GAME PLAYING

CHAPTER 5, SECTIONS 1–5
Outline

◊ Perfect play
◊ Resource limits
◊ $\alpha-\beta$ pruning
◊ 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>chess, checkers, go, othello</td>
<td>backgammon monopoly</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Imperfect information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bridge, poker, scrabble nuclear war</td>
<td></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 \textit{minimax value} = best achievable payoff against best play

E.g., 2-ply game:

\begin{itemize}
  \item MAX
  \begin{itemize}
    \item A_1
    \begin{itemize}
      \item A_{11}
      \begin{itemize}
        \item 3
      \end{itemize}
    \end{itemize}
  \end{itemize}
  \item A_2
  \begin{itemize}
    \item A_{21}
    \begin{itemize}
      \item 2
    \end{itemize}
  \end{itemize}
  \item A_3
  \begin{itemize}
    \item A_{31}
    \begin{itemize}
      \item 4
    \end{itemize}
  \end{itemize}
\end{itemize}

\item MIN
  \begin{itemize}
    \item 3
    \begin{itemize}
      \item A_{12}
      \begin{itemize}
        \item 12
      \end{itemize}
    \end{itemize}
    \item 8
    \begin{itemize}
      \item A_{22}
      \begin{itemize}
        \item 2
      \end{itemize}
    \end{itemize}
    \item 6
    \begin{itemize}
      \item A_{32}
      \begin{itemize}
        \item 14
      \end{itemize}
    \end{itemize}
    \item 5
    \begin{itemize}
      \item A_{33}
      \begin{itemize}
        \item 2
      \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
Minimax algorithm

function MINIMAX-DECISION(state, game) returns an action
    action, state ← the a, s in SUCCESSORS(state)
    such that MINIMAX-VALUE(s, game) is maximized
    return action

function MINIMAX-VALUE(state, game) returns a utility value
    if TERMINAL-TEST(state) then
        return UTILITY(state)
    else if MAX is to move in state then
        return the highest MINIMAX-VALUE of SUCCESSORS(state)
    else
        return the lowest MINIMAX-VALUE of SUCCESSORS(state)
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
Resource limits

Suppose we have 100 seconds, explore $10^4$ nodes/second
$\Rightarrow 10^6$ nodes per move

Standard approach:

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

• evaluation function
  = estimated desirability of position
Evaluation functions

For chess, typically linear weighted sum of features

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

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 \textit{monotonic} transformation of $\text{EVAL}$.

Only the order matters:
- payoff in deterministic games acts as an \textit{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
\[ \alpha - \beta \text{ pruning example} \]
$\alpha-\beta$ pruning example

MAX

MIN

\begin{center}
\begin{tikzpicture}
  \node (3) at (0,0) {3} ;
  \node (12) at (1,-1) {12} ;
  \node (8) at (2,-1) {8} ;
  \node (2) at (3,-1) {2} ;
  \node (3) at (4,0) {$\geq 3$} ;
  \node (2) at (4,-1) {$\leq 2$} ;
  \path [->] (3) -- (12) ;
  \path [->] (3) -- (8) ;
  \path [->] (3) -- (2) ;
  \path [->] (2) -- (X) ;
  \path [->] (2) -- (X) ;
\end{tikzpicture}
\end{center}
$\alpha-\beta$ pruning example

MAX

MIN

3
12
8
2

3
≤ 2
X
X

≤ 14
\( \alpha - \beta \) pruning example

MAX

MIN

3

12

8

2

14

5

\( \geq 3 \)

\( \leq 2 \)

\( \leq 5 \)
\( \alpha - \beta \) pruning example
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* depth of search
  $\Rightarrow$ can easily reach depth 8 and play good chess

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 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 ALPHA-BETA-SEARCH($state$, $game$) returns an action
action, $state$ ← the $a, s$ in SUCCESSORS[$game$]($state$)
such that $\text{MIN-VALUE}(s, game, -\infty, +\infty)$ is maximized
return action

function MAX-VALUE($state$, $game$, $\alpha$, $\beta$) returns the minimax value of $state$
if CUTOFF-TEST($state$) then return EVAL($state$)
for each $s$ in SUCCESSORS($state$) do
$\alpha$ ← max($\alpha$, $\text{MIN-VALUE}(s, game, \alpha, \beta)$)
if $\alpha \geq \beta$ then return $\beta$
return $\alpha$

function MIN-VALUE($state$, $game$, $\alpha$, $\beta$) returns the minimax value of $state$
if CUTOFF-TEST($state$) then return EVAL($state$)
for each $s$ in SUCCESSORS($state$) do
$\beta$ ← min($\beta$, $\text{MAX-VALUE}(s, game, \alpha, \beta)$)
if $\beta \leq \alpha$ then return $\alpha$
return $\beta$
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.
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
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

Games are to AI as grand prix racing is to automobile design