CHAPTER 6

GAME PLAYING
Games of imperfect information

Games of chance

Resource limits and approximate evaluation

$\alpha$-pruning

$\beta$-pruning

Minimax decisions

Perfect play

Games

Outline
Pruning to allow deeper search (McCarthy, 1956)

Machine learning to improve evaluation accuracy (Samuel, 1952-57)

First chess program (Turing, 1951)

Shannon, 1950)

Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948)

Algorithm for perfect play (Zermelo, 1912; von Neumann, 1944)

Computer considers possible lines of play (Babbage, 1846)

Plan of attack:

Time limits unlikely to find goal, must approximate

Specifying a move for every possible opponent reply

"Unpredictable" opponent solution is a strategy

**Games vs. search problems**
### Types of Games

<table>
<thead>
<tr>
<th>Deterministic</th>
<th>Imperfect Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuclear war, bridge, poker, scrabble</td>
<td>Battleships, blind tic-tac-toe, battleships, backgammon, go, othello, chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers</td>
</tr>
<tr>
<td>Monopoly, backgammon</td>
<td>Chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers, chess, checkers</td>
</tr>
<tr>
<td>Chance</td>
<td>Perfect Information</td>
</tr>
</tbody>
</table>

**Chapter 6**
Game tree (2-player, deterministic, turns)
E.g., 2-play game:

\[
\begin{array}{c}
\text{MAX} \\
A_1 \\
A_2 \\
A_3 \\
\text{MIN}
\end{array}
\]

Idea: Choose move to position with highest minimax value

\[
\begin{array}{c}
\text{MAX} \\
A_1 \\
A_2 \\
A_3 \\
\text{MIN}
\end{array}
\]

Perfect play for deterministic, perfect-information games
return $v$

for $a_i$ in successors(state) do $a_i \rightarrow \min(a_i, \text{MAX-VALUE}(s))$

$\infty \rightarrow a$

if terminal-state(state) then return utility(state)

function MIN-VALUE(state) returns a utility value

return $v$

for $a_i$ in successors(state) do $a_i \rightarrow \max(a_i, \text{MIN-VALUE}(s))$

$\infty \rightarrow a$

if terminal-state(state) then return utility(state)

function MAX-VALUE(state) returns a utility value

return the $a$ in ACTIONS(state) maximizing MIN-VALUE(REsult(a, state))

inputs: state, current state in game

function MIN-MAX-DECISION(state) returns an action

function MIN-MAX algorithm
Properties of minimax

Optimal?

Only if tree is finite (chess has specific rules for this).

NB a finite strategy can exist even in an infinite tree!
Properties of minimax

- Time complexity?
  - Yes, against an optimal opponent. Otherwise?
  - Optimality?
  - Yes, if tree is finite (chess has specific rules for this)
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? ??
But do we need to explore every path?

exact solution completely infeasible

For chess, \( q \approx 35, \) \( m \approx 100 \)

space complexity \( O(m^q) \) (depth-first exploration)

time complexity \( O(m^q) \)

Optimal? Yes, against an optimal opponent. Otherwise?

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

Properties of minimax
Pruning Example
Pruning example

α →
$\alpha - \beta$ pruning example
\( \alpha - \beta \)

**Example of Pruning Example**
Why is \( \alpha \) called \( \alpha \)?

Define \( \beta \) similarly for \( \text{MIN} \).

If \( \alpha \) is worse than \( \alpha \), \( \text{MAX} \) will avoid it. Prune that branch.

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

\begin{align*}
\text{MIN} \\
\text{MAX} \\
\vdots \\
\vdots \\
\text{MIN} \\
\text{MAX}
\end{align*}
same as MAX-VALUE but with roles of α, β reversed

function MIN-VALUE(state, α, β) returns a utility value

α ← −∞ for a, s in SUCCESSORS(state) do
if (a, v) MAX(v) → a
if v < α then return (v, a)
α ← Max(α, v)

if TERMINAL-TEST(state) then return UTILITY(state)

if G, the value of the best alternative for MIN along the path to state
G, the value of the best alternative for MAX along the path to state
α, the value of the best alternative for MAX along the path to state
β, the value of the best alternative for MIN along the path to state
input: state, current state in game

function MAX-VALUE(state, α, β) returns a utility value

return the α in ACTIONS(state) maximizing MIN-VALUE(PRESENT(state, α, β))

function ALPHA-BETA-DECISION(state) returns an action

The α–β algorithm
Unfortunately, $\exists \delta_0$ is still impossible.

relevant (a form of meteorasoning)

A simple example of the value of reasoning about which computations are

doubles solvable depth

$(z/w)O = O$ (z/w)

With „perfect ordering“, time complexity

Good move ordering improves effectiveness of pruning

Pruning does not affect final result

Properties of $\alpha$–$\beta$
Suppose we have 100 seconds, explore 10^4 nodes/second, evaluate 10^6 nodes per move, and a search depth of 8 or more. A pretty good chess program would be one that evaluates functions that estimate desirability of position, e.g., prize seeking and quiescence search.

