Foundations of Artificial Intelligence
CSE 573 — Fall 2001
Game Playing, Continued

Henry Kautz

Slide CSE573–1

Design Issues of Heuristic Minimax

Search: search to a constant depth

Problems:

Slide CSE573–2
Design Issues of Heuristic Minimax

- Some portions of the game tree may be “hotter” than others. Should search to quiescence. Continue along a path as long as one move’s static value stands out (indicating a likely capture).
- Horizon effect
- Secondary search. (*singular extension heuristic*)

Slide CSE573–3

Improving Minimax — $\alpha - \beta$ pruning

Slide CSE573–4
Two More Example

\[ A \]
\[ (\text{attack queen}) \]
\[ B \]
\[ \text{(get mated)} \]
\[ C \]
\[ J \]
\[ K \]
\[-1 \]
\[ \text{?} \]

\[ A \]
\[ (\text{attack queen}) \]
\[ B \]
\[ .03 \]
\[ C \]
\[ D \]
\[ E \]
\[ F \]
\[ .05 \]

Slide CSE573–5

A deep $\alpha - \beta$ cutoff

\[ A \]
\[ B \]
\[ .03 \]
\[ C \]
\[ D \]
\[ E \]
\[ F \]
\[ .1 \]

Slide CSE573–6
If $m$ is better than $n$ for Player, never get to $n$ in play.

Slide CSE573–7

- Add automatic pruning to minimax. Called alpha-beta pruning.
- Idea: Note minimax expands in depth-first manner. Let $\alpha$ be best choice we’ve found so far at any choice point along the path for MAX, and $\beta$ best for MIN (i.e., lowest value). Alpha-beta search updates $\alpha$ and $\beta$ during search, and prunes subtree as soon as it’s worse than the current $\alpha$ or $\beta$.

Slide CSE573–8
\( \alpha - \beta \) Search

\( c = \) search cutoff
\( \alpha = \) lower bound on Max's outcome; initially set to \(-\infty\)
\( \beta = \) upper bound on Min's outcome; initially set to \(+\infty\)

We'll call \( \alpha - \beta \) procedure recursively with a narrowing range between \( \alpha \) and \( \beta \).

Maximizing levels may reset \( \alpha \) to a higher value; Minimizing levels may reset \( \beta \) to a lower value.

---

function \textsc{Max-Value}(state, game, \( \alpha \), \( \beta \)) returns the minimax value of state

\textbf{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

\textbf{if} \textsc{Cutoff-Test}(state) \textbf{then return} \textsc{Eval}(state)

\textbf{for each} \( s \) in \textsc{Successors}(state) \textbf{do}

\( \alpha = \textsc{Max-Value}(s, \text{game}, \alpha, \beta) \)

\textbf{if} \( \alpha \geq \beta \) \textbf{then return} \( \beta \)

\textbf{end}

return \( \alpha \)

---

function \textsc{Min-Value}(state, game, \( \alpha \), \( \beta \)) returns the minimax value of state

\textbf{if} \textsc{Cutoff-Test}(state) \textbf{then return} \textsc{Eval}(state)

\textbf{for each} \( s \) in \textsc{Successors}(state) \textbf{do}

\( \beta = \textsc{Min-Value}(s, \text{game}, \alpha, \beta) \)

\textbf{if} \( \beta \leq \alpha \) \textbf{then return} \( \alpha \)

\textbf{end}

return \( \beta \)
Search Space Size Reductions
1. Reduction in search space depends on ordering nodes!
2. Optimistic analysis: branching only $\sqrt{b}$ — for chess, 6 instead of 35. Why?

3. $b^{d/2}$: Can go twice as deep! Tremendous difference in practice.

Including chance
- Read Section 5.5, R&N.
- E.g. Backgammon.
- expectiminimax.
State of the Art in Checkers

- 1952: Samuel developed a checkers program that learned its own evaluation function through self play.
- 1992: Chinook (J. Schaeffer) wins the U.S. Open. At the world championship, Marion Tinsley beat Chinook

State of the Art in Backgammon

- 1980: BKG using one-ply search and lots of luck defeated the human world champion.
- 1992: Tesauro combines Samuel’s learning method with neural networks to develop a new evaluation function, resulting in a program ranked among the top 3 players in the world.
State of the Art in Go

$2,000,000 prize available for first computer program to defeat a top level player.

Slide CSE573–15

History of Chess in AI

<table>
<thead>
<tr>
<th>Year</th>
<th>Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Legal chess</td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>Occasional player</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>World-ranked</td>
<td></td>
</tr>
<tr>
<td>2900</td>
<td>Gary Kasparov</td>
<td></td>
</tr>
</tbody>
</table>

Early 1950’s Shannon and Turing both had programs that (barely) played legal chess (500 rank).

1950’s Alex Bernstein’s system, (500+ε).

1957 Herb Simon claims that a computer chess program would be world chess champion in 10 years...yeah, right.

Slide CSE573–16


1968 McCarthy, Michie, Papert bet Levy (rated 2325) that a computer program would beat him within 10 years.


1973 By 1973...Slate: “It had become too painful even to look at Chess 3.6 any more, let alone work on it.”

1973 Chess 4.0: smart plausible-move generator rather than speeding up the search. Improved rapidly when put on faster machines.

1976 Chess 4.5: ranking of 2070.


1980’s Programs depend on search speed rather than knowledge (2300 range).

1993 DEEP THOUGHT: Sophisticated special-purpose computer; $\alpha - \beta$ search; searches 10 ply; singular extensions; rated about 2600.

1995 DEEP BLUE: searches 14-ply; considers 100–200 billion positions per move.