

# ROB311 Quiz 3

Hanhee Lee

March 20, 2025

## Contents

<b>1</b>	<b>Reinforcement Learning</b>	<b>2</b>
1.1	Estimating Q-Star Empirically . . . . .	2
1.1.1	Running Average Update Rule . . . . .	2
1.2	Q-Learning Algorithm . . . . .	3
1.3	Modified Q-Learning Algorithm . . . . .	3
1.4	Training vs. Testing . . . . .	4
1.4.1	$K$ Sims, 1 Test . . . . .	4
1.4.2	$K$ Tests . . . . .	4
<b>2</b>	<b>Partially Observable MDPs (POMDPs)</b>	<b>5</b>
2.1	Bayesian Network . . . . .	5
<b>3</b>	<b>Estimating the Optimal Quality Function</b>	<b>8</b>
3.1	Estimating the Optimal Quality Function . . . . .	8
3.2	Exploration versus Exploitation . . . . .	8
3.2.1	Simplified Case: . . . . .	8
3.3	Alternate Policies . . . . .	10
<b>4</b>	<b>Multi-Agent Problems</b>	<b>11</b>
4.1	Action Equilibria . . . . .	11
4.1.1	Finding Action Equilibria . . . . .	11
4.2	Strategy Equilibria . . . . .	11
4.2.1	Finding Strategy Equilibria . . . . .	11
4.2.2	Existence of Strategy Equilibria . . . . .	11
4.2.3	Convergence of Strategy Equilibria . . . . .	11
4.3	Examples . . . . .	11
4.3.1	Optimal Action Profiles . . . . .	11
<b>5</b>	<b>Turn-Based Games</b>	<b>12</b>
5.1	Zero-Sum Turn-Based Games . . . . .	12
5.2	$\alpha/\beta$ Pruning . . . . .	13
5.2.1	$\alpha$ Cuts . . . . .	13
5.2.2	$\beta$ Cuts . . . . .	13
5.3	Examples . . . . .	14
5.3.1	Zero Sum Turn-Based Games . . . . .	14
5.3.2	$\alpha$ Cuts . . . . .	15
5.3.3	$\beta$ Cuts . . . . .	15

# Partially Observable Probabilistic Decision Problems

## 1 Reinforcement Learning

**Summary:** In a RL problem,  $p(\cdot | \cdot, \cdot)$  and/or  $r(\cdot, \cdot)$  unknown.

### 1.1 Estimating Q-Star Empirically

**Summary:**

$\gamma$	Equation
0	$q^*(s, a) = \lim_{K \rightarrow \infty} \bar{R}_K$ <ul style="list-style-type: none"> <li><math>\bar{R}_K = \frac{1}{K} \sum_{k=1}^K r_k</math>: empirical average reward.</li> <li><math>r_k</math>: reward obtained in the <math>k^{\text{th}}</math> simulation.</li> <li><math>K</math>: # of times action <math>a</math> taken in state <math>s</math> (# of simulations)</li> </ul>
0	$q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} (r(s, a, s') - q^*(s, a))$ <ul style="list-style-type: none"> <li><math>N(s, a)</math>: # of times action <math>a</math> taken in state <math>s</math>.</li> </ul>
$\neq 0$	$q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left( \left[ r(s, a, s') + \gamma \max_{a'} q^*(s', a') \right] - q^*(s, a) \right)$ <ul style="list-style-type: none"> <li>Using old <math>q^*</math> values to estimate.</li> </ul>

#### 1.1.1 Running Average Update Rule

**Definition:**

$$\bar{x} \leftarrow \bar{x} + \alpha(x_{\text{new}} - \bar{x}).$$

- $\alpha$ : learning rate

## 1.2 Q-Learning Algorithm

### Algorithm:

```

1 procedure Q_LEARNING():
2   for each episode do
3     set initial state  $s \leftarrow s_0$ 
4     while  $s \notin \mathcal{T}$  do #  $\mathcal{T}$ : terminal states
5       randomly choose an action in  $\mathcal{A}(s)$ 
6       get next state,  $s'$ , and reward  $r$ 
7       update  $N(s, a)$  and  $q^*(s, a)$  as follows:
8
9        $q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left( r(s, a, s') + \gamma \max_{a'} q^*(s', a') - q^*(s, a) \right)$ 
10
11        $N(s, a) \leftarrow N(s, a) + 1$ 
12
13        $s \leftarrow s'$ 
14     end while
15   end for

```

- **Note:** Possible infinite while loop if  $\mathcal{T}$  is not reached.

