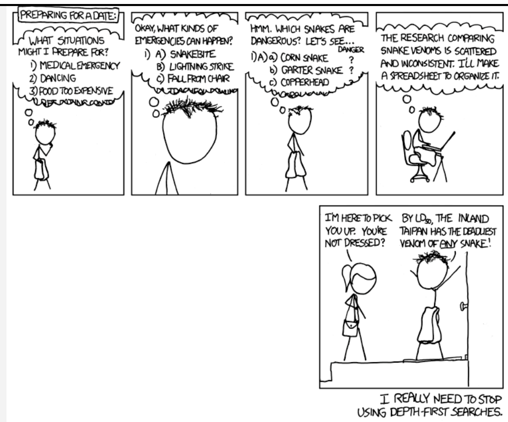


http://xkcd.com/761/

More Adversarial Search

CS151
David Kauchak
Fall 2010

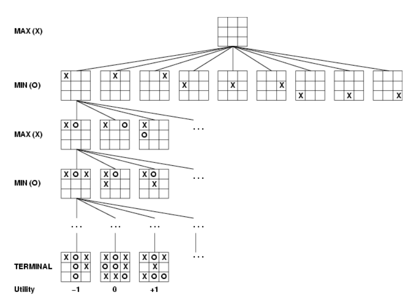
Some material borrowed from :
Sara Owsley Sood and others

Admin

- Written 2 posted
- Machine requirements for mancala
  - Most of the lab machines:
    - 2 x Dual-core 2.66Ghz Intel Xeon
    - 4 GB of RAM
    - If you want to try it out:
      - ssh to vpn.cs.pomona.edu (though don't run anything here!)
      - ssh to one of the lab machines
        - cs227-33.cs.pomona.edu (cs227-43, cs227-44)
        - `who'
        - `top'

Last time

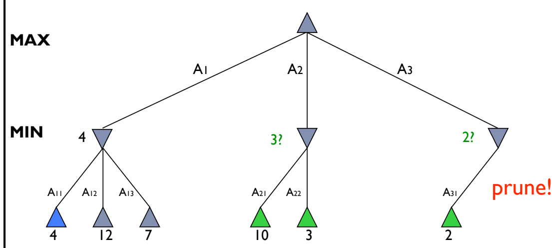
- Game playing as search



Last time

- Game playing as search
- Assume the opponent will play optimally
  - MAX is trying to maximize utility
  - MIN is trying to minimize utility
- MINIMAX algorithm backs-up the value from the leaves by alternatively minimizing and maximizing action options
  - plays "optimally", that is can play no better than
- Won't work for deep trees or trees with large branching factors
- Pruning alleviates this be excluding paths
- Alpha-Beta pruning retains the optimality, by pruning paths that will never be chosen
  - alpha is the best choice down this path for MAX
  - beta is the best choice down this path for MIN

Minimax example 2



Effectiveness of alpha-beta pruning

- As we gain more information about the state of things, we're more likely to prune
- What affects the performance of pruning?
  - key: which order we visit the states
  - can try and order them so as to improve pruning

Effectiveness of pruning

- If perfect state ordering:
  - $O(b^m)$ becomes $O(b^{m/2})$
  - We can solve a tree twice as deep!
- Random order:
  - $O(b^m)$ becomes $O(b^{3m/4})$
  - still pretty good
- For chess using a basic ordering
  - Within a factor of 2 of $O(b^{m/2})$

Evaluation functions

- $O(b^{m/2})$ is still exponential (and that's assuming optimal pruning)
  - for chess, this gets us ~10-14 ply deep (a bit more with some more heuristics)
    - 200 million moves per second (on a reasonable machine)
    - 35^5 = 50 million, or < 1 second
  - not enough to solve most games!
- Ideas?
  - heuristic function – evaluate the desirability of the position
  - This is not a new idea:
    - Claude Shannon (think-- information theory, entropy), "Programming a Computer for Playing Chess" (1950)
    - http://vision.unipv.it/IA1/ProgrammingaComputerforPlayingChess.pdf
      - page 3
      - page 5

Cutoff search

- How does an evaluation function help us?
  - search until some stopping criterion is met
  - return our heuristic evaluation of the state at that point
- When should we stop?
  - as deep as possible, for the time constraints
  - generally speaking, the further we are down the tree, the more accurate our evaluation function will be
  - based on a fixed depth
    - keep track of our depth during recursion
    - if we reach our depth limit, return EVAL(state)

