ROB311 Quiz 3

Hanhee Lee

April 4, 2025

Contents

Turn-Taking Multi-Agent Decision Algorithms

1 Zero-Sum Turn-Based Games

Summary: In a **zero-sum turn-based games**, we assume that

- **Agents and Environment:**
  - there are two agents, called the **maximizer** and **minimizer**
  - the environment is always in one of a discrete set of states, $\mathcal{S}$
  - a subset of the states, $\mathcal{T} \subseteq \mathcal{S}$, are terminal states
  - there is only one decision maker for each non-terminal state, $s \in \mathcal{S} \setminus \mathcal{T}$
  - For each non-terminal state, $s \in \mathcal{S} \setminus \mathcal{T}$, the decision-maker has a discrete set of actions, $\mathcal{A}(s)$
- **Decision Process:** At time-step t, the decision-maker will:
  - **Observe:** Observe the state s_t
  - **Select:** Select an action $a_t \in \mathcal{A}(s_t)$
  - **Move:** Make the move (s_t, a_t)
- **State Transitions:**
  - Environment transitions to a deterministic state, s_{t+1}, based on a stationary fn,

$$s_{t+1} = \text{tr}(s_t, a_t)$$

  - Once a terminal state is reached (if $s_{t+1} \in \mathcal{T}$), the maximizer obtains a reward for the final transition based on a reward fn, $r(\cdot, \cdot, \cdot)$:

$$r(s_t, a_t, s_{t+1}) = \text{maximizer's reward for reaching state } s_{t+1}$$

$$-r(s_t, a_t, s_{t+1}) = \text{minimizer's reward for reaching state } s_{t+1}$$

Warning:

- Maximizer is trying to maximize the reward of agent 1
- Minimizer is trying to minimize the reward of agent 1 (i.e. maximize the reward of agent 2)

1.1 α/β Pruning

Motivation: Don't explore the entire game tree by pruning branches that are unreachable under perfect play.

Definition: For each state s:

- α_s: Maximum value at s thus far (initially $-\infty$)
- β_s: Minimum value at s thus far (initially $+\infty$)

1.1.1 α Cuts

Definition: If the **maximizer** is the turn-taker at s, then α_s increases to the maximum value of s's successors as they are explored, and $\beta_s = \beta_{\text{parent}(s)}$.

- If α_s increases beyond β_s, then s unreachable under perfect play.

1.1.2 β Cuts

Definition: If the **minimizer** is the turn-taker at s, then β_s decreases to the minimum value of s's successors as they are explored, and $\alpha_s = \alpha_{\text{parent}(s)}$.

- If β_s decreases beyond α_s, then s unreachable under perfect play.

1.2 Examples

1.2.1 Min-Max Algorithm

Example:

- **Given:** Cavemen is injured from his hunt. He has extra food, but needs medicine.
  - He meets another caveman who is willing to trade.

