Algorithmic Game Theory and Applications

Lecture 11: Games of Perfect Information

Kousha Etessami

finite games of perfect information

Recall, a perfect information (PI) game has only 1 node per information set.

Theorem([Kuhn'53]) Every finite n-person extensive PI-game, $\mathcal{G}$, has a NE, in fact, a subgame-perfect NE (SPNE), in pure strategies.
I.e., some pure profile, $s^* = (s_1^*, \ldots, s_n^*)$, is a SPNE.

Before proving this, we need some definitions.
For a game $\mathcal{G}$ with game tree T, and for $w \in T$, define the **subtree** $T_w \subseteq T$, by:

$$T_w = \{w' \in T \mid w' = ww'' \text{ for } w'' \in \Sigma^*\}$$

Since the game tree is finite, we can just associate payoffs to the leaves.

Thus, the subtee T_w, in an obvious way, defines a "**subgame**", $\mathcal{G}_w$, which is also a PI-game.

Let the **depth** of a node w in T be its length $|w|$ (as a string). Let the depth of a tree T be the maximum depth of any node in T. Let the depth of a game $\mathcal{G}$ be the depth of its game tree.

proof of Kuhn's theorem (easy)

Proof We prove by induction on the depth of a subgame $\mathcal{G}_w$ that it has a pure strategy SPNE, $s^w = (s_1^w, \ldots, s_n^w)$. Then $s^* := s^\epsilon$.

Base case, depth 0: In this case we are at a leaf w. there is nothing to show: each player i gets payoff $u_i(w)$, and the strategies in the SPNE s^* are "empty" (it doesn't matter which player's node w is, since there are no actions to take.)

Inductive step: Suppose depth of $\mathcal{G}_w$ is $k+1$. Let $Act(w) = \{a'_1, \ldots, a'_r\}$ be the set of actions available at the root of $\mathcal{G}_w$. The subtrees $T_{wa'_j}$, for $j = 1, \ldots, r$, each define a PI-subgame $\mathcal{G}_{wa'_j}$, of depth $\leq k$.

Thus, by induction, each game $\mathcal{G}_{wa'_j}$ has a pure strategy SPNE, $s^{wa'_j} = (s_1^{wa'_j}, \ldots, s_n^{wa'_j})$.

To define $s^w = (s_1^w, \ldots, s_n^w)$, there are two cases to consider .....

two cases

1. $pl(w) = 0$, i.e., the root node, w, of T_w is a chance node (belongs to "nature").
   Let the strategy s_i^w for player i be just the obvious "union" $\bigcup_{a' \in \mathit{Act}(w)} s_i^{wa'}$, of its pure strategies in each of the subgames. (Explanation of "union" of disjoint strategy functions.)
   Claim: $s^w = (s_1^w, \ldots, s_n^w)$ is a pure SPNE of $\mathcal{G}_w$.
   Suppose not. Then some player i could improve its expected payoff by switching to a different pure strategy in one of the subgames. But that violates the inductive hypothesis on that subgame.
2. $pl(w) = i > 0$. In this case the root node, w, of T_w belongs to player i. For $a \in \mathit{Act}(w)$, let $h_i^{wa}(s^{wa})$ be the expected payoff to player i in the subgame $\mathcal{G}_{wa}$.
   For some a', $h_i^{wa'}(s^{wa'}) = \max_{a \in \mathit{Act}(w)} h_i^{wa}(s^{wa})$.
   For players $i' \neq i$, define $s_{i'}^w = \bigcup_{a \in \mathit{Act}(w)} s_{i'}^{wa}$.
   For i, define $s_i^w = (\bigcup_{a \in \mathit{Act}(w)} s_i^{wa}) \cup \{w \mapsto a'\}$.

