Lecture 9 of 42

Game Tree Search:
Minimax and Alpha-Beta (α - β) Pruning

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

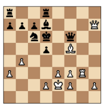
Reading for Next Class:
Section 7.1 – 7.4, p. 194 - 210, Russell & Norvig 2^{nd} edition
Outside reference:
University of Alberta GAMES page – http://www.cs.ualberta.ca/~games/

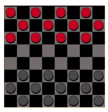
Lecture Outline

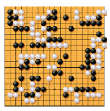
- Reading for Next Class: 7.1 – 7.4 (p. 194 – 210), R&N 2^e
- Last Class, 5.4-5.5, p. 151-158; Games Intro, 6.1-6.3, p. 161-174
  - Third CSP algorithm: constraint propagation by arc consistency (AC-3)
  - "One-step" vs. "all-steps" lookahead
- Today: Game Tree Search
  - Rudiments of game theory
  - Minimax with alpha-beta (α - β) pruning
  - Perfect information *vs.* imperfect information
- Need for Expectiminimax
  - Games of chance: dealing with nondeterminism
  - Imperfect information
- Game Analysis
  - Quiescence
  - Horizon effect
  - "Averaging over clairvoyance" and when/why it fails
- Next Class: From Search to Knowledge Representation

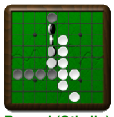
Types of Games: Review

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleship, blind tictactoe	bridge, poker, scrabble nuclear war


Chess
http://tr.im/zdTD


Checkers
http://tr.im/zdTW

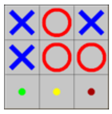

Go
http://tr.im/zdVn

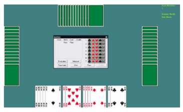

Reversi (Othello)
http://tr.im/zdVr


Backgammon
http://tr.im/ze1P


Monopoly © Parker Brothers
http://tr.im/ze2F


Battleship © Milton Bradley
http://tr.im/zdWK


Tic-Tac-Toe
http://tr.im/zdXB


Contract Bridge
http://tr.im/ze5D


Poker (Texas Hold 'Em)
http://tr.im/ze7W


Scrabble © Hasbro
http://tr.im/ze90

Alpha-Beta (α - β) Pruning – Example: Review

What are α, β values here?

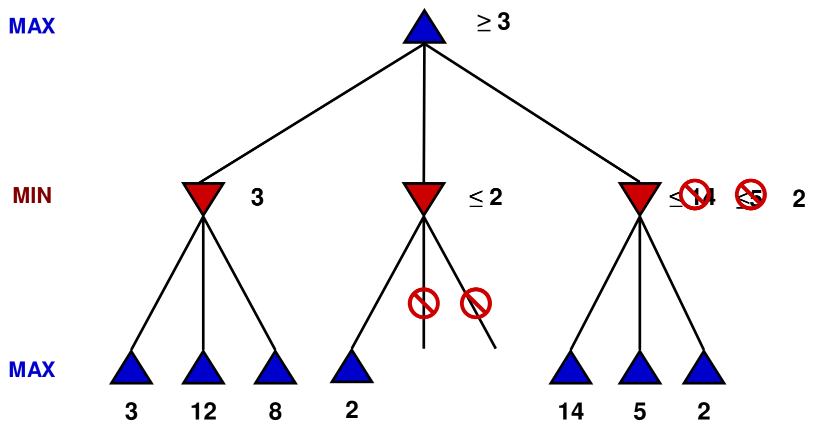


Figure 6.5 p. 168 R&N 2e

Minimax with α-β Pruning – Algorithm: Review

function Alpha-Beta-Decision(*state*) **returns** an action
  return the *a* in Actions(*state*) maximizing Min-Value(Result(*a*, *state*))

---

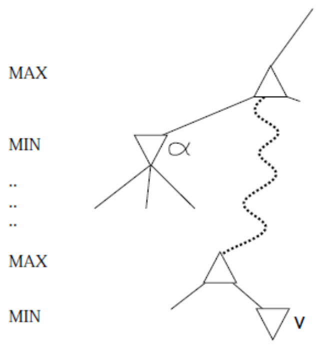
function Max-Value(*state*, α, β) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

  if Terminal-Test(*state*) **then return** Utility(*state*)
  $v \leftarrow -\infty$
  for *a*, *s* in Successors(*state*) **do**
    $v \leftarrow$ Max(v, Min-Value(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ Max(α, v)
  return v

---

function Min-Value(*state*, α, β) **returns** *a utility value*
  same as Max-Value but with roles of α, β reversed

Why Is It Called α-β?



