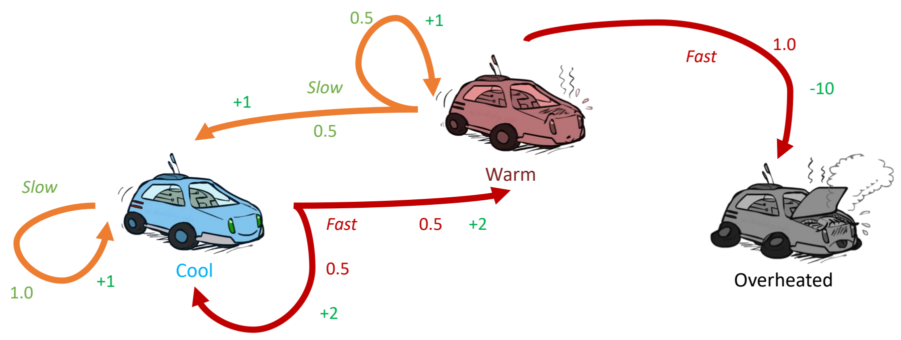

Figure 4: Problem 4.

Assume there is a discount factor $0 < \gamma < 1$ in the MDP of this problem. Calculate $V^*(s)$ for each state s and $Q^*(s, a)$ for each q-state (s, a) in this problem.

Solution. By using value iteration, we can easily find that

$$V^*(Cool) > 0, V^*(Warm) > 0, V^*(Overheated) = 0$$

Then, we only need to solve the following system of equations

$$V^*(Cool) = \max\{0.5 \times (2 + \gamma V^*(Cool)) + 0.5 \times (2 + \gamma V^*(Warm)), 1 + \gamma V^*(Cool)\}$$

$$V^*(Warm) = \max\{-10 + \gamma V^*(Overheated), 0.5 \times (1 + \gamma V^*(Warm)) + 0.5 \times (1 + \gamma V^*(Cool))\}$$

According to the conclusion derived from value iteration, we have

$$V^*(Warm) = 0.5 \times (1 + \gamma V^*(Warm)) + 0.5 \times (1 + \gamma V^*(Cool))$$

- If $V^*(Cool) = 1 + \gamma V^*(Cool)$, then we have

  $$V^*(Cool) = \frac{1}{1-\gamma}, V^*(Warm) = \frac{1}{1-\gamma}$$

  However, in this condition, we will have $0.5 \times (2 + \gamma V^*(Cool)) + 0.5 \times (2 + \gamma V^*(Warm)) > 1 + \gamma V^*(Cool)$, which contradicts.

- If $V^*(Cool) = 0.5 \times (2 + \gamma V^*(Cool)) + 0.5 \times (2 + \gamma V^*(Warm))$, then we have

  $$V^*(Cool) = \frac{4-\gamma}{2-2\gamma}, V^*(Warm) = \frac{2+\gamma}{2-2\gamma}$$

  And we can verify that $0.5 \times (2 + \gamma V^*(Cool)) + 0.5 \times (2 + \gamma V^*(Warm)) > 1 + \gamma V^*(Cool)$.

Thus, we have $V^*(Cool) = \frac{4-\gamma}{2-2\gamma}, V^*(Warm) = \frac{2+\gamma}{2-2\gamma}, V^*(Overheated) = 0$.

According to the definition of Q^*, we have

$$Q^*(Cool, Fast) = V^*(Cool) = \frac{4-\gamma}{2-2\gamma}$$

$$Q^*(Cool, Slow) = 1 + \gamma V^*(Cool) = \frac{2+2\gamma-\gamma^2}{2-2\gamma}$$

$$Q^*(Warm, Fast) = -10$$

$$Q^*(Warm, Slow) = V^*(Warm) = \frac{2+\gamma}{2-2\gamma}$$

5. **Convergence of Policy Iteration**. Prove the policy improvement method can indeed improve policies and then prove the convergence of policy iteration.