Lecture 4a: Game Playing (Adversarial Search)
CS 580 (001) - Spring 2016

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

February 17, 2016
1 Outline of Today’s Class

2 Games vs Search Problems

3 Perfect Play
   ■ Minimax Decision
   ■ Alpha-Beta Pruning

4 Resource Limits and Approximate Evaluation
   ■ Expectimiminimax

5 Games of Imperfect Information

6 Game Playing Summary
Search in a multi-agent, competitive environment → Adversarial Search/Game Playing

Mathematical game theory treats any multi-agent environment as a game, with possibly co-operative behaviors (study of economies)

Most games studied in AI:

deterministic, turn-taking, two-player, zero-sum games of perfect information
Most games studied in AI:

deterministic, turn-taking, two-player, zero-sum games of perfect information

zero-sum: utilities of the two players sum to 0 (no win-win)
deterministic: precise rules with known outcomes
perfect information: fully observable
Most games studied in AI:

- **deterministic**, **turn-taking**, **two-player**, **zero-sum** games of **perfect information**

**zero-sum**: utilities of the two players sum to 0 (no win-win)

**deterministic**: precise rules with known outcomes

**perfect information**: fully observable

Search algorithms designed for such games make use of interesting general techniques (meta-heuristics) such as evaluation functions, search pruning, and more.

However, games are to AI what grand prix racing is to automobile design.
Most games studied in AI: deterministic, turn-taking, two-player, zero-sum games of perfect information

zero-sum: utilities of the two players sum to 0 (no win-win)

deterministic: precise rules with known outcomes

perfect information: fully observable

Search algorithms designed for such games make use of interesting general techniques (meta-heuristics) such as evaluation functions, search pruning, and more.

However, games are to AI what grand prix racing is to automobile design.

Our objective: study the three main adversarial search algorithms [minimax, alpha-beta pruning, and expectiminimax] and meta-heuristics they employ.
Two turn-taking agents in a zero-sum game: Max (starts game) and Min
Max’s goal is to maximize its utility  Min’s goal is to minimize Max’s utility
Game Playing as a Search Problem

Formal definition of a game as a search problem:

- $S_0$← initial state that specifies how game starts
- $\text{PLAYER}(s)$← which player has move in state $s$
- $\text{RESULT}(s, a)$← transition model that defines result of an action $a$ on a state $s$
- $\text{TERMINAL-TEST}(s)$← true on states that are game enders, false otherwise
- $\text{UTILITY}(s, p)$← utility/objective function defines numeric value for game that ends in terminal state $s$ with player $p$

Concept of game/search tree valid here

- Chess: 35 moves per player $\rightarrow$ branching factor $b = 35$
- ends at typically 50 moves $\rightarrow m = 100$
- search tree has $35^{100} \approx 10^{40}$ distinct nodes

**Pruning:** how to ignore portions of tree without impacting strategy

**Evaluation function:** estimate utility of a state without a complete search

Some games too big search trees:

- Time limits $\Rightarrow$ unlikely to find goal, must approximate
- Many “tricks” (meta-heuristics) employed to look ahead
Early Obsession with Games before Term AI Coined

- Computer considers possible lines of play (Babbage, 1846)
- Algorithm for perfect play (Zermelo, 1912; Von Neummann, 1944)
- Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- First chess program (Turing, 1951)
- Machine learning to improve evaluation accuracy (Samuel, 1952–57)
- Pruning to allow deeper search (McCarthy, 1956)
- ...
- Today, Alphabet’s deep learning team claims to have a Go-playing program that will beat world masters
Game Tree (Two-player, Deterministic, Turns)

MAX (X)

MIN (O)

MAX (X)

MIN (O)

TERMINAL

Utility

-1  0  +1
Minimax Decisions

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest minimax value

= best achievable payoff against best play

E.g., 2-ply game:
function Minimax-Decision(state) returns an action
    inputs: state, current state in game
    return the a in Actions(state) maximizing Min-Value(Result(a, state))

function Max-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← −∞
    for a, s in Successors(state) do v ← Max(v, Min-Value(s))
    return v

function Min-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← ∞
    for a, s in Successors(state) do v ← Min(v, Max-Value(s))
    return v
Properties of Minimax

Complete?

Yes, if tree is finite (chess has specific rules for this)

Optimal?

Yes, against an optimal opponent. Otherwise?

Time complexity?

