ROB311 Quiz 3

Hanhee Lee

March 29, 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,

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

  - 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 Monte-Carlo Tree Search (MCTS) Algorithm

Algorithm:

1. Selection: Traverse using an alternate policy until a node has unexplored children.

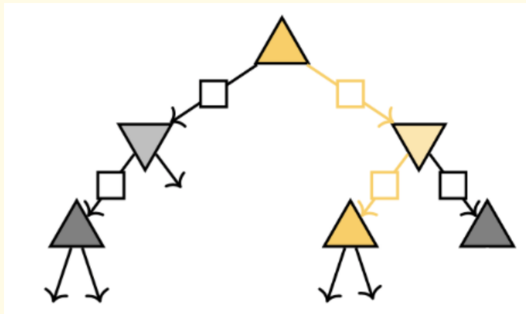

Figure 1

   - Our Agent (Upper Triangle): Uses UCB to choose the next node to explore
   - Other Agent (Down Triangle): Can't control their actions, so this agent picks w/ their own heuristic.
   - Square Boxes: Estimated values (i.e. n and $\hat{q}$)
   - Ends when there is at least one action that hasn't been explored yet. In this case, two actions ahven't been explored.
   - Can skip expansion and simulation if the most recently expanded node is a terminal state.
2. Expansion: Expand an unexplored child; initialize $n(a)$ and $\hat{q}(s, a)$.

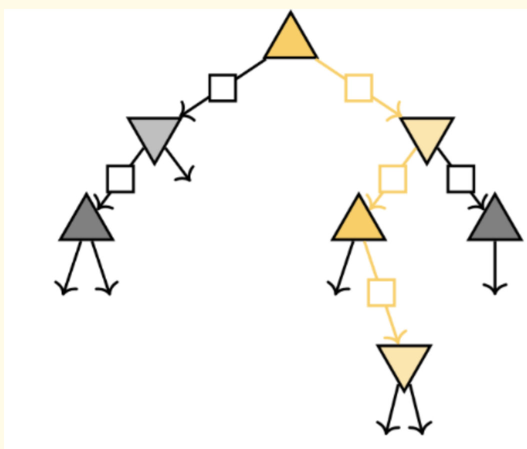

Figure 2

   - $\hat{q}(s, a)$ is initialized to 0 and $n(a)$ is initialized to 1 b/c we've visited this node once.
   - Randomly pick an unexplored action unless there is only one action left.
   - Can skip similuation if the most recently expanded node is a terminal state.
3. Simulation: Traverse using the random policy until a terminal node is reached.

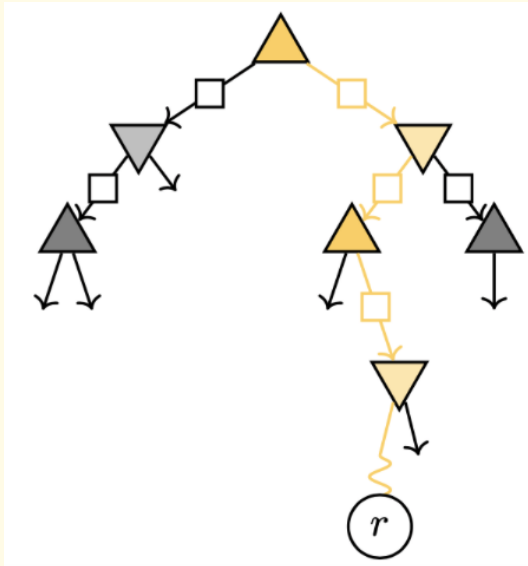

Figure 3

- Using random policy to simulate the game until a terminal state is reached (i.e. reward is obtained)

4. Back-propogation: Get the reward and reverse; update $n(a)$ and $\hat{q}(s, a)$.

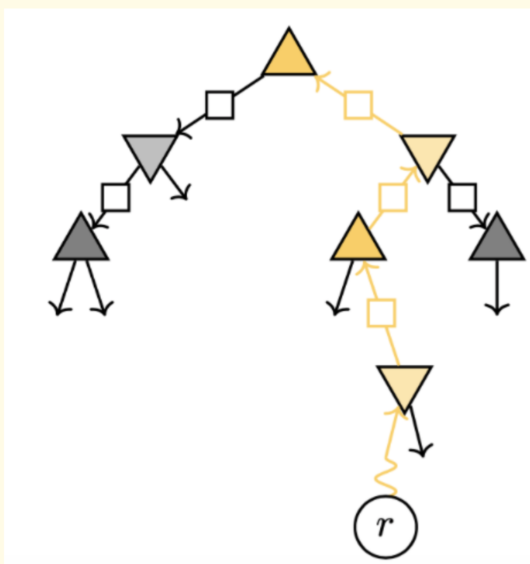

Figure 4

- Go up the path in yellow and update the values of $n(a)$ and $\hat{q}(s, a)$ for OUR agent only (i.e. the upper triangle)

Warning:
- Works for more than 2 agents.
- Don't need to know anyone else's reward function.
- Has to be turn taking but can be not alternating (i.e. immediate switch between agents)
- Can augment simultaneous actions
- Communication
- Works fo rnon-zero sum games.

