Adversarial search

Game search

- Game-playing programs developed by AI researchers since the beginning of the modern AI era
  - Programs playing chess, checkers, etc (1950s)

- **Specifics of the game search:**
  - Sequences of player’s decisions **we control**
  - Decisions of other player(s) **we do not control**

- **Contingency problem:** many possible opponent’s moves must be “covered” by the solution
  - Opponent’s behavior introduces an uncertainty in to the game
  - We do not know exactly what the response is going to be

- **Rational opponent** – maximizes its own **utility (payoff)** function
Types of game problems

- Types of game problems:
  - Adversarial games:
    - win of one player is a loss of the other
  - Cooperative games:
    - players have common interests and utility function
  - A spectrum of game problems in between the two:

Adversarial games

Fully cooperative games

we focus on adversarial games only!!

Example of an adversarial 2 person game: Tic-tac-toe

Player 1 (x) moves

Player 2 (o) moves

Player 1 (x) moves

Loss Draw Win
Game search problem

- **Game problem formulation:**
  - **Initial state:** initial board position + info whose move it is
  - **Operators:** legal moves a player can make
  - **Goal (terminal test):** determines when the game is over
  - **Utility (payoff) function:** measures the outcome of the game and its desirability

- **Search objective:**
  - find the sequence of player’s decisions (moves) maximizing its utility (payoff)
  
**Caveat:** Consider the opponent’s moves and their utility

---

Game problem formulation (Tic-tac-toe)

**Objectives:**
- **Player 1:** maximize outcome
- **Player 2:** minimize outcome

**Operators**

**Initial state**

**Terminal (goal) states**

**Utility:**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Minimax algorithm

How to deal with the contingency problem?
• Assuming that the opponent is rational and always optimizes its behavior (opposite to us) we consider the best opponent’s response
• Then the minimax algorithm determines the best move

Minimax algorithm. Example
Minimax algorithm. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5
Minimax algorithm. Example
Minimax algorithm. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

M. Hauskrecht
Minimax algorithm. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

MINIMAX

2 9 3

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

Minimax algorithm. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

M. Hauskrecht
Minimax algorithm

```
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 is to move in state then
        return the highest MINIMAX-VALUE of SUCCESSORS(state)
    else
        return the lowest MINIMAX-VALUE of SUCCESSORS(state)
```
Complexity of the minimax algorithm

• We need to explore the complete game tree before making the decision

\[
\text{Complexity: } b^m
\]

• Impossible for large games
  – Chess: 35 operators, game can have 50 or more moves
Solution to the complexity problem

Two solutions:
1. **Dynamic pruning of redundant branches** of the search tree
   - identify a provably suboptimal branch of the search tree before it is fully explored
   - Eliminate the suboptimal branch
   **Procedure:** Alpha-Beta pruning

2. **Early cutoff of the search tree**
   - uses imperfect minimax value estimate of non-terminal states (positions)

---

**Alpha beta pruning**

- Some branches will never be played by rational players since they include sub-optimal decisions (for either player)
Alpha beta pruning. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

M. Hauskrecht

Alpha beta pruning. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

$\geq 4$

M. Hauskrecht
Alpha beta pruning. Example

MAX

MIN

MAX

\[
\begin{array}{c}
4 \\
3 \\
6 \\
2 \\
2 \\
1 \\
9 \\
5 \\
3 \\
1 \\
5 \\
7 \\
5 \\
\end{array}
\]

M. Hauskrecht
Alpha beta pruning. Example
Alpha beta pruning. Example
Alpha beta pruning. Example

MAX

MIN

MAX

M. Hauskrecht
**Alpha beta pruning. Example**

nodes that were never explored !!!

---

**Alpha-Beta pruning**

```plaintext
function Max-Value(state, game, α, β) returns the minimax value of state
inputs: state, current state in game
        game, game description
        α, the best score for MAX along the path to state
        β, the best score for MIN along the path to state
if GOAL-Test(state) then return EVAL(state)
for each y in Successors(state) do
    α ← Max(α, Min-Value(y, game, α, β))
    if α ≥ β then return β
end
return α

function Min-Value(state, game, α, β) returns the minimax value of state
if GOAL-Test(state) then return EVAL(state)
for each y in Successors(state) do
    β ← Min(β, Max-Values(y, game, α, β))
    if β ≤ α then return α
end
return β
```

M. Hauskrecht
Using minimax value estimates

- **Idea:**
  - Cutoff the search tree before the terminal state is reached
  - Use imperfect estimate of the minimax value at the leaves

  - **Heuristic evaluation function**

```
MAX

MIN

Heuristic evaluation function

Cutoff level
```

Design of evaluation functions

- **Heuristic estimate** of the value for a sub-tree
- **Examples of a heuristic functions:**
  - **Material advantage in chess, checkers**
    - Gives a value to every piece on the board, its position and combines them
  - More general **feature-based evaluation function**
    - Typically a linear evaluation function:
      \[ f(s) = f_1(s)w_1 + f_2(s)w_2 + \ldots + f_k(s)w_k \]
      \[ f_i(s) \quad - \text{a feature of a state } s \]
      \[ w_i \quad - \text{feature weight} \]
Further extensions to real games

- Restricted set of moves to be considered under the cutoff level to reduce branching and improve the evaluation function
  - E.g., consider only the capture moves in chess

Heuristic estimates