## 1.3 Modified Q-Learning Algorithm

### Algorithm:

```

1 procedure Q_LEARNING():
2   for each episode do
3      $l \leftarrow 0$ 
4     set initial state  $s \leftarrow s_0$ 
5     while  $s \notin \mathcal{T}$  and  $l < l_{\max}$  do
6       randomly choose an action in  $\mathcal{A}(s)$ 
7       get next state,  $s'$ , and reward  $r$ 
8       update  $N(s, a)$  and  $q^*(s, a)$  as follows:
9
10       $q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left( r(s, a, s') + \gamma \max_{a'} q^*(s', a') - q^*(s, a) \right)$ 
11
12       $N(s, a) \leftarrow N(s, a) + 1$ 
13
14       $l \leftarrow l + 1$ 
15       $s \leftarrow s'$ 
16    end while
17  end for

```

**Notes:** Choice of  $\gamma$  and  $l_{\max}$  are coupled:

- $\gamma \approx 1$  requires large  $l_{\max}$
- $\gamma \approx 0$  requires small  $l_{\max}$

## 1.4 Training vs. Testing

**Notes:** Episodes are classified as either:

- training (sim): reward accumulated during episode does not count
- testing (test): reward accumulated during episode counts

### 1.4.1 $K$ Sims, 1 Test

**Notes:**

1. select actions randomly during  $K$  simulations
2. extract optimal policy,  $\pi^*$
3. use  $\pi^*$  during test

### 1.4.2 $K$ Tests

**Notes:**

- maximize average reward over  $K$  tests
- must balance between exploration and exploitation
- Common ways to balance exploration and exploitation:  $\varepsilon$ -greedy strategy, UCB algorithm

Strategy	Description
$\varepsilon$ -greedy	<p>choose optimal action with probability <math>\varepsilon(k)</math></p> <ul style="list-style-type: none"> <li>• In episode <math>k</math>, choose the optimal action with probability <math>\varepsilon(k)</math>, where: <ul style="list-style-type: none"> <li>– <math>\varepsilon(0) \approx 0</math></li> <li>– <math>\varepsilon(k)</math> is increasing</li> <li>– <math>\varepsilon(k) \rightarrow 1</math> as <math>k \rightarrow \infty</math></li> </ul> </li> <li>• Common choice for <math>\varepsilon(k)</math> is <math>1 - \frac{1}{k}</math>.</li> </ul>
UCB algorithm	<p>choose action that maximizes <math>\text{UCB}(\cdot)</math></p> $\text{UCB}(s, a) = \begin{cases} q^*(s, a) + C \sqrt{\frac{\log k}{N(s, a)}}, & \text{if } N(s, a) > 0 \\ \infty, & \text{otherwise} \end{cases}$ <ul style="list-style-type: none"> <li>• In episode <math>k</math>, choose the action that maximizes <math>\text{UCB}(\cdot)</math>.</li> <li>• <math>C</math>: exploration parameter</li> <li>• <math>N(s, a)</math>: # of times <math>a</math> taken from <math>s</math>.</li> </ul>

## 2 Partially Observable MDPs (POMDPs)

**Summary:** In a POMDPs, we assume that:

- environment modelled using state space,  $\mathcal{S}$
- single agent
- $S_t$  = state after transition  $t$
- $A_t$  = action inducing transition  $t$
- stochastic state transitions with memoryless property:

$$S_T \perp S_0, A_1, \dots, A_{T-1}, S_{T-2} \mid S_{T-1}, A_T$$

- $R_t$  = reward for transition  $t$ , i.e.,  $(S_{T-1}, A_T, S_T)$
- $O_t$  = observation of  $S_t$

