Adversarial Search

CS51A
David Kauchak
Spring 2019

Some material borrowed from :
Sara Owsley Sood and others

Admin

Assignment 10

A quick review of search

Problem solving via search:

- To define the state space, define three things:
  - is_goal
  - next_states
  - starting state

Uninformed search vs. informed search
- what's the difference?
- what are the techniques we've seen?
- pluses and minuses?

Why should we study games?

Clear success criteria

Important historically for AI

Fun ☺

Good application of search
- hard problems (chess 35^{100} states in search space, 10^{40} legal states)

Some real-world problems fit this model
- game theory (economics)
- multi-agent problems

Types of games

What are some of the games you've played?

Types of games: game properties

single-player vs. 2-player vs. multiplayer

Fully observable (perfect information) vs. partially observable

Discrete vs. continuous

real-time vs. turn-based

deterministic vs. non-deterministic (chance)

Strategic thinking $\stackrel{?}{=}$ intelligence

For reasons previously stated, two-player games have been a focus of AI since its inception…

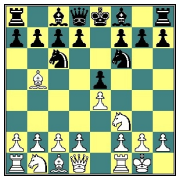
Important question: Is strategic thinking the same as intelligence?

Strategic thinking $\stackrel{?}{=}$ intelligence

Humans and computers have different relative strengths in these games:

humans

?



computers

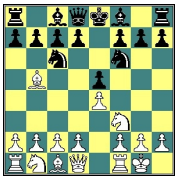
?

Strategic thinking ? = intelligence

Humans and computers have different relative strengths in these games:

humans

good at evaluating the strength of a board for a player



computers

good at looking ahead in the game to find winning combinations of moves

Strategic thinking ? = intelligence

How could you figure out how humans approach playing chess?

humans

good at evaluating the strength of a board for a player

How humans play games…

An experiment was performed in which chess positions were shown to novice and expert players…

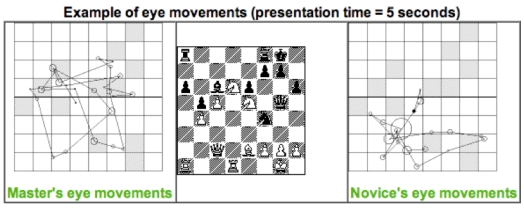
- experts could reconstruct these perfectly
- novice players did far worse…

How humans play games…

Random chess positions (not legal ones) were then shown to the two groups

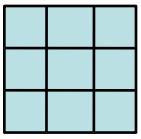
experts and novices did just as badly at reconstructing them!

People are still working on this problem…



http://people.brunel.ac.uk/~hsstffg/frg-research/chess_expertise/

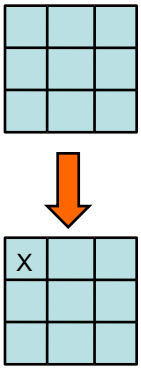
Tic Tac Toe as search



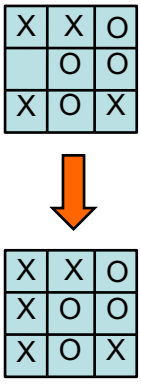
If we want to write a program to play tic tac toe, what question are we trying to answer?

Given a state (i.e. board configuration), what move should we make!

