Game playing

Chapter 5, Sections 1–5

Games vs. search problems

“Unpredictable” opponent ⇒ solution is a contingency plan
Time limits ⇒ unlikely to find goal, must approximate
Plan of attack:
- algorithm for perfect play (Von Neumann, 1944)
- finite horizon, approximate evaluation (Zuse, 1945; Shannon, 1950; Samuel, 1952–57)
- pruning to reduce costs (McCarthy, 1956)

Types of games

deterministic chance
perfect information
- chess, checkers, backgammon
- go, hanoi
imperfect information
- bridge, poker, scrabble
- nuclear war

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:

Minimax algorithm

\textbf{function} \textit{MINIMAX-DECISION}(game) \textit{returns} an operator
\begin{enumerate}
\item for each \textit{op} in \textit{OPERATORS}(game) do
\item \textit{VALUE[op]} = \textit{MINIMAX-VALUE}(\textit{APPLY}(op, \textit{game}), \textit{game})
\end{enumerate}
\textit{end}
\textit{return} the \textit{op} with the highest \textit{VALUE[op]}

\textbf{function} \textit{MINIMAX-VALUE}(node, game) \textit{returns} a utility value
\begin{enumerate}
\item if \textit{TERMINAL-TEST}(node) \textit{then}
\item \textit{return} \textit{UTILITY}(node, game)
\item else if \textit{MAX} \textit{is to move in state} \textit{then}
\item \textit{return} the highest \textit{MINIMAX-VALUE} of \textit{SUCCESSORS}(node)
\item else
\item \textit{return} the lowest \textit{MINIMAX-VALUE} of \textit{SUCCESSORS}(node)
\item \textit{end}
\end{enumerate}
Properties of minimax

- Complete??
- Optimal??
- Time complexity??
- Space complexity??

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

Resource limits

Suppose we have 100 seconds, explore \( 10^7 \) nodes/second
\( \Rightarrow \) \( 10^7 \) 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**
\[ \text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) \]
e.g., \( w_1 = 9 \) with
\[ f_1(s) = \text{(number of white queens)} - \text{(number of black queens)} \]
etc.

Digression: Exact values don't matter

MAX

MIN

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

Only the order matters:

\begin{align*}
\text{payoff in deterministic games acts as an } \text{ordinal utility function}
\end{align*}

Cutting off search

\( \text{MinimaxCutoff} \) is identical to \( \text{MinimaxValue} \) except

1. **Terminal?** is replaced by **Cutoff?**
2. **Utility** is replaced by **Eval**

Does it work in practice?

\( b^n = 10^7, \ b = 35 \ \Rightarrow \ m = 4 \)

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
**Properties of \( \alpha-\beta \)**

Pruning *does not* affect final result.

Good move ordering improves effectiveness of pruning.

With "perfect ordering," time complexity = \( O(b^{d/2}) \)

⇒ *double* depth of search

⇒ 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 *metarasoning*).
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

Basically  MINMAX + keep track of $\alpha$, $\beta$ + pure

function  MAX-VALUE(state, game, $\alpha$, $\beta$) returns the minimum value of $\alpha$, $\beta$ + pure
inputs: state, current state in game
          game, game description
          $\alpha$, the best score for MAX along the path to state
          $\beta$, the best score for MIN along the path to state
          if CUTOFF-TEST(state) then return EVAL(state)
          for each s in SUCCESSORS(state) do
            $\alpha$ := MAX($\alpha$, MIN-VALUE(s, game, $\alpha$, $\beta$))
          if $\beta$ $\geq$ $\alpha$ then return $\beta$
          end
          return $\alpha$

function  MIN-VALUE(state, game, $\alpha$, $\beta$) returns the maximum value of $\alpha$, $\beta$ + pure

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.

Algorithm for nondeterministic games

EXPECTIMINAX gives perfect play

Just like MINMAX, except we must also handle chance nodes:

... if state is a chance node then return average of EXPECTIMINAX-VALUE of SUCCESSORS(state) ...

A version of $\alpha$-$\beta$ pruning is possible but only if the leaf values are bounded. Why??

Non-deterministic games

E.g. in backgammon, the dice rolls determine the legal moves

Simplified example with coin-flipping instead of dice-rolling:

Non-deterministic 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)

depth 4 = $20 \times (21 \times 20)^4 \approx 1.2 \times 10^6$

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

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

TDG AMMON uses depth-2 search + very good EVAL $\approx$ world-champion level
Digression: Exact values DO matter

MAX

\[
\begin{array}{cccc}
2.1 & .9 & 21 & 20 \\
.1 & .9 & 1 & 40 \\
1.3 & .9 & 40.9 & 400 \\
\end{array}
\]

MIN

\[
\begin{array}{cccc}
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
3 & 3 & 3 & 3 \\
4 & 4 & 4 & 4 \\
20 & 20 & 20 & 20 \\
30 & 30 & 30 & 30 \\
1 & 1 & 1 & 1 \\
400 & 400 & 400 & 400 \\
\end{array}
\]

Behaviour is preserved only by positive linear transformation of Eval
Hence Eval should be proportional to the expected payoff

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