Adversarial Search

George Konidaris
gdk@cs.brown.edu
Games

“Chess is the Drosophila of Artificial Intelligence”
Kronrod, c. 1966

TuroChamp, 1948
Games


“The chess machine is an ideal one to start with, since: (1) the problem is sharply defined both in allowed operations (the moves) and in the ultimate goal (checkmate); (2) it is neither so simple as to be trivial nor too difficult for satisfactory solution; (3) chess is generally considered to require "thinking" for skillful play; a solution of this problem will force us either to admit the possibility of a mechanized thinking or to further restrict our concept of "thinking"; (4) the discrete structure of chess fits well into the digital nature of modern computers.”
“Solved” Games

A game is solved if an optimal strategy is known.

Strong solved: *all positions.*
Weakly solved: *some (start) positions.*
Typical Game Setting

Games are usually:

- 2 player
- Alternating
- Zero-sum
  - Gain for one loss for another.
- Perfect information
Typical Game Setting

Very much like search:
• Set of possible states
• Start state
• Successor function
• Terminal states (many)
• Objective function

*The key difference is alternating control.*
Game Trees

player 1 moves

player 2 moves

player 1 moves
Key Differences vs. Search

you select to max score

they select to min score

only get score here

only get score here
Minimax

Propagate value backwards through tree.

\[ V(s_0) = \max(V(s_1), V(s_2), V(s_3)) \]

\[ V(s_2) = \min(V(s_4), V(s_5), V(s_6)) \]

\[ V(s_5) = \max(V(g_1), V(g_2), V(g_3)) \]
Minimax Algorithm

Compute value for each node, going backwards from the end-nodes.

Max (min) player: select action to maximize (minimize) return.

Optimal for both players (if zero sum).
Assumes perfect play, worst case.

Can run as depth first:
  • Time $O(b^d)$
  • Space $O(bd)$

Require the agent to evaluate the whole tree.
Minimax
Games of Chance

What if there is a chance element?
Stochasticity

An outcome is called **stochastic** when it is determined at random.

A green dice with the numbers 1 to 6 is shown. Each number is associated with a probability of $p = \frac{1}{6}$.

The probabilities sum to $1$.

$1 \rightarrow p = \frac{1}{6}$
$2 \rightarrow p = \frac{1}{6}$
$3 \rightarrow p = \frac{1}{6}$
$4 \rightarrow p = \frac{1}{6}$
$5 \rightarrow p = \frac{1}{6}$
$6 \rightarrow p = \frac{1}{6}$
Stochasticity

How to factor in stochasticity?

Agent does not get to choose.
  • Selecting the $\max$ outcome is optimistic.
  • Selecting the $\min$ outcome is pessimistic.

Must be probability-aware.

Be aware of **who is choosing** at each level.
  • Sometimes it is you.
  • Sometimes it is an adversary.
  • Sometimes it is a random number generator.

**insert randomization layer**
ExpectiMax

- you select
  (max)
- they select to
  min score

stochastic
Expectation

**How to compute value of stochastic layer?**

What is the *average die value*?

\[
\frac{(1 + 2 + 3 + 4 + 5 + 6)}{6} = 3.5
\]

*This factors in both probabilities and the value of event.*

In general, given random event \( x \) and function \( f(x) \):

\[
E[f(x)] = \sum_x P(x)f(x)
\]
ExpectiMax

They select to min score

You select (max)

Stochastic (expectation)

Stochastic (expectation)
Minimax

```
  5
   |
 p1
   |
  5
   |
 p2
   |
-3
   |
 p2
   |
-3
   |
 p2
   |
10
   |
 p2
   |
-5
   |
 p2
   |
 20
   |
 p2
   |
-5
   |
 p2
   |
```

Max

Min
In Practice

Can run as depth first:
  - Time $O(b^d)$
  - Space $O(bd)$

**Depth is too deep.**
  - 10s to 100s of moves.

**Breadth is too broad.**
  - Chess: 35, Go: 361.

Full search never terminates for non-trivial games.
What Is To Be Done?

Terminate early.
Branch less often.
At a min layer:
If $V(B) \leq V(A)$ then prune B’s siblings.
At a max layer:
If $V(A) \geq V(B)$ then prune A’s siblings.
More generally:
• $\alpha$ is highest max
• $\beta$ is lowest min

If max node:
• prune if $v \geq \beta$

If min node:
• prune if $v \leq \alpha$
function ALPHA-BETA-SEARCH(state) returns an action
    \( v \leftarrow \text{MAX-VALUE}(state, -\infty, +\infty) \)
    return the action in ACTIONS(state) with value \( v \)

function MAX-VALUE(state, \( \alpha \), \( \beta \)) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    \( v \leftarrow -\infty \)
    for each \( a \) in ACTIONS(state) do
        \( v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta)) \)
        if \( v \geq \beta \) then return \( v \)
        \( \alpha \leftarrow \text{MAX}(\alpha, v) \)
    return \( v \)

function MIN-VALUE(state, \( \alpha \), \( \beta \)) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    \( v \leftarrow +\infty \)
    for each \( a \) in ACTIONS(state) do
        \( v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta)) \)
        if \( v \leq \alpha \) then return \( v \)
        \( \beta \leftarrow \text{MIN}(\beta, v) \)
    return \( v \)

(from Russell and Norvig)
Alpha Beta Pruning

Single most useful search control method:
- Throw away whole branches.
- Use the min-max behavior.

Resulting algorithm: \textit{alpha-beta pruning}.

Empirically: \textit{square roots} branching factor.
- Effectively doubles the search horizon.

Alpha-beta makes the difference between novice and expert computer game players. \textit{Most successful players use alpha-beta.}
What Is To Be Done?

Terminate early.
Branch less often.
In Practice

Solution: *substitute evaluation function.*
  - Like a heuristic - *estimate value.*
  - In this case, **probability of win or expected score.**

- Common strategy:
  - Run to fixed depth then estimate.
  - Careful lookahead to depth $d$, then *guess.*
Evaluation Functions
Evaluation Functions
Deep Blue (1997)

480 Special Purpose Chips
200 million positions/sec
Search depth 6-8 moves (up to 20)
Evaluation Functions
Search Control

Horizon Effects
- What if something interesting at horizon + 1?
- How do you know?

More sophisticated strategies:
- When to generate more nodes?
- How to selectively expand the frontier?
- How to allocate fixed move time?
Monte Carlo Tree Search

Continually estimate value
Adaptively explore
Random rollouts to evaluate
Monte Carlo Tree Search

Step 1: path selection.
Monte Carlo Tree Search

Step 1: path selection.

\[ \frac{w_i}{n_i} + c \sqrt{\frac{\log n}{n_i}} \]

UCT

...
Monte Carlo Tree Search

Step 2: expansion.
Monte Carlo Tree Search

Step 3: rollout.

terminal state
Step 4: update.
Games Today

World champion level:
• Backgammon
• Chess
• Checkers (solved)
• Othello
• Some poker types:

Perform well:
• Bridge
• Other poker types

Far off: Go
Very Recently

Lee Sedol

1 - 4

AlphaGo

AlphaGo (Google Deepmind)
Board Games

“... board games are more or less done and it's time to move on.”