

Admisitrivia:

Programs due today!

READ Chapter 4! Midterm Next week?!

Covering Chapters 1-4. Rest of term: Chs 5-7 or 8 May post program?

Game playing

Similar to state-space search but:

1. There's an *Opponent*

whose moves must be considered

2. Because of the combinatorial explosion:

Can't search the entire tree:

Have to *commit* to a move

based on an *estimate* of its goodness.

Mechanisms we used to decide

which node to open next will be used

to decide which MOVE to make next

Game playing

History:

Shannon:

Type A

Type B

Game playing

We will usually be using

TWO-PERSON PERFECT-INFORMATION games:

You know what the other player can do at any point

E.G.

Card games:

Not an example: don't know other person's hand

Chess, checkers, etc. ARE examples

Result is WIN, LOSE or DRAW

Game playing

AND/OR (MAX/MIN) trees

Representation: AND/OR tree

- Represents moves of BOTH players

- Always from ONE player's point of view

AND branch: Other player's move

I have to be able to deal with ANY of these moves

OR branch: MY moves

I get to CHOOSE ONE of these moves

Game playing

Example: "Grundy's game"

1. Start with a stack of pennies
2. First player splits the stack into
UNEVEN piles
3. Repeatedly, each player then splits
SOME stack into uneven piles

First player who can't move loses

(every stack is either 1 or 2 pennies)

Game playing

Representation of a state:

A list of piles and whose move it is.

Players: MAX and MINNIE

E.g., (7, MAX)

from there we could have:

(1,6,MIN) or (2,5,MIN) or (3,4, MIN)

Add a function which takes a terminal node
and returns

+1 for a win for MAX,
-1 for a win for MINNIE,
0 for a draw.

We can use this function

and

the MINIMAX procedure

to decide whether MAX will win or not!

Game playing

Generate the following tree for Grundy's game:

Root MAX	Ply 1 MIN	Ply 2 MAX	Ply 3 MIN	Ply 4 MAX
(7)	(1,6)	(1,2,4)	(1,2,1,3)	(1,2,1,1,2) -1
		(1,1,5)	(1,1,1,4) *	(1,1,1,1,3) +1 (1,1,2,1,2) -1
			(1,1,2,3)	(1,2,1,1,2) -1
	(2,5)	(2,1,4)	(2,1,1,3)	(2,1,1,1,2) -1
		(2,2,3)	(2,2,1,2) +1	
	(3,4)	(1,2,4)	(1,2,1,3)	(1,2,1,1,2) -1
		(3,1,3)	(3,1,1,2)	(1,2,1,1,2) -1

Minimax evaluation of a tree

1. Generate ENTIRE GAME TREE from MAX's point of view

2. Given the terminal nodes,

a non-terminal node's value

is computed as follows:

a MAX node's value is the MAXIMUM of its successors

A MIN node's value is the MINIMUM of its successors

3. This is continued until the root is assigned a value.

This tells us whether MAX will WIN or LOSE

Now MAX can CHOOSE HIS MOVE based on these values!

At every MAX node, pick the MAXIMUM valued node below

MINNIE will choose the MINIMUM valued node below that.

In fact, we can choose EVERY MOVE from this!

Game playing

MINIMAX Search

Now,

MINIMAX applied when

the whole tree CAN'T be generated

(using a static evaluation function)

is a HEURISTIC SEARCH METHOD

We evaluate our BEST move based on how

good the HEURISTIC STATIC EVALUATION function is.

Recall:

This is usually

a weighted sum of some board features

Usually: Go some FIXED DEPTH in the tree.

Game playing

Alpha-Beta pruning:

Makes the SAME choices of "BEST MOVE" as MINIMAX

Visits FEWER nodes

How: For EACH MAX node, keep parameter ALPHA:

A LOWER BOUND on

MAX's back-up value from MINIMAX search

I.E., if we did a complete MINIMAX search,

MAX's final backed-up value for this node
would be no lower than ALPHA

Why a LOWER BOUND?

It is based on what MINNIE can hold him to
on some line of play.

MAX will always be able to CHOOSE that line
of play.

MAX will choose a BETTER line of play if he can.

Game playing

Alpha Beta Pruning

For each MIN node, keep a parameter BETA:

An UPPER BOUND on how well MINNIE can do

I.e., if we did a complete MINIMAX search,

MINNIE's value at this node
would be no higher than BETA

Why?

MINNIE is looking for LOW values of
the static evaluation function

This is based on what MAX can hold her to.

She will choose a BETTER line of play

(LOWER VALUE) if she can.

Game playing

Alpha Beta Pruning

Procedure:

While searching the tree:

The ALPHA value of a MAX node is set to the HIGHEST backed-up value of its successors

The BETA value of a MIN node is set to the LOWEST backed-up value of its successors

STOP SEARCHING WHEN:

The BETA value of a MIN node is less than

 The ALPHA value of its ancestors

The ALPHA value of a MAX node is greater than

 the BETA value of its ancestors