Foundations of Artificial Intelligence

CSE 573 — Fall 2001

Game Playing, Continued

Henry Kautz

Design Issues of Heuristic Minimax

Search: search to a constant depth

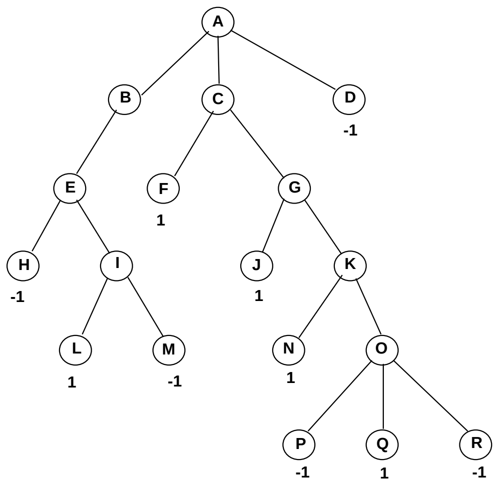
Problems:

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)*

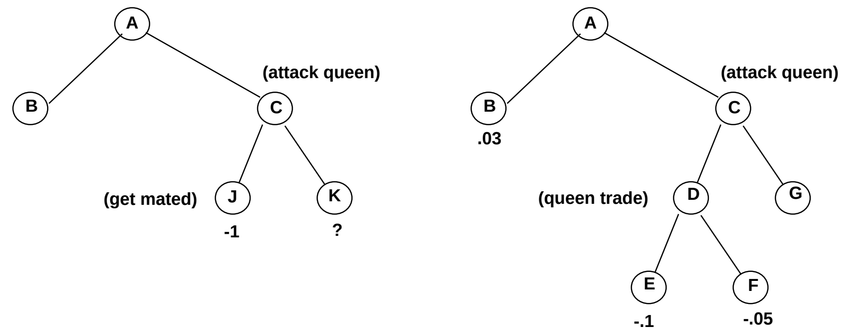
Slide CSE573–3

Improving Minimax — $\alpha - \beta$ pruning



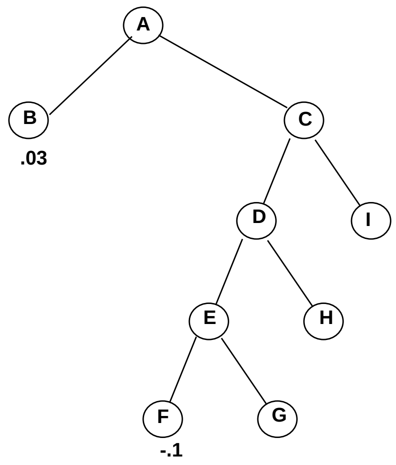
Slide CSE573–4

Two More Example

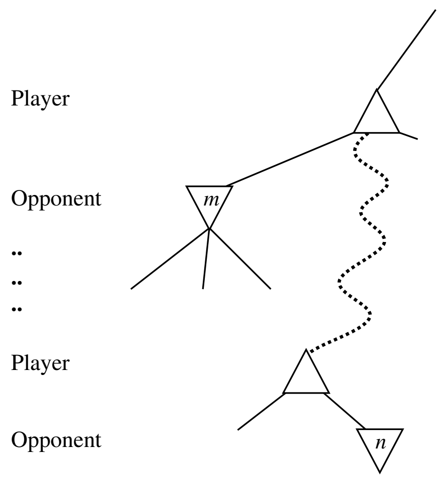


Slide CSE573–5

A deep $\alpha - \beta$ cutoff



Slide CSE573–6



If m is better than n for Player, never get to n in play.

- Add automatic pruning to minimax. Called alpha-beta pruning.
- Idea: Note minimax expands in depth-first manner. Let α be best choice we've found so far at any choice point along the path for MAX, and β best for MIN (i.e., lowest value). Alpha-beta search updates α and β during search, and prunes subtree as soon as it's worse than the current α or β.

$\alpha - \beta$ Search

c = search cutoff
α = lower bound on Max's outcome; initially set to $-\infty$
β = upper bound on Min's outcome ; initially set to $+\infty$

We'll call $\alpha - \beta$ procedure recursively with a narrowing range between α and β.

Maximizing levels may reset α to a higher value; Minimizing levels may reset β to a lower value.

Slide CSE573–9

```
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 CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     α ← MAX(α, MIN-VALUE(s, 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 CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     β ← MIN(β, MAX-VALUE(s, game, α, β))
     if β ≤ α then return α
  end
  return β
```

Slide CSE573–10

Search Space Size Reductions

1. Reduction in search space depends on ordering nodes!
2. Optimistic analysis: branching only $\sqrt{b}$ — for chess, 6 instead of 35. Why?

3. $b^{d/2}$: Can go twice as deep! Tremendous difference in practice.

Slide CSE573–11

Including chance

- Read Section 5.5, R&N.
- E.g. Backgammon.
- **expectiminimax.**

Slide CSE573–12

State of the Art in Checkers

- 1952: Samuel developed a checkers program that learned its own evaluation function through self play.
- 1992: *Chinook* (J. Schaeffer) wins the U.S. Open. At the world championship, Marion Tinsley beat *Chinook*

State of the Art in Backgammon

- 1980: *BKG* using one-ply search and lots of luck defeated the human world champion.
- 1992: Tesauro combines Samuel's learning method with neural networks to develop a new evaluation function, resulting in a program ranked among the top 3 players in the world.

State of the Art in Go

$2,000,000 prize available for first computer program to defeat a top level player.

History of Chess in AI

500	legal chess
1200	occasional player
2000	world-ranked
2900	Gary Kasparov

Early 1950's Shannon and Turing both had programs that (barely) played legal chess (500 rank).

1950's Alex Bernstein's system, ($500+\epsilon$).

1957 Herb Simon claims that a computer chess program would be world chess champion in 10 years...yeah, right.

1966 McCarthy arranges chess match, Stanford vs. Russia. Long, drawn-out match. Russia wins.

1967 Richard Greenblatt, MIT. First of the modern chess programs, *MacHack* (1100 ranking).

1968 McCarthy, Michie, Papert bet Levy (rated 2325) that a computer program would beat him within 10 years.

1970 ACM started running chess tournaments. Chess 3.0-6 (rated 1400).

1973 By 1973...Slate: "It had become too painful even to look at Chess 3.6 any more, let alone work on it."

1973 Chess 4.0: smart plausible-move generator rather than

Slide CSE573–17

speeding up the search. Improved rapidly when put on faster machines.

1976 Chess 4.5: ranking of 2070.

1977 Chess 4.5 vs. Levy. Levy wins.

1980's Programs depend on search speed rather than knowledge (2300 range).

1993 DEEP THOUGHT: Sophisticated special-purpose computer; $\alpha - \beta$ search; searches 10 ply; singular extensions; rated about 2600.

1995 DEEP BLUE: searches 14-ply; considers 100–200 billion positions per move.

Slide CSE573–18