Cutoff search

- When should we stop?
  - based on time
    - start a timer and run IDS
    - when we run out of time, return the result from the last completed depth
  - quiescence search
    - search using one of the cutoffs above
    - but if we find ourselves in a volatile state (for example a state where a piece is about to be captured) keep searching!
    - attempts to avoid large swings in EVAL scores

Heuristic EVAL

- What is the goal of EVAL, our state evaluation function?
  - estimate the *expected* utility of the game at a given state
- What are some requirements?
  - must be efficient (we're going to be asking this about a lot of states)
  - EVAL should play nice with terminal nodes
    - it should order terminal nodes in the same order at UTILITY
    - a win should be the most desirable thing
    - a loss should be the least desirable thing

Heuristic EVAL

- What are some desirable properties?
  - value should be higher the closer we are to a win
  - and lower the closer we are to a lose
- The quality of the evaluation function impacts the quality of the player
  - Remember last time (De Groot), we expert players were good at evaluating board states!

Simple Mancala Heuristic: Goodness of board = # stones in my Mancala minus the number of stones in my opponents.

Tic Tac Toe evaluation functions

O	X	X
	O	

Ideas?

Example Tic Tac Toe EVAL

Tic Tac Toe
Assume MAX is using "X"

EVAL(*state*) =

if state is win for MAX:
    $+\infty$
if state is win for MIN:
    $-\infty$
else:
    (number of rows, columns and diagonals available to MAX) -
    (number of rows, columns and diagonals available to MIN)

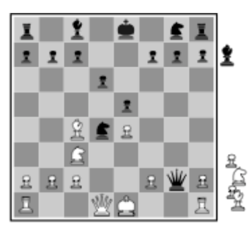
	X	O

= 6 - 4 = 2

O	X	X
	O	

= 4 - 3 = 1

Chess evaluation functions

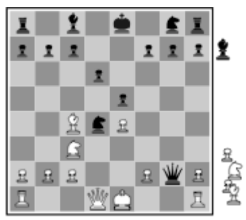


Ideas?

Chess EVAL

Assume each piece has the following values
pawn = 1;
knight = 3;
bishop = 3;
rook = 5;
queen = 9;

EVAL(*state*) =
sum of the value of white pieces –
sum of the value of black pieces

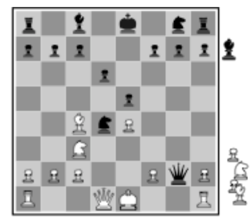


= 31 - 36 = -5

Chess EVAL

Assume each piece has the following values
pawn = 1;
knight = 3;
bishop = 3;
rook = 5;
queen = 9;

EVAL(*state*) =
sum of the value of white pieces –
sum of the value of black pieces

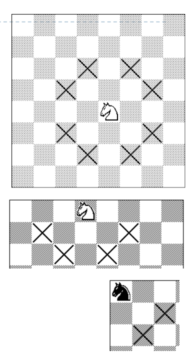


Any problems with this?

Chess EVAL

- Ignores actual positions!
- Actual heuristic functions are often a weighted combination of features

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + \ldots$$



Chess EVAL

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + \ldots$$

number of pawns

number of attacked knights

1 if king has knighted, 0 otherwise

- A feature can be any numerical information about the board
  - as general as the number of pawns
  - to specific board configurations
- Deep Blue: 8000 features!

Chess EVAL

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + \ldots$$

number of pawns

number of attacked knights

1 if king has knighted, 0 otherwise

How can we determine the weights (especially if we have 8000 of them!)?

Chess EVAL

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + \ldots$$

number of pawns

number of attacked knights

1 if king has knighted, 0 otherwise

Machine learning!
- play/examine lots of games
- adjust the weights so that the EVAL(s) correlates with the actual utility of the states

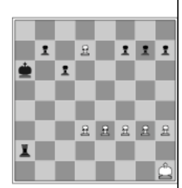
Horizon effect

Black to move

Who's ahead? What move should Black make?

