

## Games and adversarial search

### Chapter 6

Sept 17, 2008

COMP 131, Lecture 5

1

## Announcements

- Late submissions policy extended to 5 (five) free days for each student for the entire term
  - Otherwise, late submissions not accepted
  - Indicate in a README file if you'd like to use up one or more of your free days
- A1 due tonight 11:59pm
- A2 out, due Wed, Sept 24<sup>th</sup>

Sept 17, 2008

COMP 131, Lecture 5

2

## Last time – CSPs

- **Constraint satisfaction problems (CSPs)** are a simple form of **representation** allowing **domain-independent** methods
- They consist of **variables** and **constraints** on those variables. A state is an **assignment** of values to variables
- **Backtracking search** is a form of DFS used to solve CSPs
- The **minimum remaining values (MRV)** and **degree** heuristics are good for deciding which variable to choose for assignment next.
- The **least constraining value** heuristic helps ordering variable values to try
- Branching factor can be reduced by **propagating constraints** on partial variable assignments. **Forward checking** is the simplest method: it deletes inconsistent values from neighbors.

Sept 17, 2008

COMP 131, Lecture 5

3

## Last time – CSPs (cont.)

- **Arc consistency** enforcement (AC-3) is more powerful and checks value consistency for all assignments in **directed** arcs between neighbors
- Local search using **min-conflicts** heuristic can solve CSPs successfully by operating on complete (but maybe inconsistent) assignments, randomly **reassigning** variables to values which generate the least conflicts
- CSP solving **complexity** is related to the **structure** of its graph. Tree-structured problems can be solved in linear time. **Cutset-conditioning** can reduce a general CSP to a tree-structured one and is efficient if small cutset can be found

Sept 17, 2008

COMP 131, Lecture 5

4

## Today

- Games
- Perfect (optimal) play:
  - minimax algorithm
  - alpha-beta pruning
- Resource limits
  - imperfect information
  - real-time play
- Games with elements of chance

Sept 17, 2008

COMP 131, Lecture 5

5

## Games vs. search

- 2 or more players (agents) influence outcome
- Opponent is planning against you and is unpredictable
- Solution is a *strategy* (a.k.a. a *contingency plan*)
  - specify a response for every possible opponent move
- Time and resource limits

Sept 17, 2008

COMP 131, Lecture 5

6

## Types of games

	deterministic	chance
perfect information	tic-tac-toe, checkers, chess, go	backgammon, monopoly
imperfect information	battleships	poker, bridge, scrabble, markets, war

Sept 17, 2008

COMP 131, Lecture 5

7

## More types of games

	turn-taking	continuous-time
zero-sum	tic-tac-toe, chess, backgammon, poker, scrabble	soccer, football war?
non-zero-sum	dialogue, paper writing	markets, war

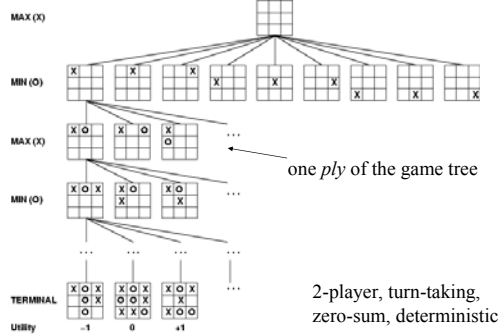
zero-sum:  
winner's payoff = -loser's loss

Sept 17, 2008

COMP 131, Lecture 5

8

## Game tree



Sept 17, 2008

COMP 131, Lecture 5

9

## Optimal strategies

- An *optimal strategy* leads to outcomes at least as good as any other strategy when playing an infallible opponent
- Rational agent chooses optimal strategy
  - given information and resource constraints

Sept 17, 2008

COMP 131, Lecture 5

10

## Minimax strategy

- Idea: best achievable payoff against best opponent
- Assuming both agents play optimally...
- ... what's the value for MAX of being in a node?
  - node is terminal: it's the utility = payoff
  - MAX's turn: will choose so that next state has max. value
  - MIN's turn: will choose so that next state has min. value

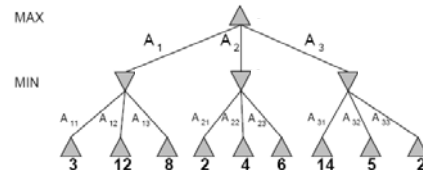
$$\text{MINIMAX-VALUE}(n) = \begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is terminal} \\ \max_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is MAX's} \\ \min_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is MIN's} \end{cases}$$