α is the best value (to MAX) found so far off the current path

If V is worse than α, MAX will avoid it $\Rightarrow$ prune that branch

Define β similarly for MIN

Figure 6.6 p. 169 R&N 2e

Depth Limit Rationale: Review

Pruning does not affect final result

Good move ordering improves effectiveness of pruning

With "perfect ordering," time complexity $= O(b^{m/2})$
$\Rightarrow$ doubles solvable depth

A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

Unfortunately, 35^{50} is still impossible!

- **Can We Do Better?**
- **Idea: Adapt Resource-Bounded Heuristic Search Techniques**
  - **Depth-limited**
  - **Iterative deepening**
  - **Memory-bounded**

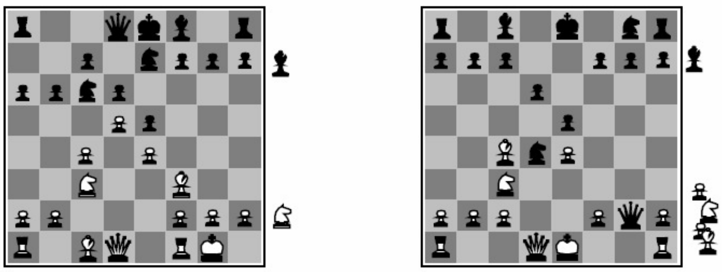
Resource Limits and Limited-Ply Search

Standard approach:

- Use Cutoff-Test instead of Terminal-Test
  e.g., depth limit (perhaps add quiescence search)
- Use Eval instead of Utility
  i.e., evaluation function that estimates desirability of position

Suppose we have 100 seconds, explore 10^4 nodes/second
$\Rightarrow 10^6$ nodes per move $\approx 35^{8/2}$
$\Rightarrow \alpha$–β reaches depth 8 $\Rightarrow$ pretty good chess program

Static Evaluation Functions: Review



Black to move

White slightly better

White to move

Black winning

For chess, typically linear weighted sum of features

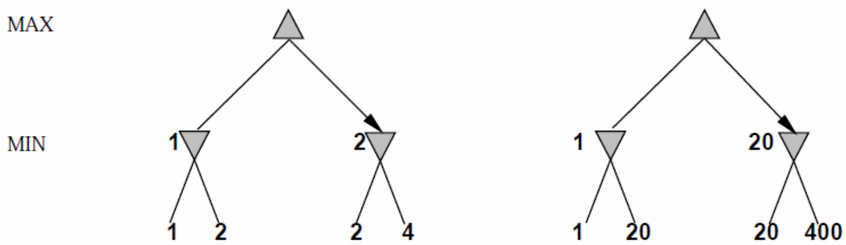
$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

e.g., $w_1 = 9$ with

$f_1(s) =$ (number of white queens) – (number of black queens), etc.

Figure 6.8 p. 173 R&N 2e

Digression: Exact Values Don't Matter



Behaviour is preserved under any monotonic transformation of Eval

Only the order matters:

payoff in deterministic games acts as an ordinal utility function

Quiescence and the Horizon Effect

- **Issues**
  - Quiescence
    - Play has "settled down"
    - Evaluation function unlikely to exhibit wild swings in value in near future
  - Horizon effect
    - "Stalling for time"
    - Postpones inevitable win or damaging move by opponent
    - See: Figure 6.9, p. 175 R&N 2e
- **Solutions?**
  - Quiescence search: expand non-quiescent positions further
  - No general solution to horizon problem at present