Horizon effect

- The White pawn is about to become a queen
- A naïve EVAL function may not account for this behavior
- We can delay this from happening for a long time by putting the king in check
- If we do this long enough, it will go beyond our search cutoff and it will appear to be a **better** move than any move allowing the pawn to become a queen
- But it's only *delaying the inevitable* (the book also has another good example)



Black to move

Other improvements

- Computers have lots of memory these days
- DFS (or IDS) is only using a linear amount of memory
- How can we utilize this extra memory?
  - transposition table
  - history/end-game tables
  - "opening" moves
  - …

Transposition table

- Similar to keeping track of the list of explored states
- Keeps us from duplicating work
- Can double the search depth in chess!

history/end-game tables

- History
  - keep track of the quality of moves from previous games
  - use these instead of search
- end-game tables
  - do a reverse search of certain game configurations, for example all board configurations with king, rook and king
  - tells you what to do in ***any*** configuration meeting this criterion
  - if you ever see one of these during search, you lookup exactly what to do

end-game tables

- Devastatingly good
- Allows much deeper branching
  - for example, if the end-game table encodes a 20-move finish and we can search up to 14
  - can search up to depth 34
- Stiller (1996) explored all end-games with 5 pieces
  - one case check-mate required 262 moves!
- Knoval (2006) explored all end-games with 6 pieces
  - one case check-mate required 517 moves!
- Traditional rules of chess require a capture or pawn move within 50 or it's a stalemate

Opening moves

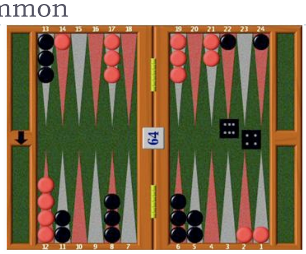
- At the very beginning, we're the farthest possible from any goal state
- People are good with opening moves
- Tons of books, etc. on opening moves
- Most chess programs use a database of opening moves rather than search

Chance/non-determinism in games

- All the approaches we've looked at are only appropriate for deterministic games
- Some games have a randomness component, often imparted either via dice or shuffling

- Why consider games of chance?
  - because they're there!
  - more realistic… life is not deterministic
  - more complicated, allowing us to further examine search techniques

Backgammon



Basic idea: move your pieces around the board and then off

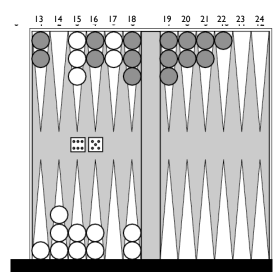
Amount you get to move is determined by a roll of two dice

Backgammon

- If we know the dice rolls, then it's straightforward to get the next states

- For example, white rolls a 5 and a 6

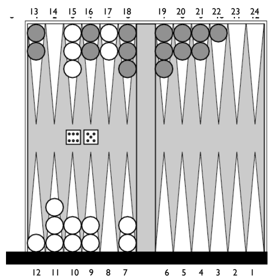
- Possible moves (7-2,7-1), (17-12,17-11), …



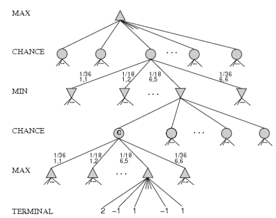
Backgammon

Which is better:
(7-2,7-1) or
(17-12,17-11)?

We'd like to search…

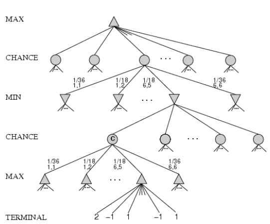


Searching with chance



- We know there are 36 different dice rolls (21 unique)
- Insert a "chance" layer in each ply with branching factor 21
- What does this do to the branching factor?
  - drastically increases the branching factor (by a factor of 21!)

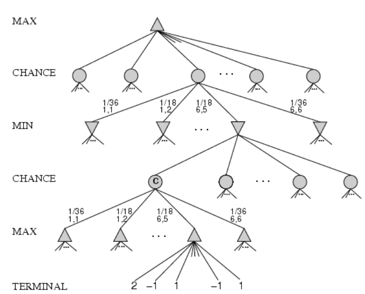
Searching with chance



- Associate a probability with each chance branch
- each double ([1,1], [2,2], …) have probability 1/36
- all others have probability 1/18

- Generally the probabilities are easy to calculate

Searching with chance



Assume we can reach the bottom. How can we calculate the value of a state?

Expected minimax value

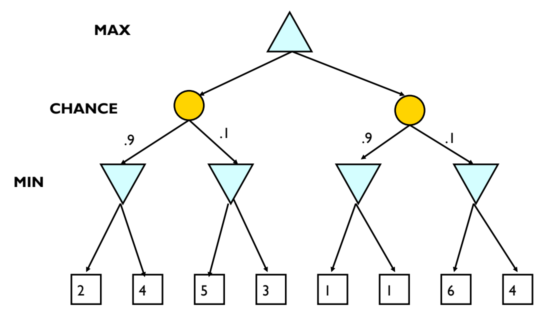
- Rather than the actual value calculate the expected value based on the probabilities

