Chapter 6
Search: Adversarial Search

Introduction to Artificial Intelligence

Chandra Gummaluru
University of Toronto

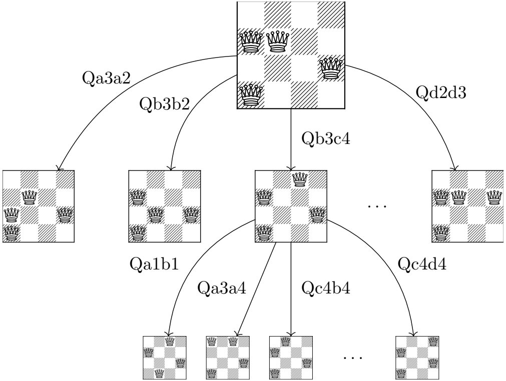
Version W22.1

- The following is based on material developed by many individuals, including (but not limited to):
  - Sheila McIlraith
  - Bahar Aameri
  - Fahiem Bacchus
  - Sonya Allin

- Recall the formal definition of a search problem.
- **Definition**: Search Problem
  - Let $\mathcal{S}$ be a set of **states** that we want to search through.
  - From any given state, $s \in \mathcal{S}$, there exist a set of **actions**, $A(s)$.
  - When an action, $a \in A(s)$, is applied to s, the result is a new state, denoted $a(s)$.
  - A sequence of actions, $\langle a_1, \ldots, a_n \rangle$ defines a **path** between two states.
  - The **length** of the path is the number of actions that make it up.
  - Each action, a, may have an associated **cost**, $c(a) > 0$. In this case, the cost of the path is the cumulative cost of its actions.
  - Given some initial state, s_0, we seek a path (often the shortest/cheapest one) to some state in a subset, $\mathcal{G} \subseteq \mathcal{S}$, called the **goal space**.

- We saw that a graph could be used to represent the structure of a search problem. However, the searching itself takes place on a tree of possible paths.

  Example: Search Tree for 4-Queens Puzzle



Formalizing a Search Problem: Assumptions

- So far, we have assumed our agent has complete control over the states.
- In particular, we assumed:
  1. Our agent is the only actor.
  2. The actions have deterministic consequences.
- However, these are very strong assumptions.
- We need to modify our algorithms to work even when the aforementioned assumptions do not hold.

- If the first assumption is violated, our agent is said to be playing a game.

- A **game** is a situation in which:

  - There are multiple agents capable of making changes to the environment.
  - Each agent has their own goals and tries to alter the environment to its own benefit.

- Using search to make decisions in a game is difficult because the optimal action from any given state depends on the future actions of other agents.

- In general, a game can be:

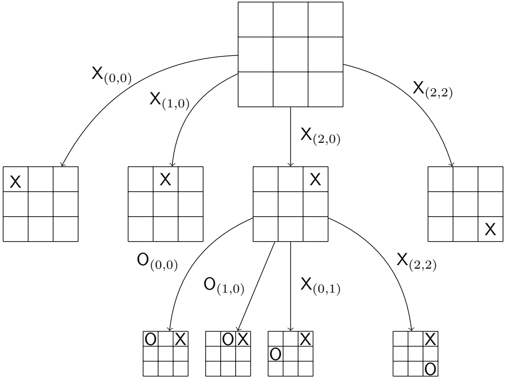
1. for any number of players, say N.
2. **deterministic** (i.e., do not involve randomness), or **stochastic** (i.e., not deterministic).
3. **finite** (i.e., the state-space and action sets are finite), or **infinite** (i.e., not finite).
4. **discrete** (i.e., state-space is discrete), or **continuous** (i.e., not discrete).
5. of **perfect information** (i.e., all states are either fully observable by all players), or **partial information** (i.e., not of perfect information).
6. **zero-sum** (i.e., the cumulative utility over all players is independent of the state), or **non-zero-sum** (i.e., not zero-sum).

- For simplicity, we will focus on 2-player, deterministic, finite, discrete, zero-sum games of perfect information.

- **Definition**: Components of a Game
  - Let M denote our agent and m denote the adversary.
  - Let $\mathcal{S}$ be the set of **states** that the game could be in.
  - A **circumstance**, k, is a state-player pair, i.e., game's state, and the turn-taker.
  - For any given circumstance, k, there is a set of **actions**, $A(k)$, that must be made by the turn-taker.
  - When an action, $a \in A(k)$, is applied to k, the result is a new circumstance, $a(k)$.
  - We assume that the turn-taking player switches after each action.
  - Let $\mathcal{T} \subseteq \mathcal{S}$ denote the set of **terminal states**, i.e., those in which the game ends.
  - Each terminal state, $t \in \mathcal{T}$, provides a utility, $u(t)$ to M and $-u(t)$ to m.
  - Given a circumstance, k, in which M is the turn-taking player, we seek an action, $a \in A(k)$, that (eventually) leads to a terminal state, $t \in \mathcal{T}$, which maximizes $u(t)$.

- Like any search problem, we can create a tree of possible paths. However, in a game tree, each level corresponds to different player, namely, the turn-taker.

Example: Game Tree for Tic-Tac-Toe Puzzle



- Given any circumstance, $k = (s, M)$, in which M is the turn-taker, they should chose the action, $a_M(s)$, that eventually leads to a terminal state with the maximum utility.

