1 handout: slides

*You think you know when you can learn, are more sure when you can write, even more when you can teach, but certain when you can program.*
Adversarial Search

Problems
Different!
Minimax
Tic-tac-toe
Improvements
Break
$\alpha$-$\beta$ Pruning
$\alpha$-$\beta$ Pseudo-code
Why $\alpha$-$\beta$?
Progress
EOLQs
**Observability:** complete, partial, hidden
**State:** discrete, continuous
**Actions:** deterministic, stochastic, discrete, continuous
**Nature:** static, deterministic, stochastic
**Interaction:** one decision, sequential
**Time:** static/off-line, on-line, discrete, continuous
**Percepts:** discrete, continuous, uncertain
**Others:** solo, cooperative, competitive
Multi-agent is Different

- Shortest-path (M&C, vacuum, tile puzzle)
  - want least-cost path to goal at unknown depth
- Decisions with an adversary (chess, tic-tac-toe)
  - adversary might prevent path to best goal
  - want best assured outcome
Adversarial Search: Minimax

Each ply corresponds to half a move. Terminal states are labeled with value. Can also bound depth and use a static evaluation function on non-terminal states.
A 3-length is a complete row, column, or diagonal. The value of a position is:

- $\infty$ if a win for me,
- $-\infty$ if a win for you,
- otherwise $= \# 3$-lengths open for me $- \# 3$-lengths open for you
Tic-tac-toe: two-ply search

Adversarial Search
- Problems
- Different!
- Minimax

Tic-tac-toe
- Improvements
- Break
- $\alpha-\beta$ Pruning
- $\alpha-\beta$ Pseudo-code
- Why $\alpha-\beta$?
- Progress
- EOLQs

Fig. 3.8 Minimax applied to tic-tac-toe (stage 1).
Fig. 3.9 Minimax applied to tic-tac-toe (stage 2).
Adversarial Search
- Problems
- Different!
- Minimax

Tic-tac-toe
- Improvements
- Break
- $\alpha$-$\beta$ Pruning
- $\alpha$-$\beta$ Pseudo-code
- Why $\alpha$-$\beta$?
- Progress
- EOLQs

Fig. 3.10 Minimax applied to tic-tac-toe (stage 3).
Improving the Search

- partial expansion, SEF
- symmetry (‘transposition tables’)
- search more ply as we have time (De Groot figure)
- avoid unnecessary evaluations
Break

- asst 3
- asst 4
- projects: physical TSP!
- fyi: SAT, Barbie
Which Values are Necessary?

Adversarial Search

- Problems
- Different!
- Minimax
- Tic-tac-toe
- Improvements

Break

- $\alpha$-$\beta$ Pruning
- $\alpha$-$\beta$ Pseudo-code
- Why $\alpha$-$\beta$?
- Progress
- EOLQs
**α-β Pruning**

- **α** best outcome Max can force at previous decision on this path (init to \(-\infty\))
- **β** best outcome Min can force at previous decision on this path (init to \(\infty\))

**α** and **β** values are copied down the tree (but not up). Minmax values are passed up the tree, as usual.
Max-value (state, $\alpha$, $\beta$):
when depth-cutoff (state), return SEF(state)
for each child of state
    $\alpha \leftarrow \max(\alpha, \text{Min-value (child, } \alpha, \beta))$
    when $\alpha \geq \beta$, return $\alpha$
return $\alpha$

Min-value (state, $\alpha$, $\beta$):
when depth-cutoff (state), return SEF(state)
for each child of state
    $\beta \leftarrow \min(\beta, \text{Max-value (child, } \alpha, \beta))$
    when $\beta \leq \alpha$, return $\beta$
return $\beta$
α-β in action

Problems

Different!

Minimax

Tic-tac-toe

Improvements

Break

Progress

Why α-β?

α-β Pruning

α-β Pruning

Pseudo-code

α-β

α-β

α-β

Why α-β?

Progress

EOLQs

α-β

α-β

α-β

Start Node

Start Node

(bad is 1)

(bad is 1)

MAX Nodes

MAX Nodes

WIN Nodes

WIN Nodes

Fig. 3.12. An example illustrating the alpha-beta search procedure.
Time complexity of $\alpha$-$\beta$ is about $O(b^{d/2})$
Computers best: chess, checkers, backgammon, Scrabble, Jeopardy!
Computers competitive: bridge, crosswords, poker, small Go
Computers amateur: full Go
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

*Thanks!*