Adversarial Search

Chapter 6
Sections 1 – 3 (and a little of the rest)
Outline

• Games
• Optimal decisions
• $\alpha$-$\beta$ pruning
• Imperfect, real-time decisions (briefly)
Game Search

- Game-playing programs developed by AI researchers since the beginning of the modern AI era (chess, checkers in 1950s)

- Game Search
  - Sequences of player’s decisions *we control*
  - Decision of other player(s) *we do not control*

- Contingency problem: many possible opponent’s moves must be “covered” by the solution
  - Introduces uncertainty to the game since we do not know what the opponent will do

- Rational opponent: maximizes it’s own utility function
Types of Game Problems

• Adversarial
  – Win of one player is a loss of the other
  – Focus of this course

• Cooperative
  – Players have common interests and utility function

• A spectrum of others in between
Games vs. search problems

- Adversarial search or “games”
  - Multi-agent and competitive environment
  - "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate
Typical AI “Games:

- Deterministic and Fully Observable Environment
- Two agents with turn-taking for actions
- Zero-sum (adverserial)
- Abstract (robotic soccer notable exception)
  - state easy to represent, few action choices, well-defined goals
  - hard to solve
Game Search

• Problem Formulation
  – **Initial state**: initial board position + information about whose move it is
  – **Successors**: legal moves a player can make
  – **Goal (terminal test)**: determines when the game is over
  – **Utility function**: measures the outcome of the game and its desirability

• Search objective
  – Find the sequence of player’s decisions (moves) maximizing its utility
  – Consider the opponent’s moves and their utility
Game Tree

• Initial State and Legal Moves for Each Side
Game Tree
(2-player, deterministic, turns)
Game Tree
(2-player, deterministic, turns)

- MAX and MIN are the 2 players
- MAX goes first
- Players then take turns
Game Tree
(2-player, deterministic, turns)

- MAX has 9 possible legal first moves (ignoring symmetry)
Game Tree
(2-player, deterministic, turns)

Utility of terminal states (when game is over) is from MAX’s point of view

Points are awarded to both players at the end of the game
- -1 is a loss
- 0 is a draw
- 1 is a win
Minimax Algorithm

• How do we deal with the contingency problem?

  – Assuming that the opponent is rational and always optimizes its behavior (opposite to us), we consider the opponent’s best response
  – Then the minimax algorithm determines the best move
Minimax

- Finds an optimal (contingent) strategy, assuming perfect play for deterministic games

- Idea: choose move to position with highest MINIMAX VALUE = best achievable payoff against best play

- MINIMAX-VALUE ($n$)
  - $\text{UTILITY} \ (n)$ if $n$ is a terminal state
  - $\max_s \text{MINIMAX-VALUE} \ (s)$ if $n$ is a MAX node
  - $\min_s \text{MINIMAX-VALUE} \ (s)$ if $n$ is a MIN node
    (where $s$ is an element of the successors of $n$)
Minimax Example

- E.g., 2-ply game (with utility values at the leaves)
Another Example

• In class
Minimax algorithm

function MINIMAX-DECISION(state) returns an action
    \( v \leftarrow \text{MAX-VALUE}(state) \)
    return the action in SUCCESSORS(state) with value \( v \)

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    \( v \leftarrow -\infty \)
    for \( a, s \) in SUCCESSORS(state) do
        \( v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s)) \)
    return \( v \)

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    \( v \leftarrow \infty \)
    for \( a, s \) in SUCCESSORS(state) do
        \( v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s)) \)
    return \( v \)

Notes: recursive (backs up from leaves), depth-first
Properties of minimax

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
- **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 really need to explore every path???
Solutions to the Complexity Problem

• Dynamic pruning of redundant branches of the search tree
  – Some branches will never be played by rational players since they include sub-optimal decisions (for either player)
    • Identify a provably suboptimal branch of the search tree before it is fully explored
    • Eliminate the suboptimal branch
  – Procedure: Alpha-Beta Pruning