Name	Function:
Initial state distribution	$p_0(s) := \mathbb{P}[S_0 = s]$
Transition distribution	$p(s' s, a) := \mathbb{P}[S_t = s'   A_t = a, S_{t-1} = s]$
Reward function	$r(s, a, s') := \text{reward for transition } (s, a, s')$
<ul style="list-style-type: none"> <li>• Since actual state is unknown, so are legal actions.</li> <li>• Can fix by assuming <math>\mathcal{A}(s) = \mathcal{A}(s') := \mathcal{A} \forall s, s'</math>: <ul style="list-style-type: none"> <li>– if <math>a \notin \mathcal{A}(s)</math>, then <math>p(s' s, a) = 0</math> for all <math>s' \neq s</math></li> <li>– if <math>a \notin \mathcal{A}(s)</math>, then <math>r(s, a, s') = 0</math> for all <math>s'</math></li> </ul> </li> </ul>	
Policy for choosing actions	$\pi_t(a o_0, \dots, o_t) := \mathbb{P}[A_t = a   O_0 = o_0, \dots, O_t = o_t]$
Measurement model	$m(o s) := \mathbb{P}[O_t = o   S_t = s]$
<ul style="list-style-type: none"> <li>• Observe that policy is now time-dependent.</li> <li>• <b>Special Case:</b> If we assume the agent cannot use past observations, <math>A_t \perp O_0, \dots, O_{t-1} \mid O_t</math>, policy becomes time-independent, <math display="block">\pi_t(a o_0, \dots, o_t) = \pi_0(a o_t).</math> <ul style="list-style-type: none"> <li>– Only need to specify <math>\pi_0</math>.</li> </ul> </li> </ul>	
Belief after $t$ observations	$b_t(s_t a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t   A_t = a_t, O_{0:t} = o_{0:t}]$ $b_t(s_t a_{1:t}, o_{0:t}) = m(o_t s_t) \sum_{s_{t-1}} p(s_t s_{t-1}, a_t) b_{t-1}(s_{t-1} a_{1:t-1}, o_{1:t-1})$
<ul style="list-style-type: none"> <li>• <math>b_t</math>: Probability distribution</li> <li>• <math>b_0(s_0) = \mathbb{P}[S_0 = s_0]</math>: Initial belief distribution</li> <li>• Only holds for <math>t \geq 1</math>.</li> <li>• For <math>t = 0</math> (assuming uniform prior): <math>b_0(s_0 o_0) = \frac{m(o_0 s_0)}{\sum_s m(o_0 s)}</math>.</li> </ul>	

### 2.1 Bayesian Network

**Notes:**  $S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2, \dots$  form a Bayesian network:

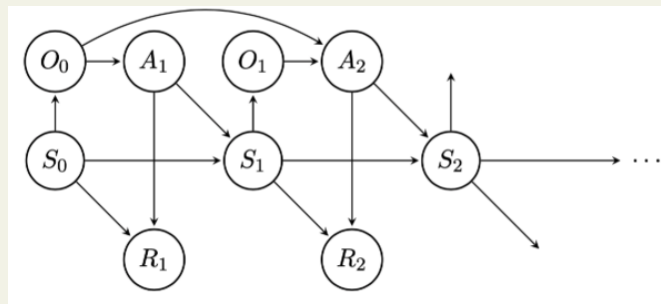


Figure 1

- Assuming  $A_t \perp O_0, \dots, O_{t-1} \mid O_t$ . WHERE DOES THIS COME INTO PLAY.

**Example:**

### 3 Estimating the Optimal Quality Function

#### 3.1 Estimating the Optimal Quality Function

**Motivation:** The agent need not know the model of the environment. However, it must actually make moves, even when learning.

If the agent doesn't have a model, it must estimate  $q^*$ ,  $\mathcal{A}^*$ , and  $\pi^*$ .

**Definition:** When the environment is in state  $s$ , the agent can take an action  $a$  and:

- **Update  $\hat{q}$ :**  $\hat{q}(s, a; t) \leftarrow (1 - \alpha)\hat{q}(s, a; t) + \alpha \left( r' + \gamma \max_{a'} \hat{q}(s', a'; t + 1) \right)$   
–  $0 \leq \alpha \leq 1$ : learning rate
- **Compute  $\hat{\mathcal{A}}$ :**  $\hat{\mathcal{A}}(s; t) = \arg \max_{a' \in \mathcal{A}(s)} \hat{q}(s, a'; t)$
- **Compute  $\hat{\pi}$ :**  $\hat{\pi}(a' | s; t) = 0 \ \forall a' \notin \hat{\mathcal{A}}(s; t)$

#### 3.2 Exploration versus Exploitation

**Motivation:** To ensure  $\hat{q}$  converges to  $q^*$  and the agent's expected return is maximized, the agent must balance exploration and exploitation.

**Definition:**

- **Exploitation:** Choose the most promising actions based on current knowledge.  
– Use optimal policy:  $\hat{\pi}(\cdot, \cdot; t)$
- **Exploration:** Choose the least tried actions to improve current knowledge.  
– Choose actions randomly

##### 3.2.1 Simplified Case:

**Example:**

- **Given:** Assume the environment is stateless, but rewards are random.