- If s is "almost-terminal", i.e., one action away from being terminal, the choice is straight-forward:

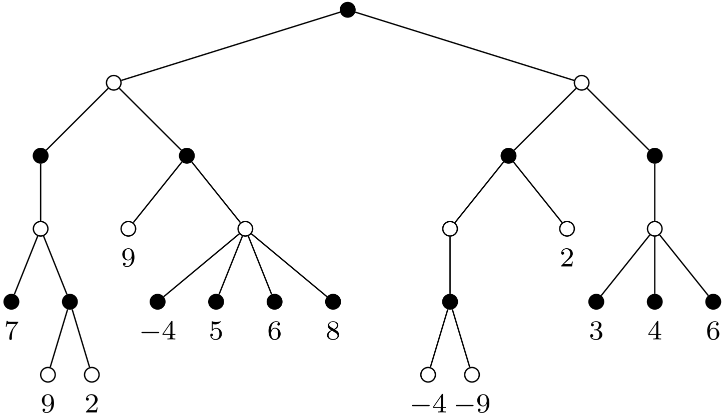
$$a_M(s) = \mathrm{argmax}_{a \in A(s)} \{u(a(s))\}$$

- If s is not "almost-terminal", $u(a(s))$, depends on m's actions.

- If M knew that m will play $a_m(s)$, in any circumstance, $k = (s, m)$, in which it is the turn-taker, then:

$$U(s,t) = \begin{cases} u(s), s \in \mathcal{T} \\ U(a_m(s), M), t = m \\ \max\left\{U(s', m), s' \in S(s)\right\}, t = M \end{cases}$$

- We will analyse different strategies on the game tree shown below, where • and ∘ respectively, denote circumstances in which the turn-taker is M and m:



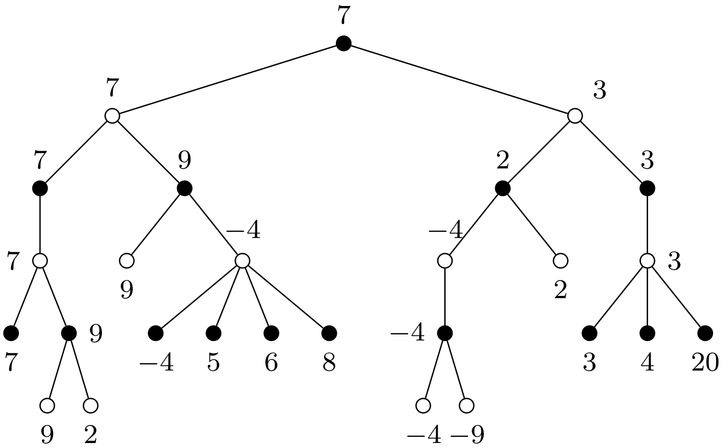
- In the **min-max** strategy, we assume m always plays its best-response, i.e.,

$$a_m(s) = \text{argmin}_{a \in A(s)} \{U(a(s), m)\}$$

- It follows that

$$U(s,t) = \begin{cases} u(s), s \in \mathcal{T} \\ \min\{U(s', M), s' \in S(s)\}\ t = m \\ \max\{U(s', m), s' \in S(s)\}, t = M \end{cases}$$

- Using the min-max strategy, we compute U as follows:

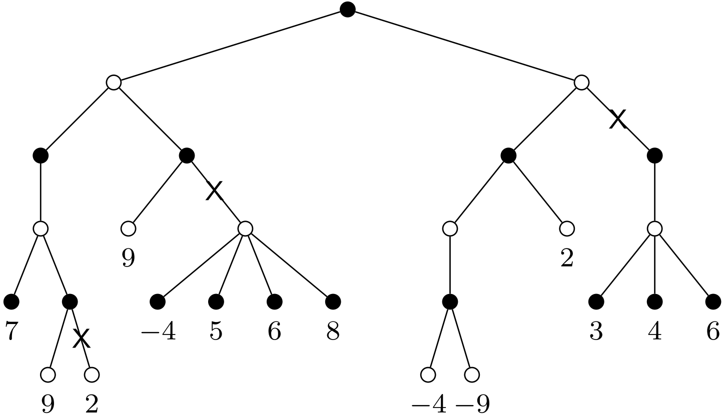


- Let s be some state and define two quantities:
  - α_s: the minimum utility guaranteed at s thus far.
  - β_s: the maximum utility guaranteed to s thus far.

- If M is the turn-taker at s:
  - α_s increases to the maximum value of s's successors as they explored.
  - β_s will remain fixed.
- If m is the turn-taker at s:
  - α_s will remain fixed.
  - β_s decreases to the minimum value of s's successors are explored.

- After exploring a successor of s, if α_s increases beyond β_s, then we need explore any other successors of s since m will ensure s is never reached.

- Using the min-max strategy with α/β-pruning, we prune the nodes shown below:



- The min-max strategy maximizes M's minimum utility.
- This means that M will never do worse than what is guaranteed by the min-max strategy, even if m does not always play its best response.
- However, it is very much possible for M to have done better.
- In the **expectation-maximization** strategy, we assume m chooses its action based on a probability distribution, P_s, over $A(s)$,.
- It follows that $a_m(S)$, and consequently $U(s,t)$), are random variables.
- We also assume that M wishes to maximize $\mathbb{E}(U(s_0, M))$, where

$$\mathbb{E}\big[U(s,t)\big] = \begin{cases} u(s), s \in \mathcal{T} \\ \mathbb{E}\big[U(a_m(s), M)\big], t = m \\ \max\left\{\mathbb{E}\big[U(s', m)\big], s' \in S(s)\right\}, t = M \end{cases}$$

and $\mathbb{E}\big[U(a_m(s), M)\big] = \sum_{a \in A(s)} U(a, M) p_s(a)$.

- Given p_S as shown below, it seems to make more sense for M to play the right action since there is an 85.5% chance that the resulting utility will be 20.
- However, if M uses the min-max strategy, it would play the action on the left, resulting in an expected utility of ≈ 7.04.