Sept 17, 2008

COMP 131, Lecture 5

11

## A 2-ply game tree example



Sept 17, 2008

COMP 131, Lecture 5

12

## Minimax algorithm

```

function MINIMAX-DECISION(state) returns an action
  v ← MAX-VALUE(state)
  return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← -∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s))
  return v
    
```

Sept 17, 2008

COMP 131, Lecture 5

13

## Properties of minimax

- Complete? Yes (if tree finite)
- Optimal? Yes (against optimal opponent)
- Time complexity?  $O(b^m)$
- Space complexity?  $O(bm)$  (depth first exploration)
- For chess,  $b \approx 35$ ,  $m \approx 100$  for "reasonable" games  
→ exact solution completely infeasible

Sept 17, 2008

COMP 131, Lecture 5

14

## Can we NOT look at every node?

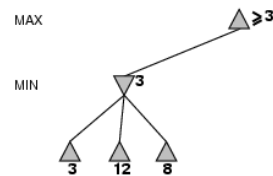
- *Pruning* – eliminating subtrees from consideration
- Alpha beta pruning
  - returns the same move as minimax but prunes branches which cannot influence the decision
  - cuts time in half

Sept 17, 2008

COMP 131, Lecture 5

15

## Alpha-beta pruning example

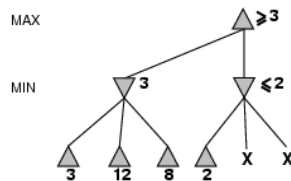


Sept 17, 2008

COMP 131, Lecture 5

16

## Alpha-beta pruning example

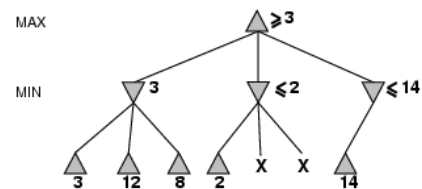


Sept 17, 2008

COMP 131, Lecture 5

17

## Alpha-beta pruning example

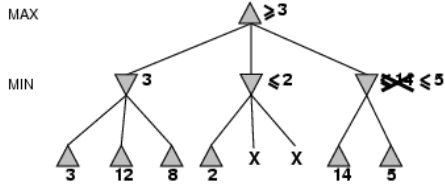


Sept 17, 2008

COMP 131, Lecture 5

18

## Alpha-beta pruning example

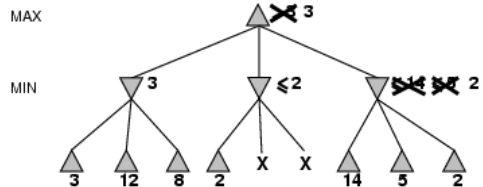


Sept 17, 2008

COMP 131, Lecture 5

19

## Alpha-beta pruning example



Can often prune entire subtrees

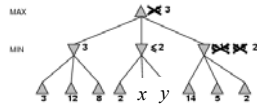
Sept 17, 2008

COMP 131, Lecture 5

20

## Simplification of MINIMAX formula

- Another way to look at alpha-beta
- Let  $x, y$  be the unevaluated children of node  $C$
- Let  $z = \min(x, y)$



$$\begin{aligned} \text{MINIMAX-VALUE}(\text{root}) &= \\ &= \max(\min(3, 12, 8), \min(2, x, y), \min(14, 5, 2)) \\ &= \max(3, \min(2, x, y), 2) \\ &= \max(3, z, 2) \quad \text{where } z \leq 2 \\ &= 3. \end{aligned}$$

Value of root is independent of  $x, y$

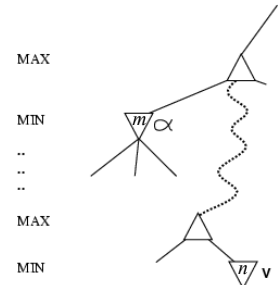
Sept 17, 2008

COMP 131, Lecture 5

21

## Why is it called alpha-beta?

- Idea:
  - if  $n$  is a choice for a player, but  $m$  is a better choice further up, then  $n$  will never be reached
  - so, can prune  $n$
- $\alpha$  – value of the best choice so far for MAX
- $\beta$  – value of the best choice so far for MIN



bounds on the backup values: prune nodes that are worse than that

Sept 17, 2008

COMP 131, Lecture 5

22

## Alpha-beta pruning algorithm

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game
  v ← MAX-VALUE(state, -∞, +∞)
  return the action in SUCCESSORS(state) with value v