Deterministic Games in Practice

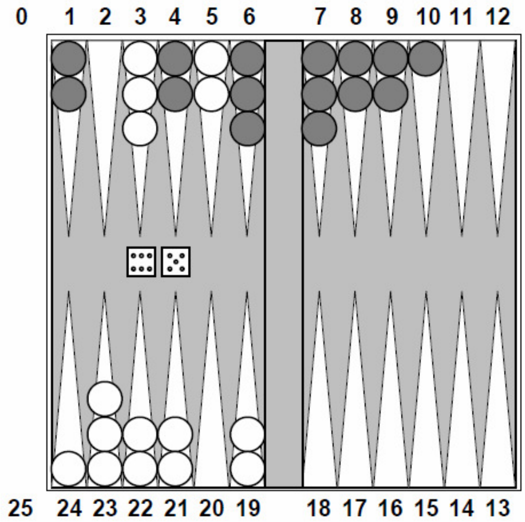
Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

Othello: human champions refuse to compete against computers, who are too good.

Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

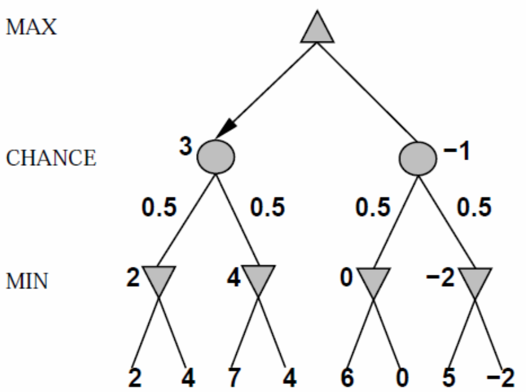
Nondeterministic Games: Backgammon



Nondeterministic Games in General

In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:



EXPECTIMINIMAX: ALGORITHM FOR NONDETERMINISTIC GAMES

EXPECTIMINIMAX gives perfect play

Just like MINIMAX, except we must also handle chance nodes:

```
...
if state is a MAX node then
    return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a MIN node then
    return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a chance node then
    return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
...
```


NONDETERMINISTIC GAMES IN PRACTICE

Dice rolls increase b: 21 possible rolls with 2 dice
Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)

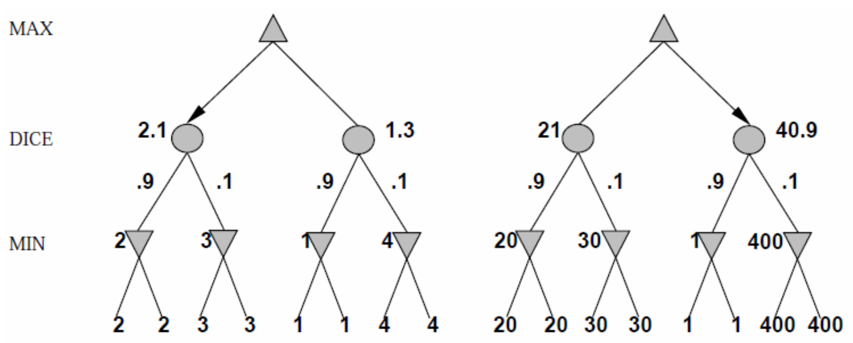
$$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

As depth increases, probability of reaching a given node shrinks
$\Rightarrow$ value of lookahead is diminished