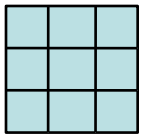
Tic Tac Toe as search



Tic Tac Toe as search

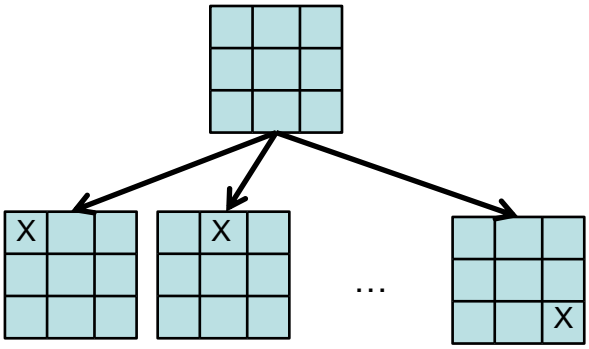


Tic Tac Toe as search

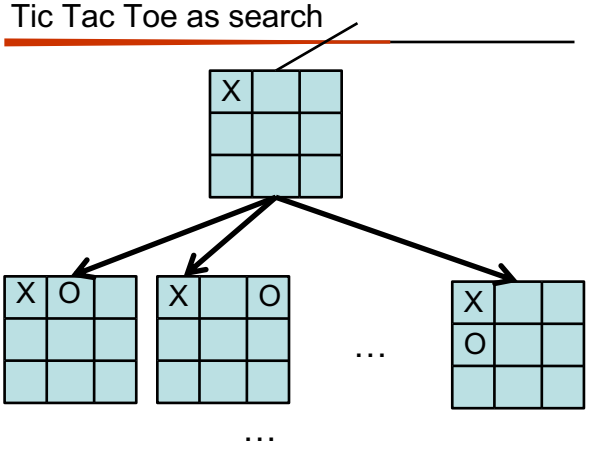


How can we pose this as a search problem?

Tic Tac Toe as search

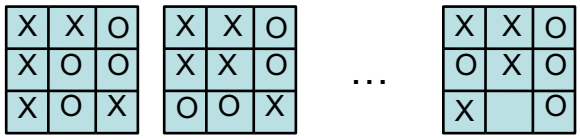


Tic Tac Toe as search



Tic Tac Toe as search

Eventually, we'll get to a leaf

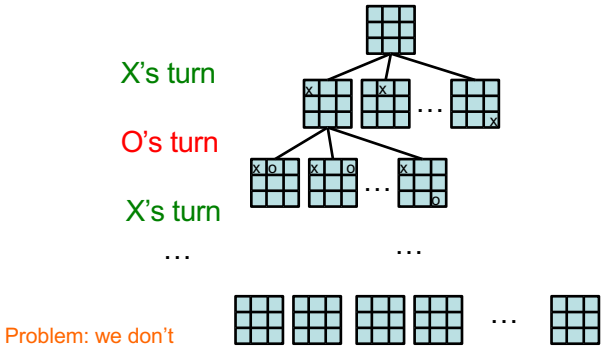


WIN TIE LOSE

How does this help us?

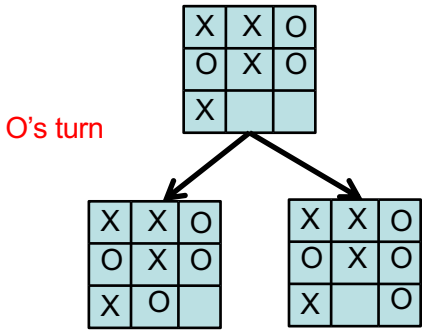
Try and make moves that move us towards a win, i.e. where there are leaves with a WIN.

Tic Tac Toe



Problem: we don't know what O will do

I'm X, what will 'O' do?

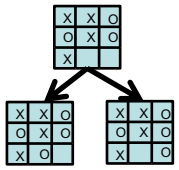


Minimizing risk

The computer doesn't know what move O (the opponent) will make

It can *assume* that it will try and make the best move possible

Even if O actually makes a different move, we're no worse off. Why?



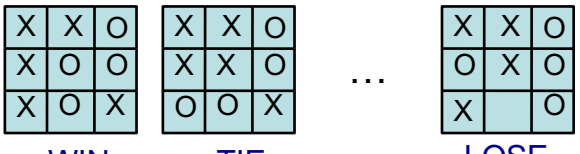
Optimal Strategy

An Optimal Strategy is one that is at least as good as any other, no matter what the opponent does

– If there's a way to force the win, it will

– Will only lose if there's no other option

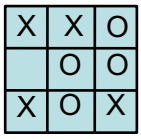
Defining a scoring function



Idea:
- define a function that gives us a "score" for how good each state is
- higher scores mean better

Defining a scoring function

Our (X) turn

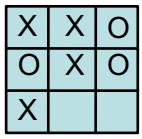


What should be the score of this state?

+1: we can get to a win

Defining a scoring function

Opponent's (O) turn

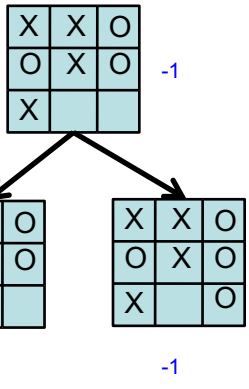


What should be the score of this state?