$O(b^m)$

Space complexity?

$O(b^m)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games

$\Rightarrow$ exact solution completely infeasible

Do we need to explore every path?

Amarda Shehu (580)
Properties of Minimax

Complete? Yes, if tree is finite (chess has specific rules for this)
Properties of Minimax

Complete? Yes, if tree is finite (chess has specific rules for this)

Optimal? Yes, against an optimal opponent. Otherwise?
Properties of Minimax

Complete?

Yes, if tree is finite (chess has specific rules for this)

Optimal?

Yes, against an optimal opponent. Otherwise?
Properties of Minimax

Complete??? Yes, if tree is finite (chess has specific rules for this)

Optimal??? Yes, against an optimal opponent. Otherwise??

Time complexity???
Properties of Minimax

Complete? Yes, if tree is finite (chess has specific rules for this)

Optimal? Yes, against an optimal opponent. Otherwise?

Time complexity? $O(b^m)$
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? \( O(b^m) \)

Space complexity??
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? \( O(b^m) \)

Space complexity?? \( O(bm) \) (depth-first exploration)

For chess, \( b \approx 35, m \approx 100 \) for “reasonable” games
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

For chess, $b \approx 35, m \approx 100$ for “reasonable” games

$\Rightarrow$ exact solution completely infeasible
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games
  $\Rightarrow$ exact solution completely infeasible

Do we need to explore every path?
Properties of Minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games

⇒ exact solution completely infeasible

Do we need to explore every path?
$\alpha - \beta$ Pruning Example

MAX

MIN

3
12
8

$\geq 3$
\( \alpha - \beta \) Pruning Example
$\alpha - \beta$ pruning example

MAX

MIN

$\geq 3$

$\leq 2$

$\leq 14$

3

12

8

2

14
\( \alpha - \beta \) Pruning Example
\( \alpha - \beta \) Pruning Example

```plaintext
MAX

MIN

3  
12 
8  
2 

3  
\leq 2 
2

3  
14 
5  
2
```

Amarda Shehu (580)
Why is it Called $\alpha-\beta$?

$\alpha$ is the best value (to MAX) found so far off the current path

If $V$ is worse than $\alpha$, MAX will avoid it $\Rightarrow$ prune that branch

Define $\beta$ similarly for MIN
The $\alpha-\beta$ Algorithm

function $\text{Alpha-Beta-Decision}(\text{state})$ returns an action
return $a$ in $\text{Actions}(\text{state})$ maxim. $\text{Min-Value}(\text{Result}(a, \text{state}))$

function $\text{Max-Value}(\text{state}, \alpha, \beta)$ returns a utility value
inputs: $\text{state}$, current state in game
$\alpha$, value of best alternative for $\text{MAX}$ along the path to $\text{state}$
$\beta$, value of best alternative for $\text{MIN}$ along the path to $\text{state}$

if $\text{Terminal-Test}(\text{state})$ then return $\text{Utility}(\text{state})$
$v \leftarrow -\infty$
for $a, s$ in $\text{Successors}(\text{state})$ do
$v \leftarrow \text{Max}(v, \text{Min-Value}(s, \alpha, \beta))$
if $v \geq \beta$ then return $v$
$\alpha \leftarrow \text{Max}(\alpha, v)$
return $v$

function $\text{Min-Value}(\text{state}, \alpha, \beta)$ returns a utility value
same as $\text{Max-Value}$ but with roles of $\alpha, \beta$ reversed
Properties of $\alpha-\beta$

Pruning **does not** affect final result

Good move ordering improves effectiveness of pruning

With “perfect ordering,” time complexity $= O(b^{m/2})$
\[ \Rightarrow \text{doubles solvable depth} \]

A simple example of the value of reasoning about which computations are relevant (a form of **metareasoning**)

Unfortunately, $35^{50}$ is still impossible!
Resource Limits

Standard approach:

- Use **Cutoff-Test** instead of **Terminal-Test**
  e.g., depth limit (perhaps add quiescence search)
- Use **Eval** instead of **Utility**
  i.e., **evaluation function** that estimates desirability of position

Suppose we have 100 seconds, explore $10^4$ nodes/second
⇒ $10^6$ nodes per move $\approx 35^{8/2}$
⇒ $\alpha-\beta$ reaches depth 8 ⇒ pretty good chess program
For chess, typically linear weighted sum of features

\[\text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)\]

e.g., \(w_1 = 9\) with \(f_1(s) = (\text{number of white queens}) - (\text{number of black queens})\), etc.
Digression: Exact Values do not Matter

Behaviour is preserved under any **monotonic** transformation of $\text{Eval}$

Only the order matters:
- payoff in deterministic games acts as an **ordinal utility** function
Deterministic Games in Practice

- **Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

- **Chess:** Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

- **Othello:** human champions refuse to compete against computers, who are too good.