- Use Eval instead of Utility
- Use Cutoff-Test instead of Terminal-Test

**Resource limits**
Black to move
White slightly better

White to move
Black winning

Evolution Functions
payoff in deterministic games acts as an ordinal utility function.

Only the order matters:

Behaviour is preserved under any monotonic transformation of \( EVA \).

**Diagnosis:** Exact values don't matter.
suggest plausible moves.
bad. In 80, q < 300, so most programs use pattern knowledge bases
too. human champions refuse to compete against computers, who are too

Othello: human champions refuse to compete against computers, who are

some lines of search up to 40 ply.
uses very sophisticated evaluation, and undisclosed methods for extending
8-game match in 1997. Deep Blue searches 200 million positions per second,
Chess: Deep Blue defeated human world champion Gary Kasparov in a six-
positions.
positions involving 8 or fewer pieces on the board, a total of 443, 748, 401, 247
Tinsley in 1994. Used an endgame database defining perfect play for all
Chess: Chinook ended 40-year reign of human world champion
Marion

Deterministic Games in Practice
Non-deterministic Games: Backgammon
Simplified example with coin-flipping:

In nondeterministic games, chance introduced by dice, card-shuffling

Non-deterministic games in general
... return average of \( \text{ExpectimaxValue}(\text{state}) \)

if state is a chance node then

return the lowest \( \text{ExpectimaxValue}(\text{state}) \)

if state is a MIN node then

return the highest \( \text{ExpectimaxValue}(\text{state}) \)

if state is a MAX node then

... 

Just like \( \text{Minimax} \), except we must also handle chance nodes:

\( \text{ExpectimaxValue} \) gives perfect play.

Algorithm for nondeterministic games.
world-champion level \approx
TD-GAMMON uses depth-2 search + very good EVAL
& pruning is much less effective
val of lookahead is diminished
As depth increases, probability of reaching a given node shrinks
\[
\text{depth} \ 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9
\]
Backgammon \approx 20 \text{ legal moves} (can be 6,000 with 1-T roll)
Dice rolls increase by 21 possible rolls with 2 dice
Hence $\textsc{Eval}$ should be proportional to the expected payoff.

Behaviour is preserved only by positive linear transformation of $\textsc{Eval}$.

**Diagram:** Exact values do matter.
Games of Imperfect Information

2) Picking the action that wins most tricks on average
1) Generating 100 deals consistent with bidding information

GIB, current best bridge program, approximates this idea by

* Special case: if an action is optimal for all deals, it’s optimal.

then choose the action with highest expected value over all deals

Idea: compute the minimax value of each action in each deal,

* Seems just like having one big dice roll at the beginning of the game

Typically we can calculate a probability for each possible deal

E.g., card games, where opponent’s initial cards are unknown
Four-card bridge/whist/hearts hand, N/AX to play first

Example
Example
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.
take the right fork and you'll find a mound of jewels.

take the left fork and you'll be run over by a bus:

Road B leads to a fork:
Road A leads to a small heap of gold pieces

take the right fork and you'll be run over by a bus.

take the left fork and you'll find a mound of jewels:

Road B leads to a fork:
Road A leads to a small heap of gold pieces

Commonsense example
Guess incorrectly and you'll be run over by a bus.
Guess correctly and you'll find a mound of jewels.

Road B leads to a fork:
Road A leads to a small heap of gold pieces

Road B leads to a fork:
Road A leads to a small heap of gold pieces

Road B leads to a fork:
Road A leads to a small heap of gold pieces

Road B leads to a fork:
Road A leads to a small heap of gold pieces

Commonsense example
Proper analysis

Acting randomly to minimize information disclosure

Signaling to one’s partner

Acting to obtain information

Leads to rational behaviors such as

Can generate and search a tree of information states

information state or belief state the agent is in

With partial observability, value of an action depends on the

Wrong

Intuition that the value of an action is the average of its values

in all actual states is Wrong
Games are to AI as Grand Prix racing is to automobile design.

Optimal decisions depend on information state, not real state.

Uncertainty constrains the assignment of values to states.

A good idea to think about what to think about.

Perfection is unattainable, must approximate.

They illustrate several important points about AI.

**Summary**