Game playing

Chapter 6

Outline

♦ Games
♦ Perfect play
  – minimax decisions
  – α/β 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

deterministic  |  chance
--- | ---
perfect information  |  chess, checkers, go, othello  |  backgammon
imperfect information  |  battleships, blind tic-tac-toe  |  monopoly

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:
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?? 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?? $O(bm)$ (depth-first exploration)
For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games
⇒ exact solution completely infeasible
But do we need to explore every path?
\( \alpha - \beta \) pruning example

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
3 & 12 & 8 \\
\end{align*}
\]

\( \alpha - \beta \) pruning example

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
3 & 12 & 8 \\
\end{align*}
\]

\( \alpha - \beta \) pruning example

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
3 & 12 & 8 \\
\end{align*}
\]

\( \alpha - \beta \) pruning example

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
3 & 12 & 8 \\
\end{align*}
\]

\( \alpha - \beta \) pruning example

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
3 & 12 & 8 \\
\end{align*}
\]

Why is it called \( \alpha - \beta \)?

\[
\begin{align*}
\text{MAX} & \\
\text{MIN} & \\
\end{align*}
\]

\( \alpha \) is the best value (to 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-DECISION(state) returns an action
return the $a$ in ACTIONS(state) maximizing MIN-VALUE(Result(a, state))

function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
inputs: state, current state in game
$\alpha$, the value of the best alternative for $\max$ along the path to state
$\beta$, the value of the best alternative for $\min$ along the path to state
if TERMINAL-TEST(state) then return UTILITY(state)
$v \leftarrow -\infty$
for $a$, $s$ in SUCCESSORS(state) do
$v \leftarrow \max(v, \min-VALUE(s, \alpha, \beta))$
if $v \geq \beta$ then return $v$
$\alpha \leftarrow \max(\alpha, v)$
return $v$

function MIN-VALUE(state, $\alpha$, $\beta$) returns a utility value
same as 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 $= O(b^{m/2})$
$doubles$ 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
$= 10^8$ nodes per move $\approx 35^{5/2}$
$\alpha-\beta$ reaches depth 8 $\Rightarrow$ pretty good chess program

Evaluation functions

Black to move
White slightly better

White to move
Black winning

For chess, typically linear weighted sum of features
$Eval(s) = w_1f_1(s) + w_2f_2(s) + \ldots + w_nf_n(s)$
e.g., $w_1 = 9$ with $f_1(s) = (number$ of white queens) – (number of black queens), 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

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:

\[
\begin{align*}
\text{MAX} & \quad 3 \\
\text{CHANCE} & \quad 0.5 \quad 0.5 \\
\text{MIN} & \quad 2 \quad 4 \quad 7 \quad 4 \quad 0 \quad -2
\end{align*}
\]

Algorithm for nondeterministic games

**EXPECTIMINMAX** gives perfect play

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

\[
\begin{align*}
\text{if} \; \text{state} \; \text{is a MAX node then} \\
\text{return} \; \text{the highest EXPECTIMINMAX-VALUE of Successors(state)} \\
\text{if} \; \text{state} \; \text{is a MIN node then} \\
\text{return} \; \text{the lowest EXPECTIMINMAX-VALUE of Successors(state)} \\
\text{if} \; \text{state} \; \text{is a chance node then} \\
\text{return} \; \text{average of EXPECTIMINMAX-VALUE of Successors(state)} \\
\end{align*}
\]

Nondeterministic games in general

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

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

Nondeterministic games in practice

Dice rolls increase \( 6 \): 21 possible rolls with 2 dice

Backgammon \( \approx 20 \) legal moves (can be 6,000 with 1-1 roll)

\[\text{depth} \, \text{4} = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9\]

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**

\[
\begin{align*}
\text{DICE} & \quad 2.1 \quad 0.9 \\
\text{MIN} & \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 200
\end{align*}
\]

Behaviour is preserved only by positive linear transformation of **Eval**

Hence **Eval** should be proportional to the expected payoff
Example

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

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.

Example

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

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.

Example

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

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.

Example

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

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.
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