   Claim: $s^w = (s_1^w, \ldots, s_n^w)$ is a pure SPNE of $\mathcal{G}_w$.
   The proof is basically the same argument as the previous case. Even for the root player, i, to (unilaterally) improve its payoff, it would need to do so in some (strict) subgame. ∎

easy "bottom-up" algorithm for SPNE's in finite PI-games

The proof yields an EASY "bottom up" algorithm for computing a pure SPNE in a finite PI-game:
We inductively "attach" to the root of every subtree T_w, a SPNE s^w for the game $\mathcal{G}_w$, together with the expected payoff vector $h^w := (h_1^w(s^w), \ldots, h_n^w(s^w))$.

1. Initially: Attach to each leaf w the empty profile $s^w = (\emptyset, \ldots, \emptyset)$, & payoff vector $h^w := (u_1(w), \ldots, u_n(w))$.
2. **While** (there is an unattached node w all of whose children are attached)
   - if $(pl(w) = 0)$ then
     $s^w := (s_1^w, \ldots, s_n^w)$,
     where $s_i^w := \bigcup_{a \in Act(w)} s_i^{wa}$;
     note, h^w is given by:
     $h_i^w(s^w) := \sum_{a \in Act(w)} q_w(a) * h_i^{wa}(s^{wa})$;
     else if $(pl(w) = i > 0)$ then
     Let $s^w := (s_1^w, \ldots, s_n^w)$, & $h^w := h^{wa'}$, where
     $s_{i'}^w := \bigcup_{a \in Act(w)} s_{i'}^{wa}$, for $i' \neq i$,
     $s_i^w := (\bigcup_{a \in Act(w)} s_i^{wa}) \bigcup \{w \mapsto a'\}$,
     and where $a' \in Act(w)$ such that
     $h_i^{wa'}(s^{wa'}) = \max_{a \in Act(w)} h_i^{wa}(s^{wa})$

consequences for zero-sum finite PI-games

Recall that, by the Minimax Theorem, for every finite zero-sum game Γ, there is a value v^* such that for any NE (x_1^*, x_2^*) of Γ, $v^* = U(x_1^*, x_2^*)$, and

$$\max_{x_1 \in X_1} \min_{x_2 \in X_2} U(x_1, x_2) = v^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} U(x_1, x_2)$$

But it follows from Kuhn's theorem that for extensive PI-games $\mathcal{G}$ there is in fact a pure NE (in fact, SPNE) (s_1^*, s_2^*) such that $v^* = u(s_1^*, s_2^*) := h(s_1^*, s_2^*)$, and thus that in fact

$$\max_{s_1 \in S_1} \min_{s_2 \in S_2} u(s_1, s_2) = v^* = \min_{s_2 \in S_2} \max_{s_1 \in S_1} u(s_1, s_2)$$

Definition We say a finite zero-sum game Γ is **determined** or **has a value**, precisely if

$$\max_{s_1 \in S_1} \min_{s_2 \in S_2} u(s_1, s_2) = \min_{s_2 \in S_2} \max_{s_1 \in S_1} u(s_1, s_2)$$

It thus follows from Kuhn's theorem that:

Proposition ([Zermelo'1912]) Every finite zero-sum PI-game, $\mathcal{G}$, is determined (has a value).
Moreover, the value & a pure minimax profile can be computed "efficiently" from $\mathcal{G}$.

chess

Chess is a finite PI-game (after 50 moves with no piece taken, it ends in a draw). In fact, it's a **win-lose-draw** PI-game: there are no chance nodes and the only possible payoffs $u(s_1, s_2)$ are 1, -1, and 0.

Proposition([Zermelo'1912]) In Chess, either

1. White has a "winning strategy", or
2. Black has a "winning strategy", or
3. Both players have strategies to force a draw.

A "**winning strategy**", e.g., for White (Player 1) is a pure strategy s_1^* that guarantees value $u(s_1^*, s_2) = 1$.

Question: Which one is the right answer??
Problem: The tree is far too big!!