Figure 2

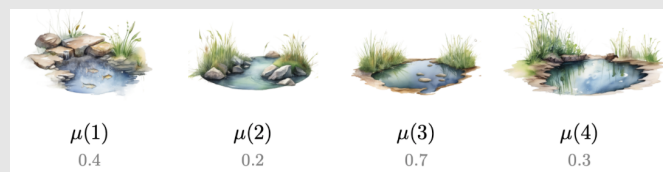


Figure 3

- $\mu(a)$ : expected reward for action  $a$  (unknown to the agent):  
–  $0 \leq \mu(a) \leq 1$  for all  $a$ .

- **Best-case expected return:** (with  $\gamma = 1$  under  $\pi^*$ ) from transition  $t$  is:



$$u^*(t) := (T - t) \max_{a'} \mu(a')$$

where in this case:

$$\pi^*(a; t) = 0 \quad \text{if } a \notin \arg \max_{a'} \mu(a').$$

- **Estimation of  $\mu(\cdot)$ .** Since the agent does not have a model, it must estimate  $\mu(\cdot)$ .

The agent can take an action  $a$  and:

1. **Update**  $n(\cdot)$  and  $\hat{\mu}(\cdot)$ :

$$n(a) \leftarrow n(a) + 1$$

$$\hat{\mu}(a) \leftarrow \left(1 - \frac{1}{n(a)}\right) \hat{\mu}(a) + \frac{1}{n(a)} r'$$

2. **Compute  $\hat{\pi}$ :**

$$\hat{\pi}(a; t) = 0 \quad \text{for all } a \notin \arg \max_{a'} \hat{\mu}(a').$$

- **Alternate Policies** We want to compare the expected return under various policies. The expected return from transition  $t$  under a policy  $\rho$  is:

$$u^\rho(t) := \mathbb{E}^\pi[G_t] = \sum_{a'} \rho(a'; t) (\mu(a') + u^\rho(t+1)).$$

### 3.3 Alternate Policies

**Summary:** To ensure the agent's expected return is maximized, the agent must strike a balance exploration and exploitation.

In the following cases, the expected return from transition  $t$  is

$$u^{\text{avg}}(t) \equiv \frac{T-t}{|\mathcal{A}|} \sum_a \mu(a)$$

We want to choose  $\rho$  so that  $u^\rho > u^{\text{avg}}$ .

Policy	Function:
Exploitation only	Choose a random action, same for all transitions
Exploration only	Choose a random action, different for each transition
Softmax	Apply a soft-max over $\hat{u}$ $\rho(a; t) = \left[ \sum_{a'} \exp \left( \frac{\hat{\mu}(a')}{\tau} \right) \right]^{-1} \exp \left( \frac{\hat{\mu}(a)}{\tau} \right)$ <ul style="list-style-type: none"> <li>Choose a temperature value decrease with <math>t</math>.</li> <li><math>\tau(t) \in [0, \infty), \tau \rightarrow 0</math></li> </ul>
$\epsilon$ -greedy	Use $\hat{\pi}$ w/ prob. $1 - \epsilon$ , otherwise take a random action $\rho(a; t) = \epsilon \frac{1}{ \mathcal{A} } + (1 - \epsilon) \hat{\pi}(a; t)$ <ul style="list-style-type: none"> <li>Choose an exploration rate decrease w/ <math>t</math>.</li> <li><math>\epsilon(t) \in [0, 1], \epsilon \rightarrow 0</math></li> </ul>
Upper confidence bound	Choose the action with the highest $\text{ucb}(\cdot)$ $\rho(a; t) = 0 \text{ if } a \notin \arg \max_{a'} \text{ucb}(a'; t)$ <ul style="list-style-type: none"> <li>Compute <math>\text{ucb}(\cdot)</math> for each action.</li> <li><math display="block">\text{ucb}(a; t) = \hat{\mu}(a) + \sqrt{\frac{\ln t}{n(a)}}</math></li> </ul>

