Game playing

CHAPTER 5, SECTIONS 1–5

Outline

◊ Perfect play
◊ Resource limits
◊ $\alpha-\beta$ pruning
◊ Games of chance
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

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>perfect info</td>
<td>chess, checkers, go, othello</td>
<td>backgammon monopoly</td>
</tr>
<tr>
<td>imperfect info</td>
<td></td>
<td>bridge, poker, scrabble nuclear war</td>
</tr>
</tbody>
</table>
**Minimax**

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest \textit{minimax} value

= best achievable payoff against best play

\textbf{E.g., 2-ply game:}

\textbf{MAX}

\begin{align*}
&\text{MIN} \\
&\text{MAX} \\
&3 \\
&12 \quad 8 \\
&3 \\
&8 \quad 6 \\
&4 \quad 6 \quad 14 \\
&14 \\
&5 \quad 2 \\
&2
\end{align*}

**Minimax algorithm**

```plaintext
function Minimax-Decision(game) returns an operator
for each op in Operators(game) do
    Value[op] = Minimax-Value(Apply(op, game), game)
end
return the op with the highest Value[op]

function Minimax-Value(state, game) returns a utility value
if Terminal-Test(game)(state) then
    return Utility(game)(state)
else if Max in to move in state then
    return the highest Minimax-Value of Successors(state)
else
    return the lowest Minimax-Value of Successors(state)
```

---

(Images of diagrams and text passages as shown.)
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
Nondeterministic games

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

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

```
MAX

CHANCE

MIN
```

Algorithm for nondeterministic games

**Expectiminimax** gives perfect play

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

```
\ldots
\text{if state is a chance node then}
\quad \text{return average of } \text{EXPECTIMINIMAX-VALUE of SUCCESSORS(state)}
\ldots
```

A version of \(\alpha-\beta\) pruning is possible
but only if the leaf values are bounded. Why??
**Nondeterministic games in practice**

Dice rolls increase. 21 possible rolls with 2 dice.
Backgammon ≈ 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
⇒ value of lookahead is diminished

\(\alpha=\beta\) pruning is much less effective

TDGAMMON uses depth-2 search + very good EVAL
⇒ world-champion level

---

**Digression: Exact values DO matter**

```
\begin{align*}
\text{MAX} & \\
\text{DCE} & 2.1 & 1.3 & 21 & 40.9 \\
\text{MIN} & 2 & 3 & 4 & 20 & 30 & 400 \\
\end{align*}
```

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