Even with ~ 200 depth & ~ 5 moves per node:

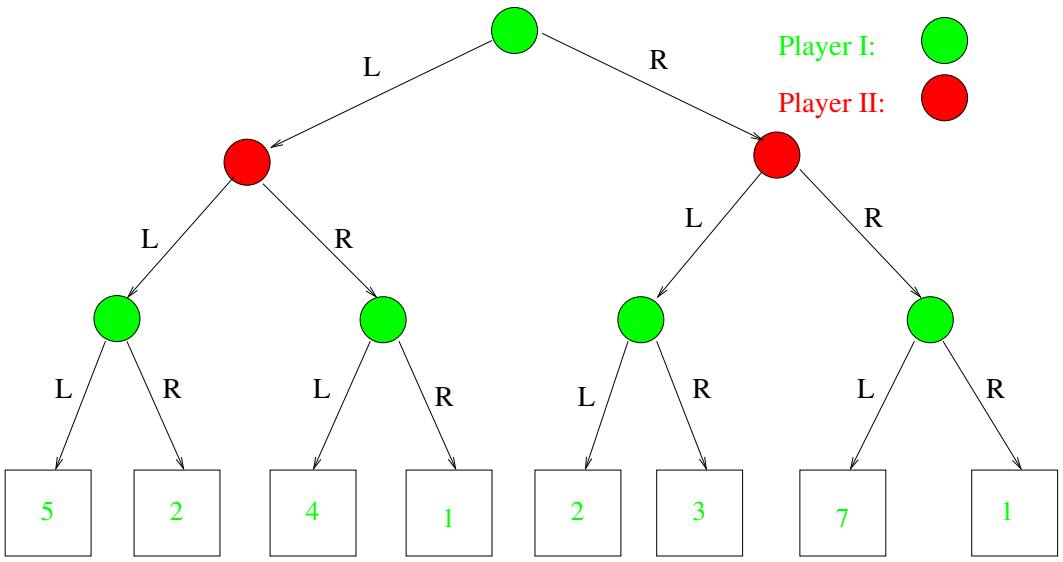
$$5^{200} \text{ nodes!}$$

Despite having an "efficient" algorithm to compute the value v^* given the tree, we can't even look at the whole tree! We need algorithms that don't look at the whole tree.

50 years of game-tree search

There's > 50 years of research on chess & other game playing programs, (Shannon, Turing, ...). Heuristic game-tree search procedures are now very refined. See any AI text (e.g., [Russel-Norvig'02]).

We can do a recursive "top-down" search in Depth-First style. If additionally we have a function *Eval*(w) that heuristically "evaluates" a node's "goodness" score, we can use *Eval*(w) to stop the search at, e.g., any desired depth.
While searching "top-down", we can "prune out" irrelevant subtrees using α-β-**pruning**. Idea: while searching the minmax tree, maintain two values:
α- "maximizer can assure a score of at least α";
β- "minimizer can assure a score of at most β";



minmax search with α-β-pruning

Assume, for simplicity, that players alternate moves, and root belongs to Player 1 (maximizer). Assume $-1 \leq \mathit{Eval}(w) \leq +1$. Score -1 ($+1$) means player 1 definitely loses (wins). Start the search by calling: **MaxVal**$(\epsilon, -1, +1)$;

