Foundations of Artificial Intelligence

CSE 573 — Fall 2001
Game Playing

Henry Kautz

Slide CSE573–1

Game Playing – Why?

Slide CSE573–2

Game Playing

An AI Favorite

- structured task
- not initially thought to require large amounts of knowledge
- focus on games of perfect information

Game Playing

Initial State

Operators

Terminal Test

Utility Function

Game Playing

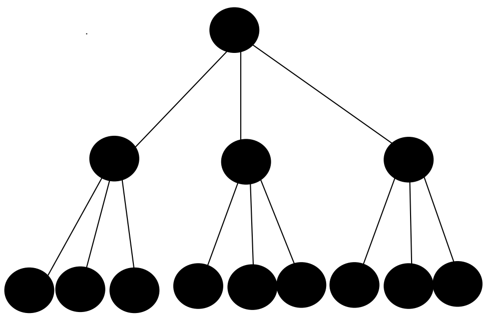
Initial State is the initial board/position

Operators define the set of legal moves from any position

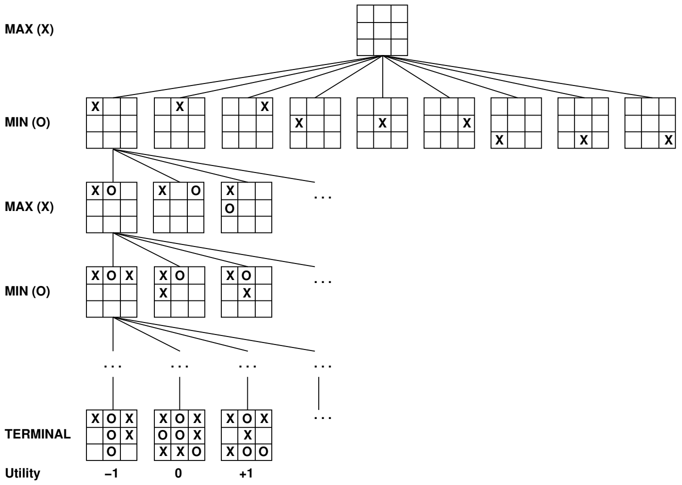
Terminal Test determines when the game is over

Utility Function gives a numeric outcome for the game

Game Playing as Search

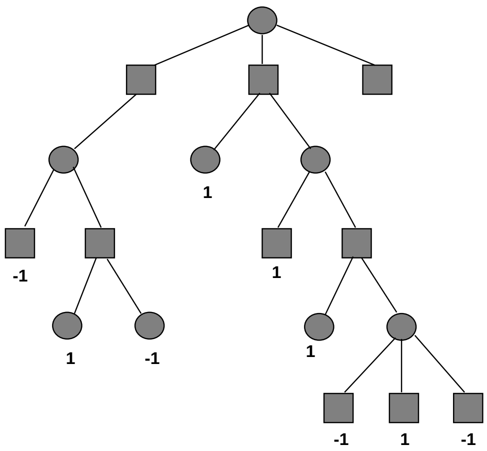


Partial Search Tree for Tic-Tac-Toe



Slide CSE573–7

Simple Minimax



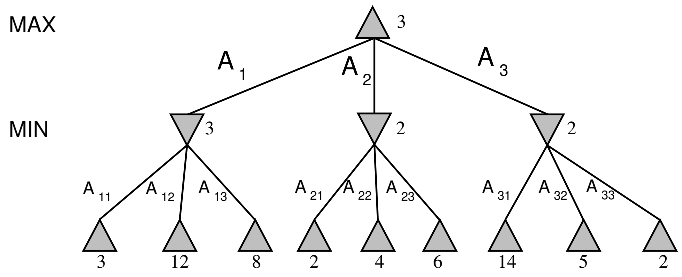
Slide CSE573–8

Simplified Minimax Algorithm

1. Expand the entire tree below the root.
2. Evaluate the terminal nodes as wins for the minimizer or maximizer.
3. Select an unlabeled node, n, all of whose children have been assigned values. If there is no such node, we're done — return the value assigned to the root.
4. If n is a minimizer move, assign it a value that is the minimum of the values of its children. If n is a maximizer move, assign it a value that is the maximum of the values of its children. Return to Step 3.

Slide CSE573–9

Another Example



Slide CSE573–10

1. In game tree search, a *move* is a pair of actions. One player's action is a *ply*. 2-ply = one move.
2. Called a **minimax** decision because it maximizes the utility under the assumption that the opponent will play perfectly to minimize it.
3. Time complexity: $O(b^m)$ (m plies and b branching.) Impractical for e.g. chess ($b \approx 30$ to 40).

The Need for Imperfect Decisions

Problem: Minimax assumes the program has time to search to the terminal nodes.

Solution:

The Need for Imperfect Decisions

Problem: Minimax assumes the program has time to search to the terminal nodes.

Solution: Cut off search earlier and apply a **heuristic evaluation function** to the leaves.

Static Evaluation Functions

Minimax depends on the translation of board quality into a single, summarizing number. Can be difficult. Expensive.

- Add up values of pieces each player has (weighted by importance of piece). E.g. intro chess: pawn = 1pnt; knight or bishop = 3pnts; rook = 5 pnts; and queen = 9pnts.
- Isolated pawns are bad. How well protected is your king? How much maneuverability to you have? Do you control the center of the board?
- How many plies with perfect eval function?

Design Issues of Heuristic Minimax

Evaluation Function: What features should we evaluate and how should we use them? An evaluation function should:

1.
2.
3.

Linear Evaluation Functions

- $w_1 f_1 + w_2 f_2 + ... + w_n f_n$
- w — weight; f — feature
- Steps in designing an evaluation function:
  1. Pick informative features
  2. Tune the weights

  Deep Blue: precision in eval (normalized, beteen 0 — 1) is 10^{-3} to 10^{-4} ! (lots of fine-tuning is important)

Creating Evaluation Functions

- Features / weights for chess?
- How tune weights?
- How find features?

Slide CSE573–17

Design Issues of Heuristic Minimax

Search: search to a constant depth

Problems:

Slide CSE573–18

Design Issues of Heuristic Minimax

- Some portions of the game tree may be "hotter" than others. Should search to *quiescence*. Continue along a path as long as one move's static value stands out (indicating a likely capture).
- *Horizon effect*
- Secondary search. (*singular extension heuristic)*