Adversarial Search

CSE 473
University of Washington

Contents

• Board Games
• Minimax Search
• Alpha-Beta Search
• Games with an Element of Chance

Games Overview

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect information</td>
<td>chess, checkers, go, othello</td>
<td>backgammon, monopoly</td>
</tr>
<tr>
<td>Imperfect information</td>
<td></td>
<td>bridge, poker, scrabble</td>
</tr>
</tbody>
</table>

Games & Game Theory

• When there is more than one agent, the future is not anymore easily predictable for the agent

• In competitive environments (conflicting goals), adversarial search becomes necessary

• In AI, we usually consider special type of games:

  board games, which can be characterized as deterministic, turn-taking, two-player, zero-sum games with perfect information
Games as Search

- Components:
  - 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 (eg, -1, 0, +1 in chess for loss, tie, win)

Conventions:
- First player is MAX, 2nd player is MIN
- State utility values from MAX's perspective
- Initial state and legal moves define the game tree
Properties of minimax

- Complete?
  - Yes (if tree is finite)

- Optimal?
  - Yes (against an optimal opponent)

- Time complexity?
  - $O(b^m)$

- Space complexity?
  - $O(bm)$ (depth-first exploration)

Good enough?

- Chess:
  - branching factor $b \approx 35$
  - game length $m \approx 100$
  - search space $b^m \approx 35^{100} \approx 10^{154}$

- The Universe:
  - number of atoms $\approx 10^{78}$
  - age $\approx 10^{21}$ milliseconds

Alpha-Beta Pruning
No - this branch is guaranteed to be worse than what max already has.
Alpha-Beta

\texttt{MaxVal(state, alpha, beta)}{
  if (terminal(state)) return utility(state);
  for (s in successors(state)){
    child = MinVal(s, alpha, beta);
    alpha = max(alpha, child);
    if (alpha>=beta) return child;
  }
  return alpha;
}

\texttt{MinVal(state, alpha, beta)}{
  if (terminal(state)) return utility(state);
  for (s in successors(state)){
    child = MaxVal(s, alpha, beta);
    beta = min(beta, child);
    if (alpha>=beta) return child;
  }
  return beta;
}

\(\alpha\) - the best value for max along the path
\(\beta\) - the best value for min along the path

\(\text{MaxVal}\) is a function that evaluates the maximin value of a game tree.

\(\text{MinVal}\) is a function that evaluates the minimax value of a game tree.

\(\alpha\) is the highest (best) value for MAX along the path.
\(\beta\) is the lowest (best) value for MIN along the path.

Pruning occurs when \(\beta < \alpha\).
**Alpha-Beta**

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

alpha = the highest (best) value for MAX along path
beta = the lowest (best) value for MIN along path

**Properties of α-β**

- Still optimal, pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = \(O(b^{m/2})\) \(\rightarrow\) doubles depth of search
- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

**Good enough?**

- **Chess:**
  - branching factor \(b \approx 35\)
  - game length \(m \approx 100\)
  - search space \(b^{m/2} \approx 35^{50} \approx 10^{77}\)
- **The Universe:**
  - number of atoms \(\approx 10^{78}\)
  - age \(\approx 10^{21}\) milliseconds

**Partial State Spaces**

- **Strategies:**
  - search to a fixed depth
  - iterative deepening (most common)
  - stop only at 'quiescent' nodes
### Cutting off search

- Does it work in practice?
  - \( b^m = 10^6, b=35 \Rightarrow m=4 \)

- 4-ply lookahead is a **hopeless chess player**!
  - 4-ply \( \approx \) human novice
  - 8-ply \( \approx \) typical PC, human master
  - 12-ply \( \approx \) Deep Blue, Kasparov

### Evaluation Function

- When search space is too large, create game tree up to a certain depth only.
- Art is to evaluate positions that are not terminal states.
- Example of simple evaluation criteria in chess:
  - Material worth: pawn=1, knight =3, rook=5, queen=9,
  - Other: king safety, good pawn structure
  - Rule of thumb: 3-point advantage = certain victory

\[
\text{eval}(s) = w_1 \times \text{material}(s) + w_2 \times \text{mobility}(s) + w_3 \times \text{king safety}(s) + w_4 \times \text{center control}(s) + \ldots
\]

### Transposition Tables

- Game trees contain repeated states
- In chess, e.g., the game tree may have \( 35^{100} \) nodes, but there are only \( 10^{40} \) different board positions
- Similar to closed list in search, maintain a transposition table
  - Got its name from the fact that the same state is reached by a transposition of moves.
Game Playing in Practice

- **Checkers**: Solved! It has been shown that there is no strategy to beat the computer. The best you can get is a draw.
- **Chess**: Deep Blue defeated human world champion Gary Kasparov in a 6 game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
- **Othello**: Human champions refuse to play against computers because software is too good.
- **Go**: Human champions refuse to play against computers because software is too bad.

Summary of Deterministic Games

- Basic idea: minimax -- too slow for most games
- Alpha-Beta pruning can increase max depth by factor up to 2
- Limited depth search may be necessary
- Static evaluation functions necessary for limited depth search and help alpha-beta
- Opening game and End game databases can help
- Computers can beat humans in some games (checkers, chess, othello) but not in others (Go)

Other Games

<table>
<thead>
<tr>
<th></th>
<th>deterministic</th>
<th>chance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perfect</strong></td>
<td>chess, checkers, go, othello</td>
<td>backgammon, monopoly</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Imperfect</strong></td>
<td></td>
<td>bridge, poker, scrabble</td>
</tr>
</tbody>
</table>

Games that Include an Element of Chance

White has just rolled 6-5 and has 4 legal moves.
Game Tree for Games with an Element of Chance

- In addition to MIN- and MAX nodes, we need chance nodes (e.g., for rolling dice).
- Search costs increase: Instead of $O(b^d)$, we get $O((bn)^d)$, where $n$ is the number of chance outcomes.

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