

For Monday

- Read chapter 7, sections 1-4
- Homework
 - Chapter 6, exercise 1

Program 1

- Any questions?

Game Playing Problem

- Instance of **general search problem**
- States where game has ended are **terminal states**
- A **utility function** (or **payoff function**) determines the value of the terminal states
- In 2 player games, MAX tries to maximize the payoff and MIN is tries to minimize the payoff
- In the search tree, the first layer is a move by MAX and the next a move by MIN, etc.
- Each layer is called a **ply**

Minimax Algorithm

- Method for determining the optimal move
- Generate the entire search tree
- Compute the utility of each node moving upward in the tree as follows:
 - At each MAX node, pick the move with maximum utility
 - At each MIN node, pick the move with minimum utility (assume opponent plays optimally)
 - At the root, the optimal move is determined

Recursive 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 highest Minimax-Value of Successors(*state*)

else

return lowest Minimax-Value of Successors(*state*)

Making Imperfect Decisions

- Generating the complete game tree is intractable for most games
- Alternative:
 - Cut off search
 - Apply some heuristic evaluation function to determine the quality of the nodes at the cutoff

Evaluation Functions

- Evaluation function needs to
 - Agree with the utility function on terminal states
 - Be quick to evaluate
 - Accurately reflect chances of winning
- Example: **material value** of chess pieces
- Evaluation functions are usually **weighted linear functions**

Cutting Off Search

- Search to uniform depth
- Use iterative deepening to search as deep as time allows (**anytime algorithm**)
- Issues
 - **quiescence** needed
 - horizon problem

Alpha-Beta Pruning

- Concept: Avoid looking at subtrees that won't affect the outcome
- Once a subtree is known to be worse than the current best option, don't consider it further

General Principle

- If a node has value n , but the player considering moving to that node has a better choice either at the node's parent or at some higher node in the tree, that node will never be chosen.
- Keep track of MAX's best choice (α) and MIN's best choice (β) and prune any subtree as soon as it is known to be worse than the current α or β value

```
function Max-Value (state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
  if Cutoff-Test(state) then return Eval(state)
  for each s in Successors(state) do
     $\alpha$  <- Max( $\alpha$ , Min-Value(s , game,  $\alpha$ ,  $\beta$ ))
    if  $\alpha$   $\geq$   $\beta$  then return  $\beta$ 
  end
  return  $\alpha$ 
```

```
function Min-Value(state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
  if Cutoff-Test(state) then return Eval(state)
  for each s in Successors(state) do
     $\beta$  <- Min( $\beta$ ,Max-Value(s , game,  $\alpha$ ,  $\beta$ ))
    if  $\beta$   $\leq$   $\alpha$  then return  $\alpha$ 
  end
  return  $\beta$ 
```

Effectiveness

- Depends on the order in which siblings are considered
- Optimal ordering would reduce nodes considered from $O(b^d)$ to $O(b^{d/2})$ --but that requires perfect knowledge
- Simple ordering heuristics can help quite a bit

Chance

- What if we don't know what the options are?
- Expectiminimax uses the **expected value** for any node where chance is involved.
- Pruning with chance is more difficult.
Why?

Imperfect Knowledge

- What issues arise when we don't know everything (as in standard card games)?

State of the Art

- Chess – Deep Blue and Fritz
- Checkers – Chinook
- Othello – Logistello
- Backgammon – TD-Gammon (learning)
- Go – Computers are very bad
- Bridge

What about the games we play?