1.3 Examples

1.3.1 Zero Sum Turn-Based Games

Example:

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

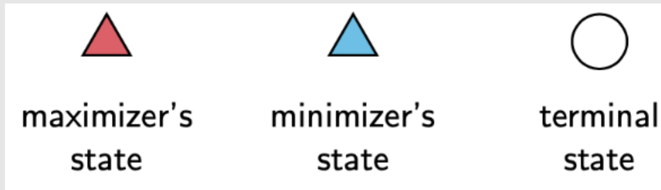

Figure 5: States

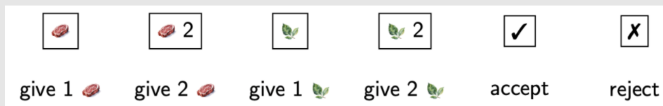

Figure 6: Actions

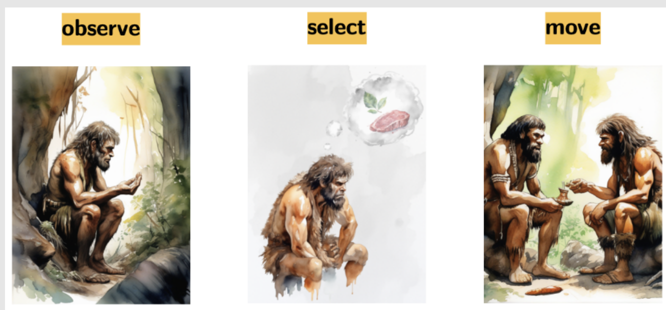

Figure 7: Decision Process

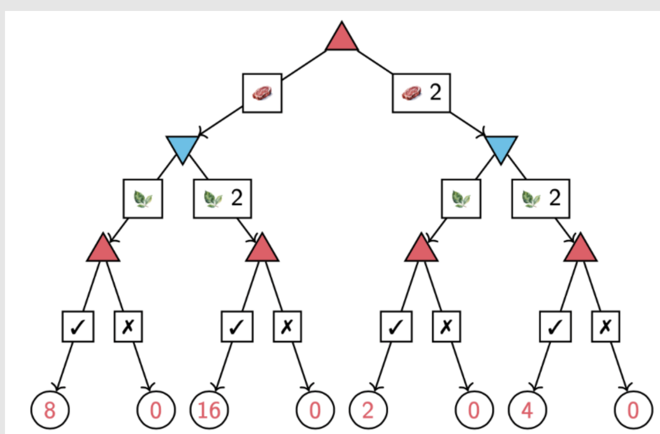

Figure 8: Game Tree

  - 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:** LL: Accept to get reward of 8, L: Accept to get reward of 16, R: Accept to get reward of 2, RR: Accept to get reward of 4
  - **Minimizer Turn:** LL: 1 medicine to make maximizer get reward of 8, R: 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.

1.3.2 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

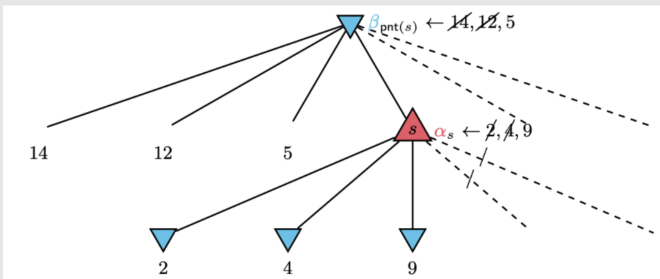

Figure 9

1.3.3 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

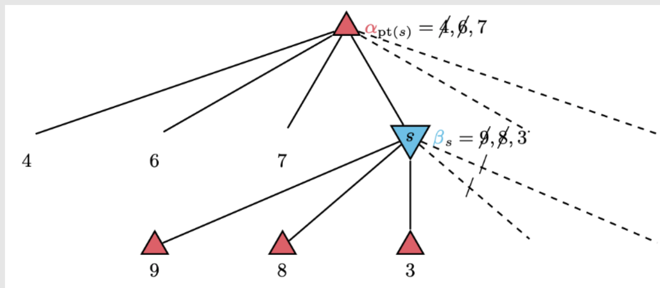

Figure 10

1.3.4 Alpha Beta Pruning

Process:

1.

Example: Alpha-Beta Pruning Practice

1.

1.3.5 Monte-Carlo Tree Search (MCTS) Algorithm

Example: