Adversarial Search

CS51A
David Kauchak
Spring 2022

Some material borrowed from:
Sara Owsley Sood and others

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 3510 states in search space, 1012 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 ≠ 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 ≠ intelligence

Humans and computers have different relative strengths in these games:

humans
computers

?
Humans and computers have different relative strengths in these games:

- Humans are good at evaluating the strength of a board for a player.
- Computers are good at looking ahead in the game to find winning combinations of moves.

How could you figure out how humans approach playing chess?

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...

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

How can we pose this as a search problem?

Now what?

O's turn!
Tic Tac Toe as search

Eventually, we’ll get to a leaf

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th></th>
<th>WIN</th>
<th>TIE</th>
<th>LOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

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

What should be the score of this state?

+1: we can get to a win

What should be the score of this state?

-1: opponent can get to a win
Defining a scoring function

Opponent's (O) turn

\[
\begin{array}{ccc}
X & X & O \\
O & X & O \\
X & & \\
\end{array}
\]

\[+1\] \[\text{-1}\]

What should be the score of this state?

Our (X) turn

\[
\begin{array}{ccc}
X & O & O \\
O & & X \\
\end{array}
\]

What's the score of this state?