- **Go:** human champions refused to compete against computers, who were too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves. Great progress made by Alphabet via deep learning: competitive playing results expected this summer!
Nondeterministic Games: Backgammon
In nondeterministic games, chance introduced by dice, card-shuffling
Simplified example with coin-flipping:
Algorithm for Nondeterministic Games

Expectiminimax gives perfect play
Just like Minimax, except we must also handle chance nodes:

\[\textbf{if } \text{state is a Max node then}\]

\textbf{return} the highest \textbf{ExpectiMinimax-Value of Successors}(\textit{state})

\[\textbf{if } \text{state is a Min node then}\]

\textbf{return} the lowest \textbf{ExpectiMinimax-Value of Successors}(\textit{state})

\[\textbf{if } \text{state is a chance node then}\]

\textbf{return} average of \textbf{ExpectiMinimax-Value of Successors}(\textit{state})

\[\ldots\]
Dice rolls increase $b$: 21 possible rolls with 2 dice
Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)

\[ \text{depth} \ 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9 \]

As depth increases, probability of reaching a given node shrinks
$\Rightarrow$ value of lookahead is diminished

$\alpha-\beta$ pruning is much less effective

TDGammon uses depth-2 search + very good Eval
$\approx$ world-champion level
Behaviour is preserved only by positive linear transformation of \( \text{Eval} \)

Hence \( \text{Eval} \) should be proportional to the expected payoff
E.g., card games, where opponent’s initial cards are unknown

Typically we can calculate a probability for each possible deal

Seems just like having one big dice roll at the beginning of the game*

**Idea:**
- compute the minimax value of each action in each deal,
- then choose the action with highest expected value over all deals*

Special case: if an action is optimal for all deals, it’s optimal.*

GIB, current best bridge program, approximates this idea by
  1) generating 100 deals consistent with bidding information
  2) picking the action that wins most tricks on average
Example

Four-card bridge/whist/hearts hand, **MAX** to play first
Example

Four-card bridge/whist/hearts hand, \text{MAX} to play first
Example

Four-card bridge/whist/hearts hand, MAX to play first
Commonsense Example

Road A leads to a small heap of gold pieces
Road B leads to a fork:
  take the left fork and you’ll find a mound of jewels;
  take the right fork and you’ll be run over by a bus.
Road A leads to a small heap of gold pieces
Road B leads to a fork:
  take the left fork and you’ll find a mound of jewels;
  take the right fork and you’ll be run over by a bus.

Road A leads to a small heap of gold pieces
Road B leads to a fork:
  take the left fork and you’ll be run over by a bus;
  take the right fork and you’ll find a mound of jewels.
Road A leads to a small heap of gold pieces
Road B leads to a fork:
   take the left fork and you’ll find a mound of jewels;
   take the right fork and you’ll be run over by a bus.

Road A leads to a small heap of gold pieces
Road B leads to a fork:
   take the left fork and you’ll be run over by a bus;
   take the right fork and you’ll find a mound of jewels.

Road A leads to a small heap of gold pieces
Road B leads to a fork:
   guess correctly and you’ll find a mound of jewels;
   guess incorrectly and you’ll be run over by a bus.
Proper Analysis

* Intuition that the value of an action is the average of its values in all actual states is **WRONG**

With partial observability, value of an action depends on the **information state** or **belief state** the agent is in

Can generate and search a tree of information states

Leads to rational behaviors such as

◊ Acting to obtain information

◊ Signalling to one’s partner

◊ Acting randomly to minimize information disclosure
Games are fun to work on! (and dangerously obsessive)

Illustrate several important points about AI

◊ perfection is unattainable \(\Rightarrow\) must approximate
◊ good idea to think about what to think about
◊ uncertainty constrains the assignment of values to states
◊ optimal decisions depend on information state, not real state

◊ Domain-specific tricks can be generalized to meta-heuristics of possible relevance for search of complex state spaces