-1: we can get to a win

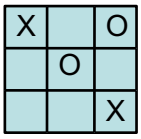
Defining a scoring function

Opponent's (O) turn



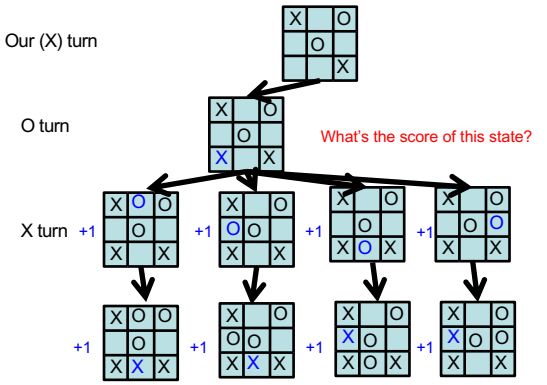
Defining a scoring function

Our (X) turn

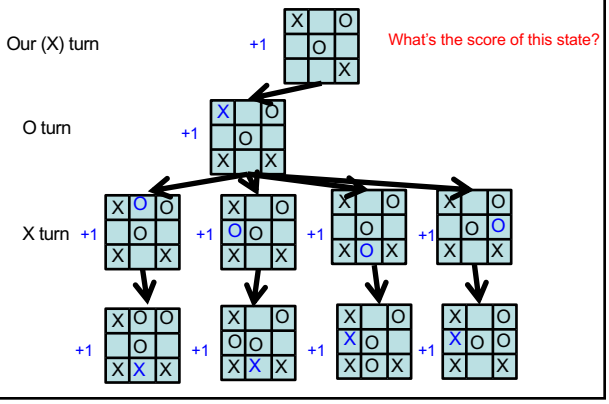


What should be the score of this state?

Defining a scoring function

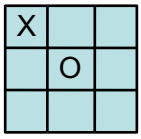


Defining a scoring function



Defining a scoring function

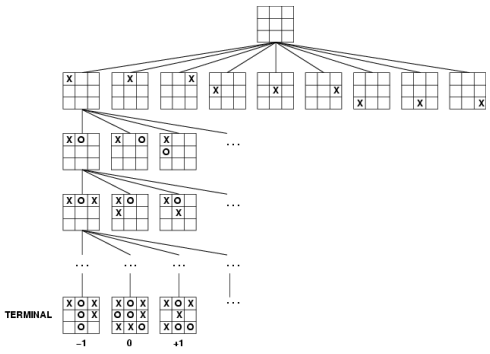
Our (X) turn



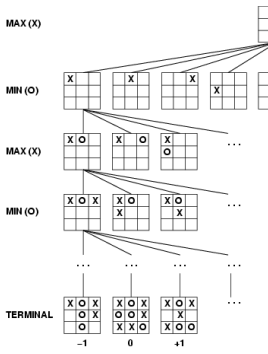
What should be the score of this state?

0: If we play perfectly and so does O, the best we can do is a tie (could do better if O makes a mistake)

How can X play optimally?



How can X play optimally?



When it's my turn, pick the highest scoring state

When it's the opponent's turn, assume the lowest scoring state (from my perspective)

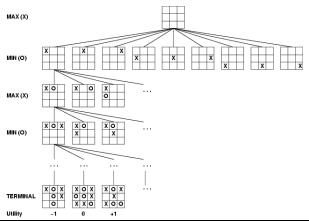
If we can reach the end games, we can percolate these answers all the way back up

How can X play optimally?

Start from the leaves and propagate the score up:

- if X's turn, pick the move that maximizes the utility
- if O's turn, pick the move that minimizes the utility

Is this optimal?



Minimax Algorithm: An Optimal Strategy

```
minimax(state) =
    if state is a terminal state
        score(state)
    else if MY turn
        over all next states, s: return the maximum of minimax(s)
    else if OPPONENTS turn
        over all next states, s: return the minimum of minimax(s)
```

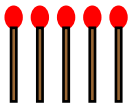
Uses recursion to compute the "value" of each state

Searches down to the leaves, then the values are "backed up" through the tree as the recursion finishes

What type of search is this?

What does this assume about how MIN will play? What if this isn't true?

Baby Nim



Take 1 or 2 at each turn
Goal: take the last match

What move should I take?