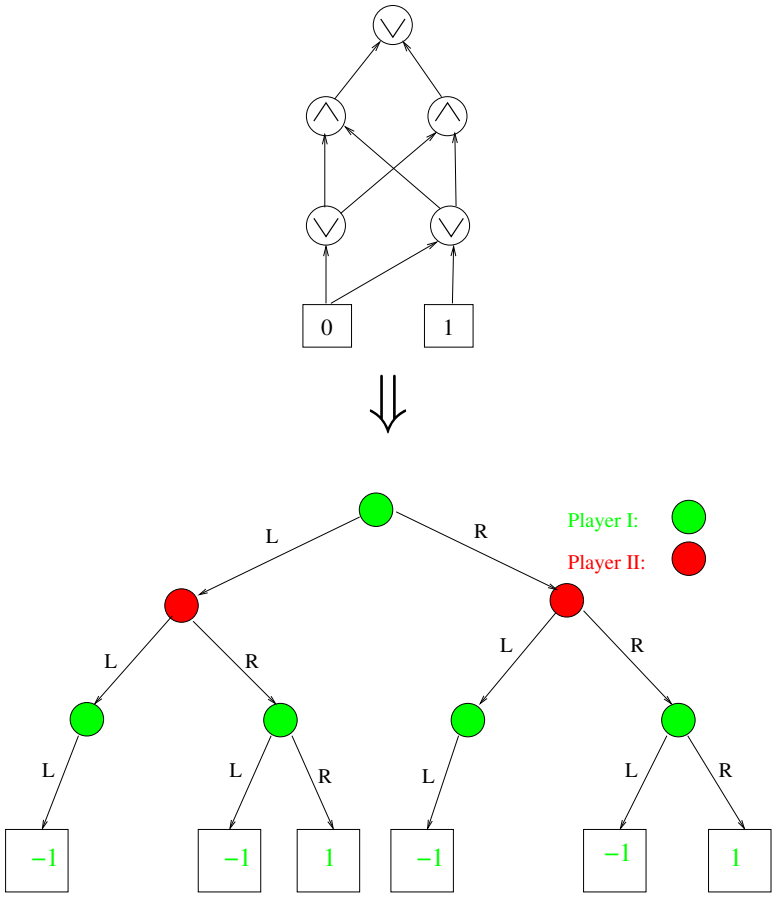
MaxVal(w, α, β)
  If *depth*$(w) \geq MaxDepth$ then **return** *Eval*(w).
  Else
    for each $a \in \mathit{Act}(w)$
      $\alpha := \max\{\alpha,$ **MinVal**$(wa, \alpha, \beta)\}$;
      if $\alpha \geq \beta$, then **return** β
  return α

MinVal(w, α, β)
  If *depth*$(w) \geq MaxDepth$, then **return** *Eval*(w).
  Else
    for each $a \in \mathit{Act}(w)$
      $\beta := \min\{\beta,$ **MaxVal**$(wa, \alpha, \beta)\}$;
      if $\beta \leq \alpha$, then **return** α
  return β

Heuristic game-tree search is a world of research onto itself. Please see, e.g., the references in AI texts.

boolean circuits as finite extensive PI-games

Boolean circuits can be viewed as a zero-sum PI-game, between AND and OR: OR the maximizer, AND the minimizer. (negations "pushed down" to bottom.) This is a **win-lose** PI-game: no chance nodes and the only possible payoffs $u(s_1, s_2)$ are 1 and -1.



Note: game tree can be exponentially bigger, but efficient bottom-up algorithm works directly on circuit.

Ok, let's try to generalize to infinite zero-sum games

For a (possibly infinite) zero-sum 2-player PI-game, we would like to similarly define the game to be "determined" if

$$\max_{s_1 \in S_1} \min_{s_2 \in S_2} u(s_1, s_2) = \min_{s_2 \in S_2} \max_{s_1 \in S_1} u(s_1, s_2)$$

But, for infinite games $\max$ & $\min$ may not exist! Instead, let's define an (infinite) zero-sum game to be **determined** (**have a value**) if:

$$\sup_{s_1 \in S_1} \inf_{s_2 \in S_2} u(s_1, s_2) = \inf_{s_2 \in S_2} \sup_{s_1 \in S_1} u(s_1, s_2)$$

In the simple setting of infinite win-lose PI-games (2 players, zero-sum, no chance nodes, and only payoffs are 1 and -1), this definition says a game is determined precisely when one player or the other has a **winning strategy**: a strategy $s_1^* \in S_1$ such that for any $s_2 \in S_2$, $u(s_1^*, s_2) = 1$ (and vice versa for player 2).

Question: Is every win-lose PI-game determined?