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 ← -∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
  α ← MAX(α, v)
  return v
```

Sept 17, 2008

COMP 131, Lecture 5

23

## Alpha-beta pruning algorithm

```
function MIN-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 ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
  β ← MIN(β, v)
  return v
```

Sept 17, 2008

COMP 131, Lecture 5

24

## Properties of alpha-beta pruning

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With “perfect ordering” time complexity =  $O(b^{m/2})$   
→ **doubles** depth of search **35<sup>50</sup> is still impossible**
- A simple example of the value of reasoning about which computations are relevant (a form of *metareasoning*)

Sept 17, 2008

COMP 131, Lecture 5

25

## Hitting resource limits

Real time decisions: no time to get to terminal states

Standard approach: **proposed by Shannon in 1950**

- CUTOFF\_TEST instead of TERMINAL\_TEST  
e.g., depth limit
- *Evaluation* function instead of *utility* function  
= estimated desirability (*expected* utility) of position

Suppose we have 100 secs, explore  $10^4$  nodes/sec

→  $10^6$  nodes per move, or  $\sim 35^4$  – beginner level

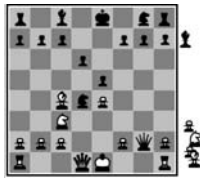
Alpha-beta pruning will go to depth 8 – pretty good chess program!

Sept 17, 2008

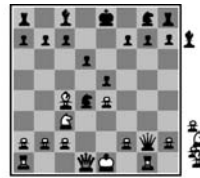
COMP 131, Lecture 5

26

## Evaluation functions



(a) White to move



(b) White to move

- For chess, typically *linear weighted sum of features*  
 $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$
- e.g.,  $w_1 = 9$  with  
 $f_1(s) = (\text{number of white queens}) - (\text{number of black queens})$

Sept 17, 2008

COMP 131, Lecture 5

27

## Good evaluation functions

- Order terminal states same way as utility
- Fast to compute
- Strongly correlated with chance of winning in non-terminal states

**uncertainty comes from computational limitations**

Evaluation functions are not part of the game rules, but come from experience

Sept 17, 2008

COMP 131, Lecture 5

28

## Minimax Cutoff ideas

- Cutoff at depth = D
  - too simple (what if this position is reached right at cutoff?)
- *Quiescence* search
  - a *quiescent* position is unlikely to show wild swings of value in near future
  - apply *Eval* only to quiescent positions
  - expand until quiescent position found



(b) White to move

In chess, for *Eval* that only counts pieces, positions where captures can be made are not quiescent

Sept 17, 2008

COMP 131, Lecture 5

29

## Deterministic games in practice

- **Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.
- **Chess:** Deep Blue defeated human world champion Garry 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.

Sept 17, 2008

COMP 131, Lecture 5

30

## Nondeterministic games

	deterministic	chance
perfect information	tic-tac-toe, checkers, chess, go	backgammon, monopoly
imperfect information	battleships	poker, bridge, scrabble, markets, war

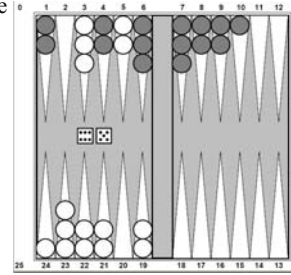
Sept 17, 2008

COMP 131, Lecture 5

31

## Random element

- Games with an element of chance are more like unpredictable real life
- E.g., Backgammon



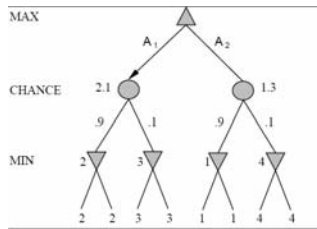
Sept 17, 2008

COMP 131, Lecture 5

32

## Randomness

- Introduced by dice roll, coin flipping, cards dealt, etc.



Sept 17, 2008

COMP 131, Lecture 5

33

## Algorithm for games of chance

- EXPECTIMINIMAX( $n$ ) is just like MINIMAX( $n$ ) except with chance nodes

- if  $state$  is a MAX node, return *highest* EXPECTIMINIMAX-VALUE of  $Successor(state)$
