Games

Deep Blue beats Gary Kasparov - 1997
(3 wins, 1 loss, 2 draws)
- Deep Blue: 32 RISC processors + 256 VLSI chess engines
- 200 million positions per second, 16 plies

Deep Blue: 32 RISC processors + 256 VLSI chess engines

Saying Deep Blue doesn’t really think about chess is like saying an airplane doesn’t really fly because it doesn’t flap its wings. — Drew McDermott

Today

Game tree search (40 min)
- Minimax
- Alpha-Beta Pruning
- Games of chance (30 min)

Tonight

- Game tree search (40 min)
- Group exercise: Reversi (50 min)
- Reversi Tournament (20 min)
- Games of chance (30 min)

Games in AI

- In AI, “games” usually refers to deterministic, turn-taking, two-player, zero-sum games of perfect information
  - Deterministic: next state of environment is completely determined by current state and action executed by the agent (not probabilistic)
  - Turn-taking: 2 agents whose actions must alternate
  - Zero-sum games: if one agent wins, the other loses
  - Perfect information: fully observable

Other Games

<table>
<thead>
<tr>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect info</td>
<td>chess, checkers, go, othello</td>
</tr>
<tr>
<td></td>
<td>backgammon, monopoly</td>
</tr>
</tbody>
</table>

Imperfect info

| Stratego   | Bridge, poker, scrabble, nuclear war |

Games as Search

- States: board configurations
- Initial state: the board position and which player will move
- Successor function: returns list of (move, state) pairs, each indicating a legal move and the resulting state
- Terminal test: determines when the game is over
- Utility function: gives a numeric value in terminal states (e.g., -1, 0, +1 for loss, tie, win)
Intuition

Mini-Max

© Patrick Winston
Mini-Max Properties

- Complete? Yes, if tree is finite
- Optimal?
  - Against an optimal opponent? Yes
  - Otherwise? No: Does at least as well, but may not exploit opponent weakness
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$

Good Enough?

- Chess:
  - branching factor $b = 35$
  - game length $m = 100$
  - search space $b^m = 35^{100} \approx 10^{154}$
- The Universe:
  - number of atoms $\approx 10^{78}$
  - age $\approx 10^{18}$ seconds
  - $10^8$ moves/sec $\times 10^{78} \times 10^{18} = 10^{104}$
Do we need to check this node?
**Alpha-Beta**

\[
\text{MinVal(state, alpha, beta)}\
\text{if (terminal(state))}\
\quad \text{return utility(state);}\
\text{for (s in children(state))}{\}\
\quad \text{child = MinVal(s,alpha,beta);}\
\quad \beta = \text{min}(\beta, child);\
\quad \text{if (alpha} \geq \beta) \text{return child;}\
\text{return beta;}\]

\[
\alpha = \text{the highest value for MAX along the path}\
\beta = \text{the lowest value for MIN along the path}\
\]

**Alpha-Beta**

\[
\text{MaxVal(state, alpha, beta)}\
\text{if (terminal(state))}\
\quad \text{return utility(state);}\
\text{for (s in children(state))}{\}\
\quad \text{child = MinVal(s,alpha,beta);}\
\quad \alpha = \text{max}(\alpha, child);\
\quad \text{if (alpha} \geq \beta) \text{return child;}\
\text{return beta;}\]

\[
\alpha = \text{the highest value for MAX along the path}\
\beta = \text{the lowest value for MIN along the path}\
\]
α - the best value for max along the path
β - the best value for min along the path

α = -∞, β = -29
α = -29, β = ∞
α = -∞, β = ∞
α = -29, β = -37

β < α
prune!
Good Enough?

Chess:
- branching factor $b \approx 35$
- game length $m \approx 100$
- search space $b^m \approx 35^{100} \approx 10^{77}$

The Universe:
- number of atoms $\approx 10^{78}$
- age $\approx 10^{18}$ seconds
- $10^8$ moves/sec $\times 10^{18} \times 10^{18} = 10^{104}$

Alpha-Beta Properties

- Still guaranteed to find the best move
- Best case time complexity: $O(bm/2)$
- Can double the depth of search!
- Best case when best moves are tried first
- Good static evaluation function helps!
- But still too slow for chess...

