ADVERSARIAL SEARCH

Chapter 5

“... every game of skill is susceptible of being played by an automaton.”
Games

Perfect play
  – minimax decisions
  – $\alpha-\beta$ pruning

Resource limits and approximate evaluation

Games of chance

Games of imperfect information
Games vs. search problems

“Unpredictable” opponent ⇒ solution is a **strategy** specifying a move for every possible opponent reply

Time limits ⇒ unlikely to find goal, must approximate

Plan of attack:

• Computer considers possible lines of play (Babbage, 1846)
• Algorithm for perfect play (Zermelo, 1912; Von Neumann, 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)
# Types of Games

<table>
<thead>
<tr>
<th>Perfect Information</th>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>chess, checkers, go, othello</td>
<td>backgammon monopoly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Imperfect Information</th>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>battleships, blind tictactoe</td>
<td>bridge, poker, scrabble nuclear war</td>
</tr>
</tbody>
</table>
Game tree (2-player, deterministic, turns)

MAX (X)

MIN (O)

MAX (X)

MIN (O)

TERMINAL

Utility

−1  0  +1
Minimax

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:

\[
\begin{align*}
\text{MAX} & : & 3 \\
\text{MIN} & : & 3 \\
& & 12 \\
& & 8 \\
& & 6 \\
& & 4 \\
& & 2 \\
& & 14 \\
& & 5 \\
& & 2
\end{align*}
\]
Minimax algorithm

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??
Properties of minimax

Complete?? Only if tree is finite (chess has specific rules for this).
NB a finite strategy can exist even in an infinite tree!

Optimal??
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?? $O(b^m)$

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)

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

$\Rightarrow$ exact solution completely infeasible

But do we need to explore every path?
\( \alpha - \beta \) pruning example

\[
\begin{array}{c}
\text{MAX} \\
\text{MIN} \\
3 \quad 12 \quad 8 \\
\end{array}
\]

\[ \geq 3 \]
\[ \alpha - \beta \] pruning example

MAX

MIN

\[
\begin{array}{c}
3 \\
12 \\
8 \\
2 \\
\end{array}
\]

\[
\begin{array}{c}
\geq 3 \\
\leq 2 \\
X \\
X \\
\end{array}
\]
\(\alpha-\beta\) pruning example
$\alpha-\beta$ pruning example
\( \alpha-\beta \) pruning example

\[
\begin{array}{c}
\text{MAX} \\
3 \\
12 \\
8 \\
2 \\
\text{MIN} \\
3 \\
\leq 2 \\
14 \\
5 \\
2
\end{array}
\]
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
Properties of $\alpha - \beta$

Pruning does not affect final result

Good move ordering improves effectiveness of pruning

With “perfect ordering,” time complexity = $O(b^{m/2})$
  $\Rightarrow$ 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
  $\Rightarrow 10^6$ nodes per move $\approx 35^{8/2}$
  $\Rightarrow \alpha-\beta$ reaches depth 8 $\Rightarrow$ pretty good chess program
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
\[ f_1(s) = \text{(number of white queens)} - \text{(number of black queens)}, \text{ etc.} \]
Digression: Exact values don’t matter

Behaviour is preserved under any monotonic transformation of Eval

Only the order matters:

payoff in deterministic games acts as an ordinal utility function
How to achieve a good game of chess?

- Extensively tuned evaluation function
- Cutoff test with quiescence search
- Large transposition table [i.e., hash of previously seen positions, saved for re-use]
- Use of alpha-beta, with extra pruning
- Large database of optimal opening and endgame moves
- Fast computer!
Exercise – Tic-tac-toe

• Define $X_n$ as the number of rows, columns, or diagonals with exactly $n$ X’s and no O’s. Similarly, $O_n$ is the number of rows, columns, or diagonals with exactly $n$ O’s and no X’s.

• The utility function assigns +1 to any position with $X_3 = 1$ and -1 for any position with $O_3 = 1$. All other terminal positions have utility 0.

• For non-terminal positions, we use a linear evaluation function defined as $Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$

a) Approximately how many games of tic-tac-toe are there?
Exercise – Tic-tac-toe

• Define $X_n$ as the number of rows, columns, or diagonals with exactly $n$ X’s and no O’s. Similarly, $O_n$ is the number of rows, columns, or diagonals with exactly $n$ O’s and no X’s.

• The utility function assigns +1 to any position with $X_3 = 1$ and -1 for any position with $O_3 = 1$. All other terminal positions have utility 0.

• For non-terminal positions, we use a linear evaluation function defined as $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$

a) Approximately how many games of tic-tac-toe are there?

$9! = \text{the number of move sequences that fill up the board (although many wins and losses occur before that)}$
b) What does the game tree look like (taking symmetry into account)?
Exercise – Prove correctness of $\alpha$-$\beta$

- Question is whether to prune $n_j$, which is a max-node and descendent of $n_1$
- Basic idea is to prune it iff the minimax value of $n_1$ can be shown to be independent of the value of $n_j$
- Node $n_1$ takes on the minimum value among its children $n_1 = \min(n_2, n_{21}, ..., n_{2b_2})$. Find a similar expression for $n_2$ and hence an expression for $n_1$ in terms of $n_j$. 
Nondeterministic games: backgammon
In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Chapter 6
Algorithm for nondeterministic games

\texttt{EXPECTIMINIMAX} gives perfect play

Just like \texttt{MINIMAX}, except we must also handle chance nodes:

\ldots

\textbf{if} \textit{state} is a \texttt{Max} node \textbf{then} \\
\hspace*{1cm} \textbf{return} the highest \texttt{EXPECTIMINIMAX-VALUE} of \texttt{SUCCESSORS(}\textit{state}\texttt{)}

\textbf{if} \textit{state} is a \texttt{Min} node \textbf{then} \\
\hspace*{1cm} \textbf{return} the lowest \texttt{EXPECTIMINIMAX-VALUE} of \texttt{SUCCESSORS(}\textit{state}\texttt{)}

\textbf{if} \textit{state} is a chance node \textbf{then} \\
\hspace*{1cm} \textbf{return} average of \texttt{EXPECTIMINIMAX-VALUE} of \texttt{SUCCESSORS(}\textit{state}\texttt{)}

\ldots
Nondeterministic games in practice

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
Digression: Exact values DO matter

MAX

DICE

MIN

Behaviour is preserved only by positive linear transformation of Eval

Hence Eval should be proportional to the expected payoff
Games of imperfect information

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
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.
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 be run over by a bus;
    take the right fork and you’ll find a mound of jewels.
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 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
Summary

Games are fun to work on! (and dangerous)
They illustrate several important points about AI

◊ perfection is unattainable ⇒ 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

Games are to AI as grand prix racing is to automobile design
Suppose you have a chess program that can evaluate 10 million nodes per second. (There are approximately $10^{47}$ legal game positions in chess.)

a) What is a compact representation of a game state for storage in a transposition table? (Note that there are 32 pieces in chess, and 64 squares on the board. Presume 8-bit bytes.)
Reading Assignment

• For next week: Chapter 7 – Logical Agents
  – Tuesday: 7.1-7.4
  – Thursday: 7.5-7.7