Adversarial Search (Game playing)

Chapter 6
Outline

◊ Games
◊ Perfect play (minimax)
◊ $\alpha-\beta$ 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)
<table>
<thead>
<tr>
<th>Types of games</th>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect information</td>
<td>chess, checkers, go, othello</td>
<td>backgammon, monopoly</td>
</tr>
<tr>
<td>Imperfect information</td>
<td>battleships, blind tictactoe</td>
<td>bridge, poker, scrabble, nuclear war</td>
</tr>
</tbody>
</table>
Game tree (2-player, deterministic, turns)
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:

```
MAX

A₁
  /\  \
A₂  A₃
  /\  /\  \
A₁₁ A₁₂ A₁₃ A₂₁ A₂₂ A₂₃ A₃₁ A₃₂ A₃₃

MIN

3
  /\  \
12 8
  /\  /\  
3 12 8 2 4 6 14 5 2
```

Chapter 6  6
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?? Yes, if tree is finite (chess has specific rules for this)

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

MAX

MIN

3

12

8

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

MAX

MIN

3
12
8
2

$\geq 3$

$\leq 2$

X
X
\[ \alpha - \beta \text{ pruning example} \]

MAX

MIN

3 12 8 2 14

3 3 X

2 X

\leq 2 \leq 14
$\alpha - \beta$ pruning example
\[ \alpha - \beta \text{ pruning example} \]

MAX

MIN

\[
\begin{aligned}
&3 \\
&12 \\
&8 \\
&2 \\
&14 \\
&5 \\
&2 \\
\end{aligned}
\]
**Why is it called \( \alpha - \beta \)?**

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

If \( V \) is worse than \( \alpha \), \( \text{MAX} \) will avoid it \( \Rightarrow \) prune that branch

Define \( \beta \) similarly for \( \text{MIN} \)
The $\alpha$–$\beta$ algorithm

function \textsc{Alpha-Beta-Decision}(state) returns an action
return the $a$ in \textsc{Actions}(state) maximizing \textsc{Min-Value}(\textsc{Result}(a, state))

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

function \textsc{Min-Value}(state, $\alpha$, $\beta$) returns a utility value
same as \textsc{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 = $O(b^{m/2})$
  \[ \Rightarrow \text{doubles depth of search} \]
  \[ \Rightarrow \text{can easily reach depth 8 and play good chess} \]

A simple example of the value of reasoning about which computations are relevant
Resource limits

Suppose we have 100 seconds, explore $10^4$ nodes/second
\[\Rightarrow 10^6 \text{ nodes per move}\]

Standard approach:

- **cutoff test**
  - e.g., depth limit (perhaps add *quiescence search*)

- **evaluation function**
  - = estimated desirability of position
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 $\text{Eval}$

Only the order matters:
- payoff in deterministic games acts as an *ordinal utility* function
Cutting off search

MinimaxCutoff is identical to MinimaxValue except
1. Terminal? is replaced by Cutoff?
2. Utility is replaced by Eval

Does it work in practice?

\[ b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4 \]

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 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.


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.
Nondeterministic games: backgammon
In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:

```
          MAX
             ▲
           /   \
CHANCE    ▼   ▼
           3    -1
          /  \
MIN      ▼  ▼
          2    4
         /  \
 MIN   ▼  ▼
        2    4
                  ▼
                  2
```
Algorithm for nondeterministic games

\textsc{Expectiminimax} gives perfect play

Just like \textsc{Minimax}, except we must also handle chance nodes:

\[\ldots\]
\textbf{if} \textit{state} is a \textsc{Max} node \textbf{then}
\hspace{1em} \textbf{return} the highest \textsc{Expectiminimax-Value of Successors}(\textit{state})
\textbf{if} \textit{state} is a \textsc{Min} node \textbf{then}
\hspace{1em} \textbf{return} the lowest \textsc{Expectiminimax-Value of Successors}(\textit{state})
\textbf{if} \textit{state} is a chance node \textbf{then}
\hspace{1em} \textbf{return} average of \textsc{Expectiminimax-Value of Successors}(\textit{state})
\[\ldots\]
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

Behaviour is preserved only by \textit{positive linear} transformation of \texttt{Eval}

Hence \texttt{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
Example

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

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

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

MAX
6♥ 6♦ 8♣ 7♠ 8♣ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ 6♦ 6♥ 7♠ 0

MIN
4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 0

MAX
6♥ 6♦ 8♣ 7♠ 8♣ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ 0

MIN
4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 0

MAX
6♥ 6♦ 8♣ 7♠ 8♣ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ -0.5

MIN
4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 0

MAX
6♥ 6♦ 8♣ 7♠ 8♣ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ 6♥ 6♦ 7♠ -0.5

MIN
4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 4♥ 9♦ 3♣ 4♠ 0

Chapter 6
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