Partial Space Search

- Strategies:
  - search to a fixed depth
  - iterative deepening (most common)
  - ignore ‘quiescent’ nodes
  - Static Evaluation Function assigns a score to a non-terminal state
**Evaluation Functions**

- **Reversi**
  - Number squares held?
  - Better: number of squares held that cannot be flipped
  - Prefer valuable squares
    - NxN array \( w[i,j] \) of position values
    - Highest value: corners, edges
    - Lowest value: next to corner or edge
    - \( s[i,j] = +1 \) player, 0 empty, -1 opponent
  
  \[
  \text{score} = \sum_{i,j} w[i,j] s[i,j]
  \]

- **Chess**:
  - \( \text{eval}(s) = w_1 * \text{material}(s) + w_2 * \text{mobility}(s) + w_3 * \text{king safety}(s) + w_4 * \text{center control}(s) + \ldots \)
  - In practice MiniMax improves accuracy of heuristic eval function
  - But one can construct pathological games where more search hurts performance!
    (Nau 1981)

**End-Game Databases**

- Ken Thompson - all 5 piece end-games
- Lewis Stiller - all 6 piece end-games
  - Refuted common chess wisdom: many positions thought to be ties were really forced wins -- 90\% for white
  - Is perfect chess a win for white?

**The MONSTER**

White wins in 255 moves
(Stiller, 1991)

**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 (!)
- **Chess**:
- **Reversi**:
  - Human champions refuse to play against computers because software is too good

<table>
<thead>
<tr>
<th></th>
<th>Chess</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of board</td>
<td>8 x 8</td>
<td>19 x 19</td>
</tr>
<tr>
<td>Average no. of moves per game</td>
<td>100</td>
<td>300</td>
</tr>
<tr>
<td>Average branching factor per turn</td>
<td>35</td>
<td>235</td>
</tr>
<tr>
<td>Additional complexity</td>
<td>Players can pass</td>
<td></td>
</tr>
</tbody>
</table>
Nondeterministic Games

- Involve chance: dice, shuffling, etc.
- Chance nodes: calculate the expected value
- E.g.: weighted average over all possible dice rolls

In Practice...

- Chance adds dramatically to size of search space
  - Backgammon: number of distinct possible rolls of dice is 21
  - Branching factor $b$ is usually around 20, but can be as high as 4000 (dice rolls that are doubles)
- Alpha-beta pruning is generally less effective
- Best Backgammon programs use other methods

Imperfect Information

- E.g. card games, where opponents' initial cards are unknown
- Idea: For all deals consistent with what you can see
  - compute the minimax value of available actions for each of possible deals
  - compute the expected value over all deals

Probabilistic STRIPS Planning

domain: Hungry Monkey

shake: if (ontable)
  Prob(2/3) -> +1 banana  
  Prob(1/3) -> no change  
  else  
    Prob(1/6) -> +1 banana  
    Prob(5/6) -> no change  

jump: if (~ontable)
  Prob(2/3) -> ontable
  Prob(1/3) -> ~ontable
  else ontable

What is the expected reward?

[1] shake
[2] jump; shake
[3] jump; shake; shake;
[4] jump; if (~ontable){ jump; shake} else { shake; shake }

ExpectiMax

$$ \text{ExpectiMax}(n) = \begin{cases} U(n) & \text{if } n \text{ is a terminal node} \\ \max \{ \text{ExpectiMax}(s) | s \in \text{children}(n) \} & \text{if } n \text{ is max node} \\ \sum_{s \in \text{children}(n)} P(s) \cdot \text{ExpectiMax}(s) & \text{if } n \text{ is a chance node} \end{cases} $$
The result of the ExpectiMax analysis is a conditional plan (also called a policy):
- Optimal plan for 2 steps: jump; shake
- Optimal plan for 3 steps:
  - jump; if (ontable) {shake; shake}
  - else {jump; shake}
Probabilistic planning can be generalized in many ways, including action costs and hidden state.
The general problem is that of solving a Markov Decision Process (MDP).
## Summary

- **Deterministic games**
  - Minimax search
  - Alpha-Beta pruning
  - Static evaluation functions
- **Games of chance**
  - Expected value
  - Probabilistic planning
- **Strategic games with large branching factors (Go)**
  - Relatively little progress