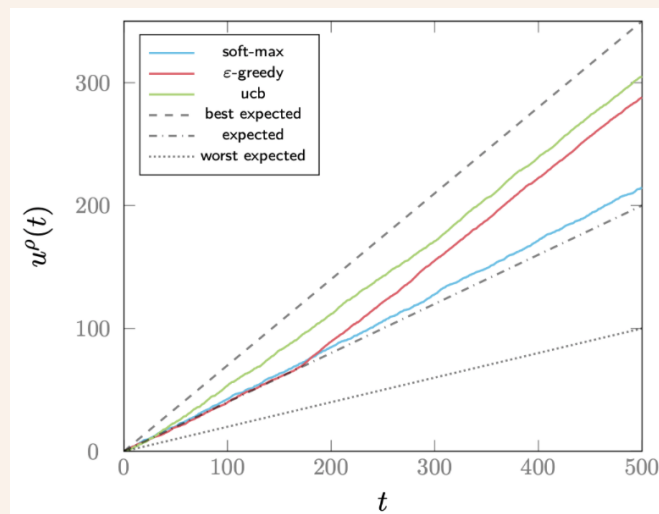


Figure 4

# One-Shot Multi-Agent Decision Problems

## 4 Multi-Agent Problems

**Summary:** In a **Multi-Agent problem**, we assume that:

- Set of states for environment is  $\mathcal{S}$
- $P$  agents within environment.
- For each state  $s \in \mathcal{S}$ :
  - possible actions for agent  $i$  is  $\mathcal{A}_i(s)$
  - set of action profiles is  $\mathcal{A}(s) = \prod_{i=1}^P \mathcal{A}_i(s)$
- possible state-action pairs are  $\mathcal{T} = \{(s, a) \text{ s.t. } s \in \mathcal{S}, a \in \mathcal{A}(s)\}$
- environment in some origin state,  $s_0$
- environment destroyed after  $N$  transitions
- agent  $j$  wants to find policy  $\pi_j(a_j | s)$  so that  $\mathbb{E}[r_j(p)]$  is maximized
- agents act independently given the environmen

Name	Function:
State transition given state-action pair defined by $\text{tr} : \mathcal{T} \rightarrow \mathcal{S}$	$\text{tr}(s, a) = \text{state transition from } s \text{ under } a$
Reward to each agent, $i$ defined by $r_i : \mathcal{Q} \times \mathcal{S} \rightarrow \mathbb{R}_+$	$r_i(s, a, \text{tr}(s, a)) = \text{rwd to agent } i \text{ for } (s, a, \text{tr}(s, a))$
State evolution of environment after $N$ transitions	$p = \langle (s_0, a^{(1)}, s_1), \dots, (s_{N-1}, a^{(N)}, s_N) \rangle$ <ul style="list-style-type: none"> <li>• Given sequence of actions: <math>p.a = \langle a^{(1)}, \dots, a^{(n)} \rangle</math></li> <li>• <math>s_N = \tau(s_{n-1}, a^{(n)})</math></li> </ul>
reward to agent $i$	$r_i(p) = \sum_{n=1}^N r_i(s_{n-1}, a^{(n)}, s_n)$
expected-reward (value) of playing $a$ from $s$ for agent $j$	$p(s' s) := \mathbb{P}[S_{t+1} = s'   S_t = s]$
Prob. that state of the env. after $T$ transitions is $s$	$p_T(s) := \mathbb{P}[S_T = s]$ $= \sum_{s'} p_{T-1}(s') p(s s')$ <ul style="list-style-type: none"> <li>• <math>p_{T-1}(s')</math>: Prob. <math>s'</math> at <math>T-1</math> (given)               <ul style="list-style-type: none"> <li>– <math>p_0(s)</math>: Base case</li> </ul> </li> <li>• <math>p(s s')</math>: Prob. <math>s</math> given <math>s'</math> (from graph)</li> </ul>

### 4.1 Action Equilibria

#### 4.1.1 Finding Action Equilibria

### 4.2 Strategy Equilibria

#### 4.2.1 Finding Strategy Equilibria

#### 4.2.2 Existence of Strategy Equilibria

#### 4.2.3 Convergence of Strategy Equilibria

### 4.3 Examples

#### 4.3.1 Optimal Action Profiles

## 5 Turn-Based Games

### 5.1 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,  $r(\cdot, \cdot, \cdot)$ :

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

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

## 5.2 $\alpha/\beta$ Pruning

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

**Definition:** For each state  $s$ :

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

### 5.2.1 $\alpha$ Cuts

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

- If  $\alpha_s$  increases beyond  $\beta_s$ , then  $s$  unreachable under perfect play.

### 5.2.2 $\beta$ Cuts

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

- If  $\beta_s$  decreases beyond  $\alpha_s$ , then  $s$  unreachable under perfect play.

