Version 2022.1

Lecture 1: Formalizing Intelligence

Introduction to Artificial Intelligence

Chandra Gummaluru *University of Toronto*

Introduction

In this course, our goal is to develop agents that exhibit "intelligent" behaviour (through computational means). To do this, we must first define "intelligence". In this lecture, we formalize the notion of intelligence.

Formalizing Intelligence

One possible definition of intelligence is as follows:

the ability to optimally deal with new situations.

This is a good starting point, but it is too imprecise to build computational algorithms with. We seek a formal definition that still captures the core idea of the above. For us, a "situation" will refer to a game in the game-theoretic sense.

Definition 0.1 (*Game*)

A game, $\mathcal{G}$, consists of the following components:

1. a set of states $\mathcal{S}$ in which there is:
   - a unique state $s_0 \in \mathcal{S}$, in which the game begins, called the initial state
   - a subset of states $\mathcal{T} \subseteq \mathcal{S}$, in which the game ends, called terminal states
2. an set-valued function, $\mathcal{A}_i$, for each player i, so that $\mathcal{A}_i(s)$ is the set of actions available to player i from state s, with $\mathcal{A}(s) := \bigtimes_{i=1}^{N} \mathcal{A}_i(s)$, and $\mathcal{A} := \bigtimes_{\forall s \in \mathcal{S}} \mathcal{A}(s)$.
3. a state transition distribution, $p : \mathcal{S}^2 \times \mathcal{A}^P \to [0, 1]$, so that $p(s, a, s')$ is the probability the game state transitions from s to s', when $a_i \in \mathcal{A}_i(s)$ is the action chosen by player i
4. a reward function $R_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, for each player, i, so that $R_i(s, a, s')$ is the reward player i incurs if the game transitions from state s to s' under the action a.

Graph-Theory Review

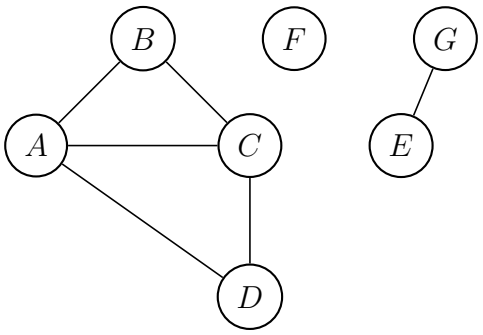
As we will see shortly, we can represent a game using a graphical structure called a tree. A tree can be best understood as a type of graph. A **graph** is used to model relationships between elements of a set, $\mathcal{V}$, called its **vertices** by connecting them through a set of **edges**, $\mathcal{E}$. An edge between two vertices, $u, v \in \mathcal{V}$ can be:

- *directed*, in which case, each edge is represented as a ordered-pair, (u, v)
- *undirected*, in which case, each edge is represented as a pair, $\{u, v\}$.

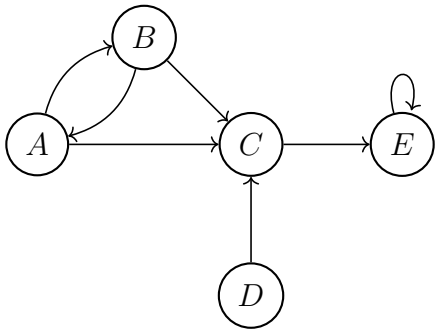
We say the graph is **directed** if it contains at least one directed edge.

An ordered sequence of edges, $\langle e_1, \ldots, e_n \rangle$, where $e_i = (v_i, v_{i+1})$ defines a **path** between v_1 and v_{n+1}; sometimes we represent the path with the corresponding vertex sequence, $\langle v_1, \ldots, v_{n+1} \rangle$. If $e_1 = (v_1, \ldots, v_2)$, and $e_n = (v_{n-1}, v_n)$, where $v_n = v_1$, the path is a **cycle**. A path is **cyclic** if any sub-sequence of edges along it form a cycle. A graph is cyclic if it contains a cyclic path.

Two vertices are **related** if there exists at least one path between them. If the path is directed and v follows u in a path, then u is an **ancestor** of v and v is a **successor** of u. If v immediately follows u in a path, then u is a **parent** of v and v is a child of u. If all vertices are related to each other, we say that the graph is **connected**.



(a) $\mathcal{E} = \{\{A, B\}, \{A, C\}, \{B, C\}, \{A, D\}, \{C, D\}, \{E, G\}\}$.



(b) $\mathcal{E} = \{(A, B), (B, A), (A, C), (B, C), (D, C), (C, E), (E, E)\}$.

Figure 1: An undirected graph (a), and a directed graph (b), in which $\mathcal{V} = \{A, B, C, D, E, F, G\}$.

A **tree** can be thought of as an acyclic, directed, connected graph. We can represent the game using a tree $(\mathcal{S}, \mathcal{E})$, in which the vertices are states, and an edge of the form (s, s'), represents one possible joint action $a \in \mathcal{A}(s)$, from s that results in the state s'.

Computational Game Problems

Let $\mathcal{G} = (P, \mathcal{S}, \mathcal{A}, R)$ be a game and $s_0 \in \mathcal{S}$ be some initial state. Each way that $\mathcal{G}$ can play out, i.e., each sequence of actions, $p = \langle a^{(1)}, \ldots, a^{(n)} \rangle$, where $a^{(j)} \in A(s_j)$, $s_j = a^{(j)}(s_{j-1})$ and $s_n \in T$, is called a **path** from s_0. The associated reward and cost for i is given by

$$R_i(p) := \sum_{j=1}^{n} R_i\left(s_{j-1}, a^{(j)}, s^{(j)}\right).$$

We think of ourselves as one of the players, say $i = 1$. Given some $s_0 \in \mathcal{S}$, we want to find the best path to some $s \in \mathcal{T}$, where by "best", we mean the path that maximizes our own utility, R_1. This is difficult to do in general because we do not complete control over the other players' actions, and consequently, the path that the game actually follows. To proceed, we will start by making several simplifying assumptions, and relax some of them over time.