GAME DEVELOPMENT

CSC 3150/9010

Instructor: Dr. Edward Kim
Adversarial AI - Games

• What does it mean to look X moves ahead in games?

• E.g. Deep Blue looks 10 moves ahead in Chess…
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
Minimax algorithm. Example
Minimax algorithm. Example
Minimax algorithm. Example
Minimax algorithm. Example
Minimax algorithm. Example
Minimax algorithm. Example
Minimax algorithm. Example
Minimax \((s)\)

If \(s\) is terminal
Return \(U(s)\)

If next move is \(A\)
Return \(\max_{s' \in \text{Succs}(s)} \text{Minimax}(s')\)

Else
Return \(\min_{s' \in \text{Succs}(s)} \text{Minimax}(s')\)
Properties of minimax

Complete??
Properties of minimax

**Complete?** Yes, if tree is finite (chess has specific rules for this)

**Optimal?** Yes, against an optimal opponent. Otherwise?

**Time complexity?**
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?**
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

But do we need to explore every path?
Cutoff search
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
- Evaluation function

```
MAX

MIN

Heuristic evaluation function

4 4 2 2 9 3 5 5 7

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:
      \[ f(s) = f_1(s)w_1 + f_2(s)w_2 + \ldots + f_k(s)w_k \]
      
      \[ f_i(s) \] - a feature of a state \( s \)
      \[ w_i \] - feature weight
Chess example

- Pawn – 1
- Rook – 5
- Bishop – 3
- Knight – 3
For chess, typically *linear* weighted sum of features

\[ \text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) \]

E.g., \( w_1 = 9 \) with
\[ f_1(s) = (\text{number of white queens}) - (\text{number of black queens}), \text{ etc.} \]
X has 6 possible win paths:

$E(n) = 6 - 5 = 1$

O has 5 possible wins:

$X$ has 4 possible win paths;
O has 6 possible wins

$E(n) = 4 - 6 = -2$

$X$ has 5 possible win paths;
O has 4 possible wins

$E(n) = 5 - 4 = 1$

Heuristic is $E(n) = M(n) - O(n)$

where $M(n)$ is the total of My possible winning lines

$O(n)$ is total of Opponent's possible winning lines

$E(n)$ is the total Evaluation for state $n$

**Figure 4.16** Heuristic measuring conflict applied to states of tic-tac-toe.