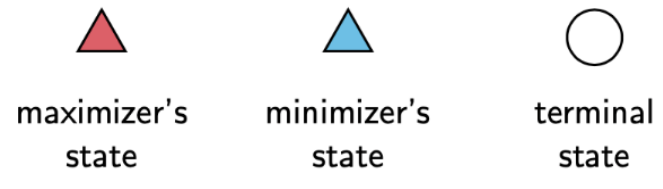


Figure 1: States

Figure 2: Actions

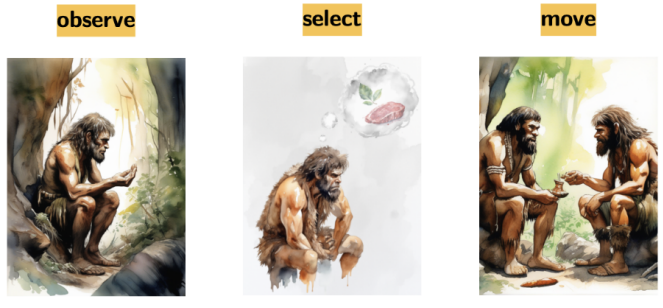


Figure 3: Decision Process

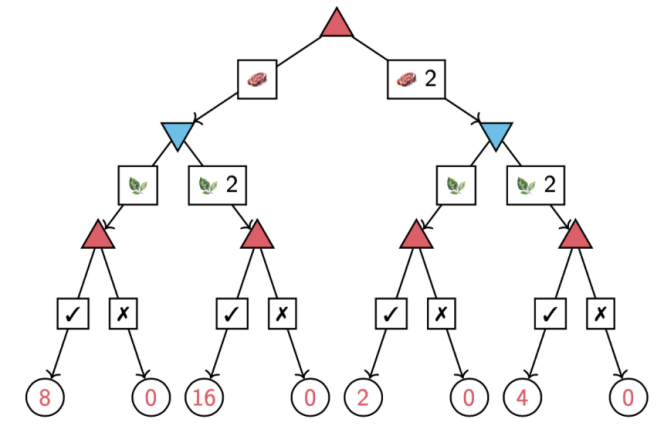


Figure 4: Game Tree

  - States
    - Red triangle: Maximizing agent
    - Blue triangle: Minimizing agent
    - White circles with #s: terminal states
    - Rewards: In red b/c it's for the maximizer. The minimizer's reward is the negative of the maximizer's reward.

  - Actions: Square boxes are actions
- **Solution:** Backtracking through the game tree, we can find the optimal path for the maximizer and minimizer.
  - **Maximizer Turn:**
    - **Left Branch:**
      - Far Left: Accept to get reward of 8,
      - Mid Left: Accept to get reward of 16,
    - **Right Branch:**
      - Mid Left: Accept to get reward of 2,
      - Far Left: Accept to get reward of 4
  - **Minimizer Turn:**
    - **Left Branch:**
      - L: 1 medicine to make maximizer get reward of 8,
    - **Right Branch:**
      - L: 1 medicine to make maximizer get reward of 2
  - **Maximizer Turn:** 1 food to make maximizer get reward of 8 b/c going right will make maximizer get reward of 2
  - **Optimal Path:** Therefore, the optimal path will be LLL b/c the maximizer will get a reward of 8, while the minimizer will reduce the reward from 16 to 8.
    - Assume boths agents play optimally, this will be the path taken.

1.2.2 α Cuts

Example:

- Explored 14, 12 and now $\beta_{\text{parent}(s)} = \beta_s = 5$, so this will be compared for α_s until $\alpha_s > \beta_s$ b/c then s unreachable under perfect play.
- Iterate:
  - $\alpha_s = -\infty < \alpha'_s = 2 \rightarrow \alpha_s = 2$, but $\alpha_s = 2 < \beta_s = 5$
  - $\alpha_s = 2 < \alpha'_s = 4 \rightarrow \alpha_s = 4$, but $\alpha_s = 4 < \beta_s = 5$
  - $\alpha_s = 4 < \alpha'_s = 9 \rightarrow \alpha_s = 9$, and $\alpha_s = 9 > \beta_s = 5$, therefore, prune all the other branches that haven't been explored yet in the children of s paths

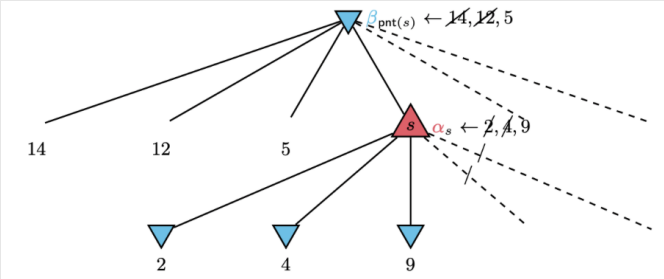


Figure 5

1.2.3 β Cuts

Example:

- Explored 4,6, and now $\alpha_{\text{parent}(s)} = \alpha_s = 7$, so this will be compared for β_s until $\beta_s < \alpha_s$ b/c then s unreachable under perfect play.
- Iterate:
  - $\beta_s = +\infty > \beta'_s = 9 \rightarrow \beta_s = 9$, but $\beta_s = 9 > \alpha_s = 7$
  - $\beta_s = 9 > \beta'_s = 8 \rightarrow \beta_s = 5$, but $\beta_s = 8 > \alpha_s = 7$
  - $\beta_s = 8 > \beta'_s = 3 \rightarrow \beta_s = 3$, and $\beta_s = 3 < \alpha_s = 7$, therefore, prune all the other branches that haven't been explored yet in the children of s paths