## 5.3 Examples

### 5.3.1 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.

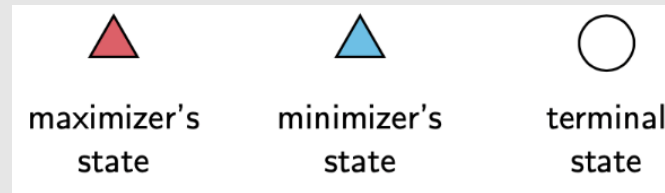


Figure 5: States

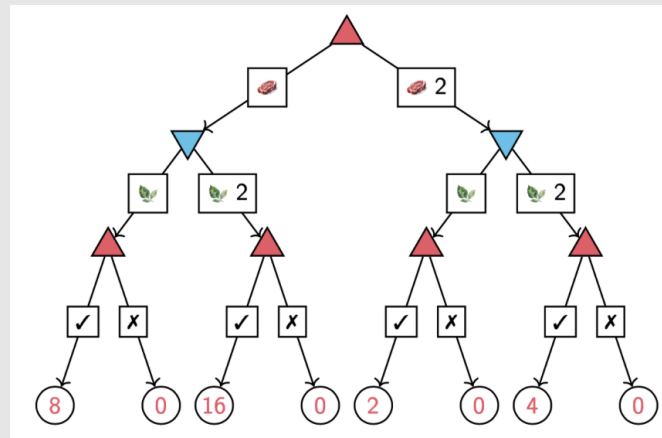


Figure 6: Game Tree

- States
  - \* Red triangle: Maximizing agent
  - \* Blue triangle: Minimizing agent
  - \* White circles with #s: terminal states
- Actions: Square boxes are actions



Figure 7: Actions

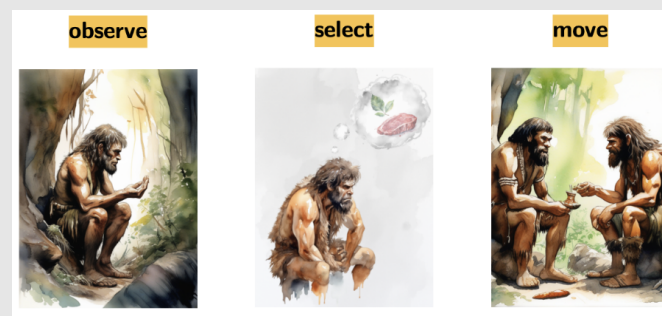


Figure 8: Decision Process

### 5.3.2 $\alpha$ Cuts

**Example:**

- Explored 14, 12 and now  $\beta_{\text{parent}(s)} = \beta_s = 5$ , so this will be compared for  $\alpha_s$  until  $\alpha_s > \beta_s$ .
- 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

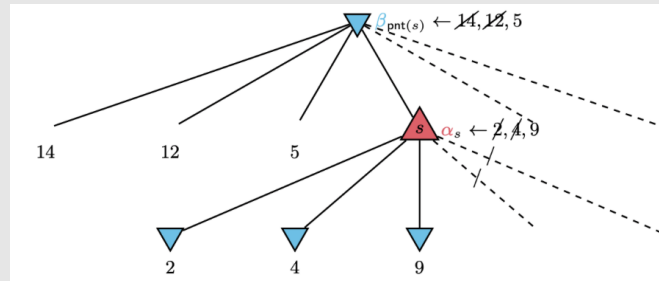


Figure 9

### 5.3.3 $\beta$ Cuts

**Example:**

- Explored 4, 6, and now  $\alpha_{\text{parent}(s)} = \alpha_s = 7$ , so this will be compared for  $\beta_s$  until  $\beta_s < \alpha_s$ .
- 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

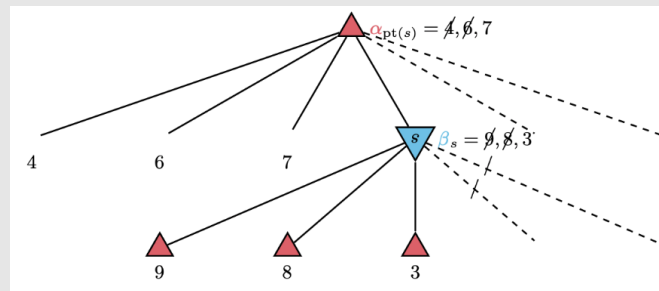


Figure 10