α–β pruning is much less effective

TDGAMMON uses depth-2 search + very good EVAL
$\approx$ world-champion level

Digression: Exact Values Do Matter



Behaviour is preserved only by positive linear transformation of EVAL

Hence EVAL should be proportional to the expected payoff

Figure 6.12 p. 178 R&N 2e

Games of Imperfect Information [1]: Solution Approach

E.g., card games, where opponent's initial cards are unknown

Typically we can calculate a probability for each possible deal

Seems just like having one big dice roll at the beginning of the game*

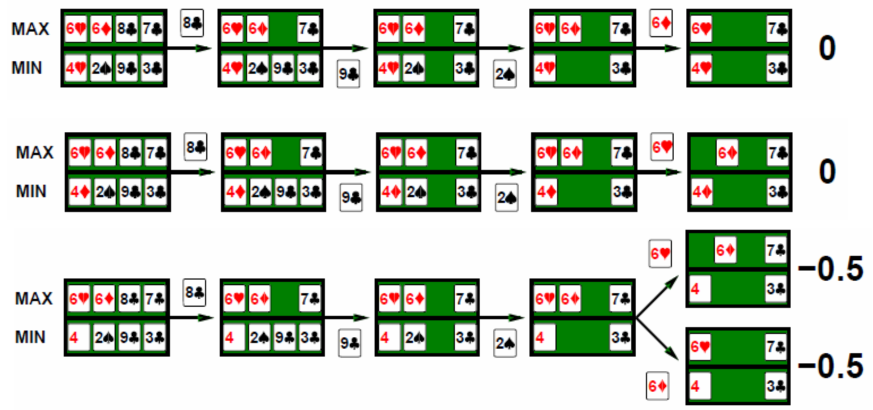
Idea: compute the minimax value of each action in each deal,
then choose the action with highest expected value over all deals*

Special case: if an action is optimal for all deals, it's optimal.*

GIB, current best bridge program, approximates this idea by
1) generating 100 deals consistent with bidding information
2) picking the action that wins most tricks on average

Games of Imperfect Information [2]: Example

Four-card bridge/whist/hearts hand, MAX to play first



Similar to example on p. 179 R&N 2e

Commonsense Example [1]: Statement

Day 1

Road A leads to a small heap of gold pieces
Road B leads to a fork:
take the left fork and you'll find a mound of jewels;
take the right fork and you'll be run over by a bus.

Day 2

Road A leads to a small heap of gold pieces
Road B leads to a fork:
take the left fork and you'll be run over by a bus;
take the right fork and you'll find a mound of jewels.

Day 3

Road A leads to a small heap of gold pieces
Road B leads to a fork:
guess correctly and you'll find a mound of jewels;
guess incorrectly and you'll be run over by a bus.

Commonsense Example [2]: Proper Analysis

* Intuition that the value of an action is the average of its values in all actual states is WRONG

With partial observability, value of an action depends on the information state or belief state the agent is in

Can generate and search a tree of information states

Leads to rational behaviors such as
- ◇ Acting to obtain information
- ◇ Signalling to one's partner
- ◇ Acting randomly to minimize information disclosure

Games: Summary

Games are fun to work on! (and dangerous)

They illustrate several important points about AI

- ◇ perfection is unattainable ⇒ must approximate
- ◇ good idea to think about what to think about
- ◇ uncertainty constrains the assignment of values to states
- ◇ optimal decisions depend on information state, not real state

Games are to AI as grand prix racing is to automobile design

Terminology

- Game Tree Search
  - Zero-sum games
  - 2-player vs. *n*-player
- Minimax Algorithm: Alternates between MAX and MIN Players
- Alpha-Beta Pruning (α - β Pruning)
  - α: best value to MAX found so far *off* current path (*v* worse than $\alpha \Rightarrow$ prune)
  - β: best value to MIN found so far *off* current path
- Resource-Bounded Minimax
  - Static evaluation function
  - Limited-ply search (compare: depth-limited search *aka* DLS)
  - Iterative deepening search (compare: ID-DFS, IDA*)
- Expectiminimax
  - Based on expectation
  - Games with chance
  - Games with imperfect information

Summary Points

- Game Theory Continued
  - Game tree representation
  - Perfect play: Minimax algorithm, speedup with alpha-beta (α - β) pruning
  - Resource-bounded Minimax: static evaluation functions, iterative deepening
  - Emphasis: two-player (with exceptions), zero-sum, perfect info
- Alpha-Beta Pruning (α - β Pruning)
- Resource-Bounded Minimax
  - Need for static evaluation (compare: heuristics)
  - Limited-ply search (compare: depth-limited search *aka* DLS)
  - Iterative deepening search (compare: ID-DFS, IDA*)
- Expectiminimax
  - Based on expectation
  - Games with chance
  - Games with imperfect information
- Significance of Games to AI
  - Understanding representation, reasoning, and learning
  - Finding out what approximations, refinements, abstractions work