EXPECTI-MINIMAX-VALUE(n)=

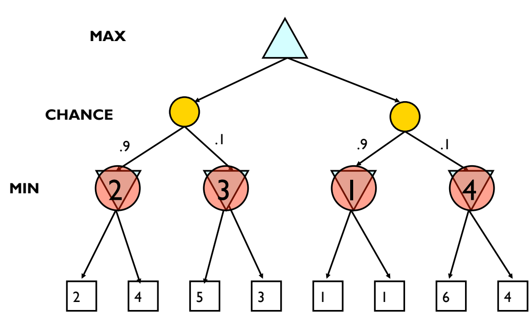
UTILITY(n)	If n is a terminal
$\max_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a max node
$\min_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a min node
$\sum_{s \in successors(n)} P(s)$. EXPECTIMINIMAX(s)	If n is a chance node

multiply the minimax value by the probability of going to that state

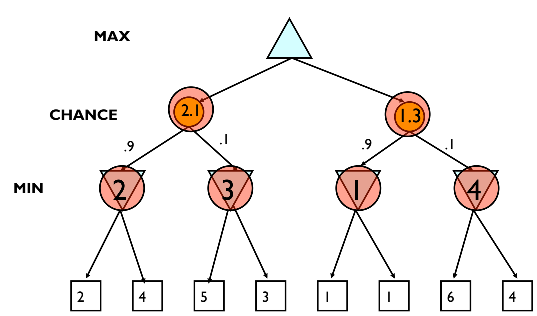
EXPECTEDMINIMAX example



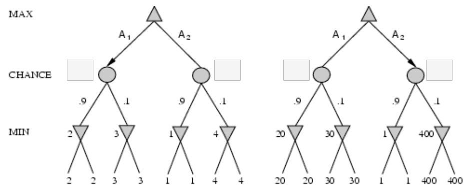
EXPECTEDIMINIMAX example



EXPECTIMINIMAX example



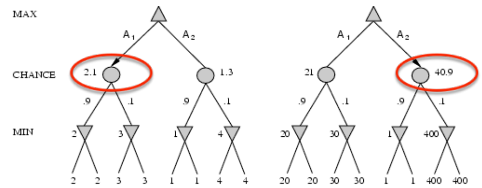
Chance and evaluation functions



Which move will be picked for these different MINIMAX trees?

Does this change any requirements on the evaluation function?

Chance and evaluation functions



Even though we haven't changed the ranking of the final states, we changed the outcome

Magnitude matters, not just ordering

Games with chance

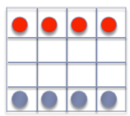
- Original branching factor *b*
- Chance factor *n*
- What happens to our search run-time?
  - $O((nb)^m)$
  - in essence, multiplies our branching factor by n
- For this reason, many games with chance don't use much search
  - backgammon frequently only looks ahead 3 ply
- Instead, evaluation functions play a more important roll
  - TD-Gammon learned an evaluation function by playing itself over a million times!

Partially observable games

- In many games we don't have all the information about the world
  - battleship
  - bridge
  - poker
  - scrabble
  - Kriegspiel
    - pretty cool game
    - "hidden" chess
  - …
- How can we deal with this?

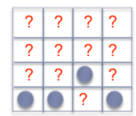
Simple Kriegspeil

- To start with:
  - I know where my pieces are
  - and I know exactly where the opponents pieces are



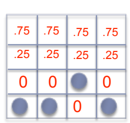
Simple Kriegspeil

- As the game progresses, though
  - I know where my pieces are
  - but I no longer know where the opponents pieces are



Simple Kriegspeil

- However, I can have some expectation/estimation of where the are



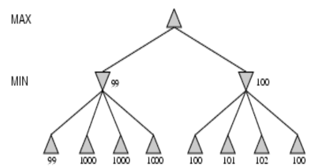
starts to look like a game of chance

Challenges with partially observable games?

- state space can be huge!
- our MINIMAX assumption is probably not true
  - reasons for the opponent to purposefully play suboptimally
- may make moves just to explore

- These are hard!
  - when humans play Kriegspeil, most of the checkmates are accidental ☺

Other things to watch out for…



- What will minimax do here?
- Is that OK?
- What might you do instead?

State of the art

- 5.7 of the book gives a pretty good recap of popular games
- Still lots of research going on!
- AAAI has an annual poker competition
- Lots of other tournaments going on for a variety of games
- New games being invented/examined all the time
  - google "quantum chess"
- University of Alberta has a big games group
  - http://webdocs.cs.ualberta.ca/~games/