- if  $state$  is a MIN node, return *lowest* EXPECTIMINIMAX-VALUE of  $Successor(state)$
- if  $state$  is a chance node, return *expected value* EXPECTIMINIMAX-VALUE of  $Successor(state)$

Expected value( $x$ ) =  $\sum(\text{value}(x) \cdot \text{probability}(x))$  over all  $x$   
a generalization of averaging

Sept 17, 2008

COMP 131, Lecture 5

34

## Chance games in practice

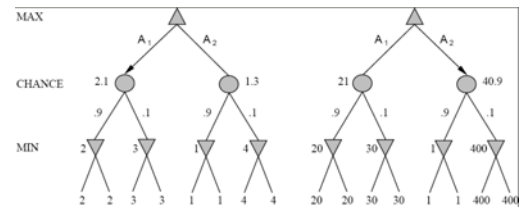
- Dice rolls increase  $b$ : 21 possible rolls with 2 dice
  - backgammon has ~20 legal moves
- Probability of reaching a particular node shrinks with tree depth
  - less value to lookahead
  - alpha-beta pruning not as effective
- TDGammon (world champion level!) uses only a 2 ply lookahead and a very, very good *Eval*

Sept 17, 2008

COMP 131, Lecture 5

35

## Exact values DO matter



Only positive linear transformations of *Eval* preserve behavior  
That's why *Eval* must be proportional to expected payoff

Sept 17, 2008

COMP 131, Lecture 5

36

## Games of imperfect information

- E.g., card games like poker, bridge, hearts...
- Initial deal is unknown, but we could calculate probability for every deal
  - seems like a big dice roll upfront?
  - compute the minimax values of all actions in all deals, and choose action that maximizes value over all deals? (wrong)

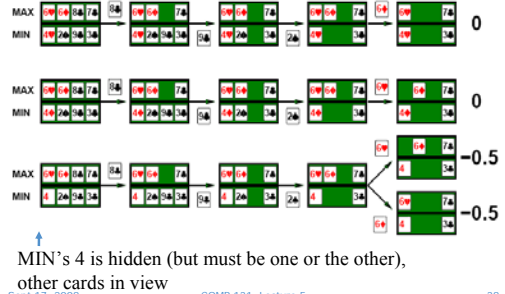
Sept 17, 2008

COMP 131, Lecture 5

37

## 2-player, 4-card tricks example

all cards in view:



Sept 17, 2008

COMP 131, Lecture 5

38

## Common-sense example

Road A leads to a brick of gold. Road B leads to a fork:  
Go **right** and you'll find a mound of jewels.  
Go **left** and you'll be run over by a bus.

Road A leads to a brick of gold. Road B leads to a fork:  
Go **left** and you'll find a mound of jewels.  
Go **right** and you'll be run over by a bus.

Road A leads to a brick of gold. Road B leads to a fork:  
Guess **right** and you'll find a mound of jewels.  
Guess **wrong** and you'll be run over by a bus.

Sept 17, 2008

COMP 131, Lecture 5

39

## Proper analysis

- In games with imperfect information, the intuition that the value of an action is the average of the values of the states it leads to is **WRONG**
- With *partial observability*, the value of an action depends on the *information state* or *belief state*
- Leads to rational behavior such as
  - acting to obtain more information
  - signaling to one's partner
  - acting randomly to confuse opponent

Sept 17, 2008

COMP 131, Lecture 5

40

## Summary

- A game is defined by **initial state**, legal **actions**, a **terminal test** and a **utility function** that applies to terminal states
- In 2-player zero-sum games with **perfect information** the **minimax** algorithm selects optimal moves using depth-first exploration of the game tree
- The **alpha-beta pruning** search computes the same optimal move as minimax, but with greater efficiency by ignoring provably irrelevant subtrees
- **Evaluation function** and **cutoff** algorithms allow search in too-big trees
- Games of chance are handled by including **chance nodes** (compute **expected utility** of all children)
- Optimal play in games with **imperfect information** requires reasoning about **belief states**

Sept 17, 2008

COMP 131, Lecture 5

41

## RoboCup

- “The ultimate goal of the RoboCup project is **By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.**”
  - from the [www.robocup.org/](http://www.robocup.org/) website
- YouTube for ‘RoboCup 2008 highlights’

Sept 17, 2008

COMP 131, Lecture 5

42