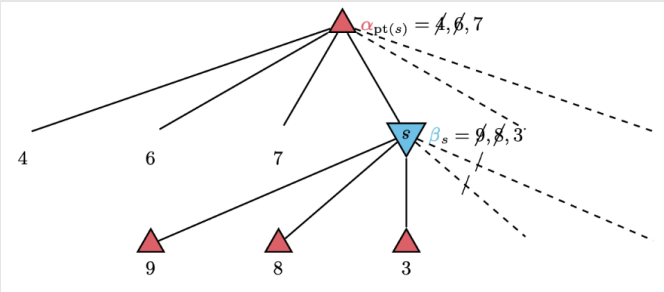


Figure 6

1.2.4 Alpha Beta Pruning

Process:

1. Initialize $\alpha = -\infty$ and $\beta = +\infty$
2. Iterate through the game tree:
   - If the maximizer is the turn-taker, then update α to the maximum value of s's direct children as they are explored.
   - If the minimizer is the turn-taker, then update β to the minimum value of s's direct children as they are explored.
3. Nodes are pruned if $\alpha \geq \beta$

Warning:

- Minimizer nodes: β gets changed and α never gets changed
- Maximizer nodes: α gets changed and β never gets changed
- Pattern: Go down the left branch of the tree until you reach a leaf node, then go back up to the parent node and go down the right branch of the tree until you reach a leaf node, ...
- Pruning: When you shortcut to just take the max value at the leaf (maximizer) or min value at the leaf (minimizer), make sure to prune the other branches that haven't been explored yet if $\alpha \geq \beta$

Example: Alpha-Beta Pruning Practice

Example:

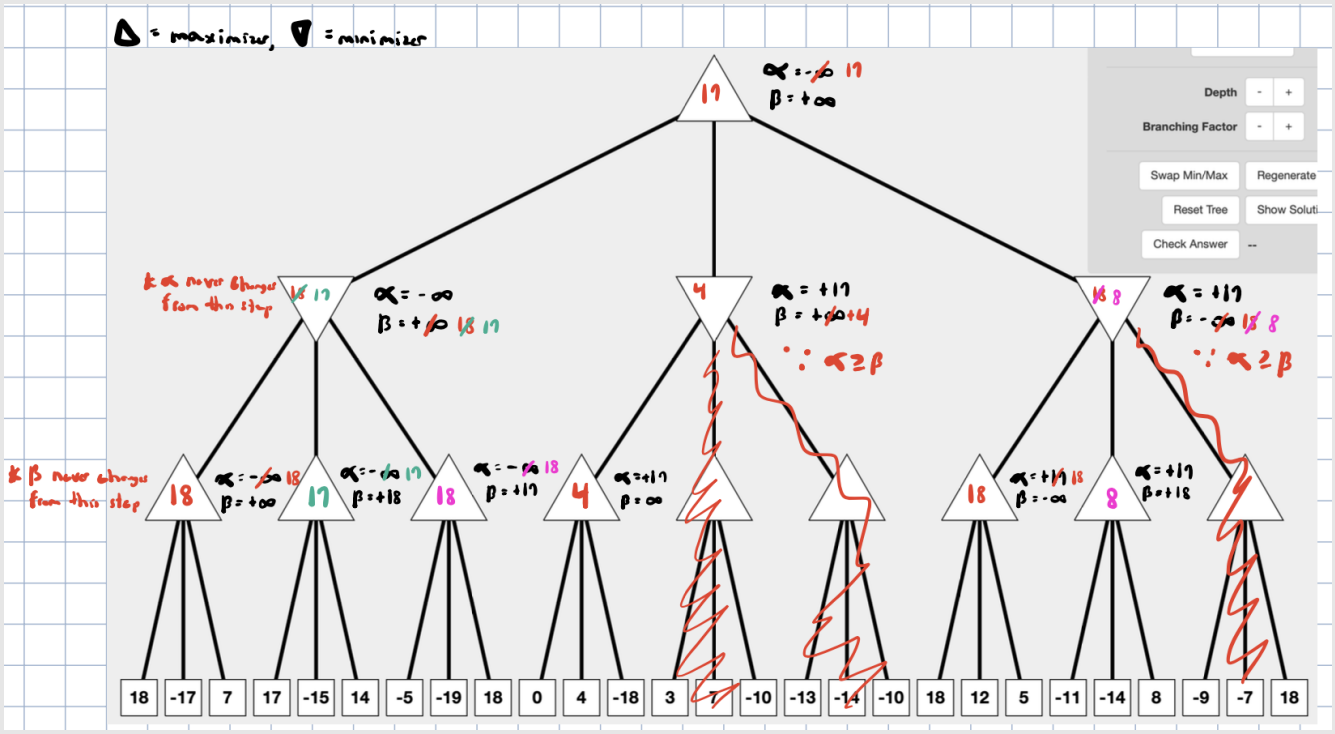
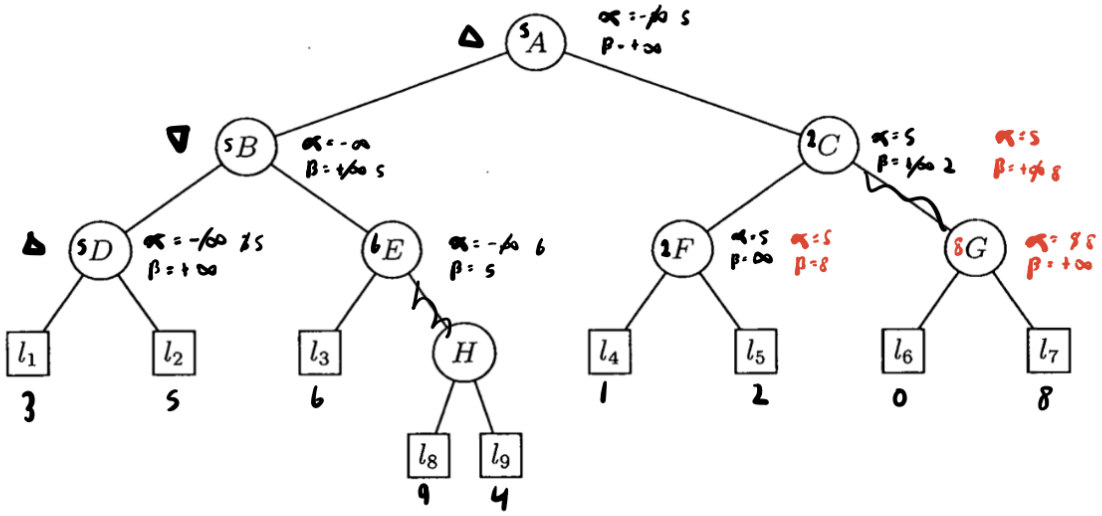


Figure 7

Example:

2. [3 pts] *α/β-Pruning on a Game-Tree.* Consider the game tree shown below:



Assume the rewards that Player 1 would receive are shown in the table below:

l_i	1	2	3	4	5	6	7	8	9
$r_1(l_i)$	3	5	6	1	2	0	8	9	4

(a) [2 pts] A depth-first search of the game-tree is used to evaluate the value of State A (for Player 1). Suppose α/β-pruning is used. Select all the nodes (including leaf nodes) that will be pruned during a min-max search (using a depth-first ordering) with α/β-pruning. If a node being pruned has children, the children are also pruned.

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☒ G ☒ H ☐ l_1 ☐ l_2 ☐ l_3 ☐ l_4 ☐ l_5
☒ l_6 ☒ l_7 ☒ l_8 ☒ l_9 ☐ None

(b) [1 pts] Repeat (a), but explore G before F. Does the number of nodes pruned increase, decrease, or remain unchanged?

◯ Increase ⊗ Decrease ◯ Remain unchanged

Figure 8