Answer: No .....

determinacy and its boundaries

For win-lose PI-games, we can define the payoff function by providing the set $Y = u_1^{-1}(1) \subseteq \Psi_T$, of complete plays on which player 1 wins (player 2 necessarily wins on all other plays).
If, additionally, we assume that players alternate moves, we can specify such a game as $\mathcal{G} = \langle T, Y \rangle$.

Fact Even for the binary tree $T = \{L, R\}^*$, there are sets $Y \subseteq \Psi_T$, such that the win-lose PI-game $\mathcal{G} = \langle T, Y \rangle$ is not determined.

(This fact is not very easy to show. Its proof uses the "axiom of choice". See, e.g., [Mycielski, Ch. 3 of Handbook of GT,1992].)

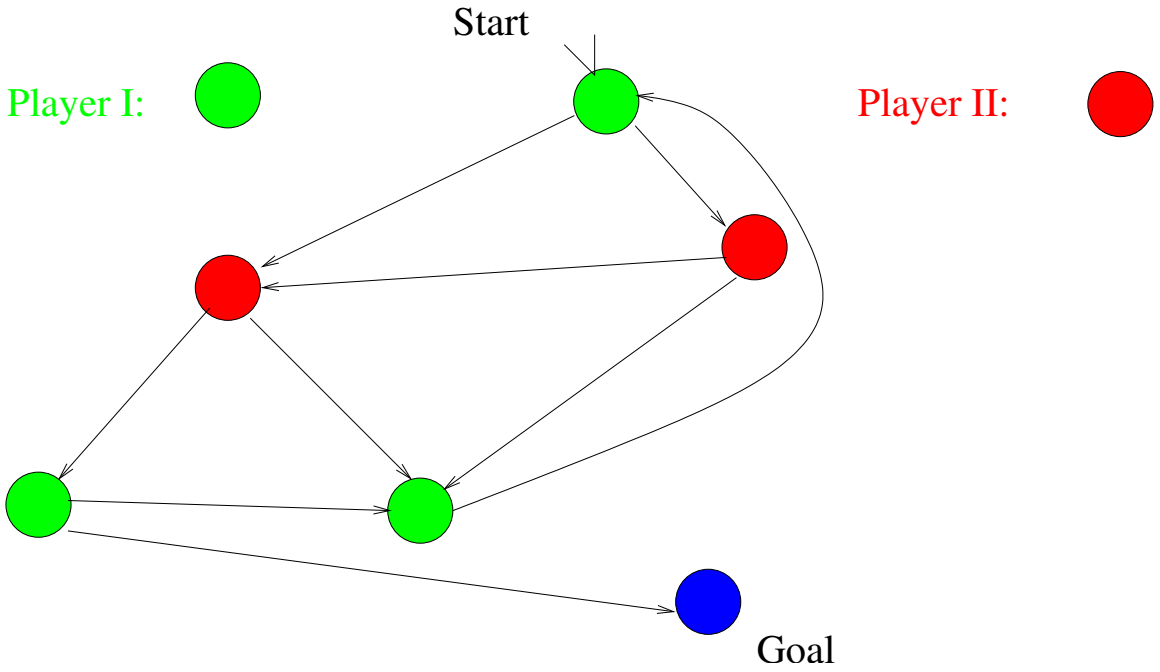
Fortunately, large classes of win-lose PI-games are determined. In particular:

Theorem([D. A. Martin'75]) Whenever Y is a so called "*Borel set*", the game $\langle \Sigma^*, Y \rangle$ is determined.

(This is a deep theorem, with deep connections to logic and set theory. The theorem holds even when the action alphabet Σ is an infinite set.)

food for thought: win-lose games on finite graphs

Instead of a tree, we have a finite directed graph:



- Starting at "Start", does Player I have a strategy to "force" the play to reach the "Goal"?
- Convince yourself that this is a (possibly infinite) win-lose PI-game.
- Is this game determined for all finite graphs?
- If so, how would you compute a winning strategy for Player 1?