• Early cutoff of the search tree
  – Use imperfect minimax value estimate of non-terminal states
α-β pruning example
α-β pruning example
$\alpha$-$\beta$ pruning example
α-β pruning example
α-β pruning example
$$\alpha$$-$$\beta$$ pruning example

MINIMAX-VALUE(root)
= max(min(3,12,8), min(2,x,y), min(14,5,2))
= max (3, min(2,x,y), 2)
= max(3, z, 2) for z <= 2
= 3
Properties of $\alpha$-$\beta$

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)
Why is it called $\alpha$-$\beta$?

- $\alpha$ is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for $max$.

- If $v$ is worse than $\alpha$, $max$ will avoid it → prune that branch.

- Similarly, $\beta$ is the value of the best (i.e., lowest-value) choice found so far at any choice point along the path for $min$. 

![Diagram of a tree search with alpha beta pruning](image)
Another Example

• In class
The $\alpha$-$\beta$ algorithm

function $\text{Alpha-Beta-Search}(state)$ returns an action

inputs: $state$, current state in game

$v$ ← $\text{Max-Value}(state, -\infty, +\infty)$

return the action in $\text{Successors}(state)$ with value $v$

function $\text{Max-Value}(state, \alpha, \beta)$ returns a utility value

inputs: $state$, current state in game

$\alpha$, the value of the best alternative for $\text{Max}$ along the path to $state$

$\beta$, the value of the best alternative for $\text{Min}$ along the path to $state$

if $\text{Terminal-Test}(state)$ then return $\text{Utility}(state)$

$v$ ← $-\infty$

for $a, s$ in $\text{Successors}(state)$ do

$v$ ← $\text{Max}(v, \text{Min-Value}(s, \alpha, \beta))$

if $v \geq \beta$ then return $v$

$\alpha$ ← $\text{Max}(\alpha, v)$

return $v$
The \( \alpha-\beta \) algorithm

\begin{verbatim}
function \textsc{Min-Value}(state, \alpha, \beta) returns a utility value
    inputs: state, current state in game
            \alpha, the value of the best alternative for \textsc{Max} along the path to state
            \beta, the value of the best alternative for \textsc{Min} along the path to state

    if \textsc{Terminal-Test}(state) then return \textsc{Utility}(state)
    \varepsilon \leftarrow +\infty
    for \ a, \ s \ in \textsc{Successors}(state) do
        \varepsilon \leftarrow \textsc{Min}(\varepsilon, \textsc{Max-Value}(s, \alpha, \beta))
        if \varepsilon \leq \alpha then return \varepsilon
        \beta \leftarrow \textsc{Min}(\beta, \varepsilon)
    return \varepsilon
\end{verbatim}
Resource limits (Section 6.4)

Recap

– Minimax explores the full search space
– Alpha Beta prunes, but still searches all the way to terminal states for a portion of the search space

Standard approaches to fix resource limits

– cutoff test:
  e.g., depth limit
– evaluation function
  = estimated desirability of position
Evaluation functions

• 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.
Cutting off search

\textit{MinimaxCutoff} is identical to \textit{MinimaxValue} except

1. \textit{Terminal?} is replaced by \textit{Cutoff?}
2. \textit{Utility} is replaced by \textit{Eval}

4-ply lookahead is a hopeless chess player!

- 4-ply \(\approx\) human novice
- 8-ply \(\approx\) typical PC, human master
- 12-ply \(\approx\) Deep Blue, Kasparov
Deterministic games in practice

- **Checkers**: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.

- **Chess**: Deep Blue defeated human world champion Garry 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 refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

- **AAAI conferences** now have *general* game-playing competitions, with a $10K$ prize!
Summary

• Games are fun to work on!
• They illustrate several important points about AI
• perfection is unattainable → must approximate
• good idea to think about what to think about