Multiagent Systems
Game Playing

Viviana Mascardi

viviana.mascardi@unige.it
1 Games

2 Perfect Play
   • Minimax
   • $\alpha-\beta$ pruning

3 Resource limits and approximate evaluation

4 Conclusions
We will not deal with...
We will not deal with...
We will not deal with...

....although....
We will not deal with...
We will not deal with...
We will not deal with...
We will not deal with...
We will not deal with...
We will not deal with...

<table>
<thead>
<tr>
<th>Prisoner A</th>
<th>Prisoner B stays silent <em>(cooperates)</em></th>
<th>Prisoner B betrays <em>(defects)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prisoner A stays silent <em>(cooperates)</em></td>
<td>Each serves 1 year</td>
<td>Prisoner A: 3 years</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Prisoner B: goes free</td>
</tr>
<tr>
<td>Prisoner A betrays <em>(defects)</em></td>
<td>Prisoner A: goes free</td>
<td>Each serves 2 years</td>
</tr>
<tr>
<td></td>
<td>Prisoner B: 3 years</td>
<td></td>
</tr>
</tbody>
</table>
We will not deal with...

**Strategy for the prisoner's dilemma**

Both cannot communicate, they are separated in two individual rooms. The normal game is shown below:

<table>
<thead>
<tr>
<th>Prisoner A</th>
<th>Prisoner B stays silent (cooperates)</th>
<th>Prisoner B betrays (defects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prisoner A stays silent (cooperates)</td>
<td>Each serves 1 year</td>
<td>Prisoner A: 3 years Prisoner B: goes free</td>
</tr>
<tr>
<td>Prisoner A betrays (defects)</td>
<td>Prisoner A: goes free Prisoner B: 3 years</td>
<td>Each serves 2 years</td>
</tr>
</tbody>
</table>

It is assumed that both understand the nature of the game, and that despite being members of the same gang, they have no loyalty to each other and will have no opportunity for retribution or reward outside the game. Regardless of what the other decides, each prisoner gets a higher reward by betraying the other ("defecting"). The reasoning involves an argument by dilemma: B will either cooperate or defect. If B cooperates, A should defect, because going free is better than serving 1 year. If B defects, A should also defect, because serving 2 years is better than serving 3. So either way, A should defect. Parallel reasoning will show that B should defect.

Because defection always results in a better payoff than cooperation, regardless of the other player's choice, it is a dominant strategy. Mutual defection is the only strong Nash equilibrium in the game (i.e. the only outcome from which each player could only do worse by unilaterally changing strategy). The dilemma then is that mutual cooperation yields a better outcome than mutual defection but it is not the rational outcome because from a self-interested perspective, the choice to cooperate, at the individual level, is irrational.
We will not deal with...

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1994

"for their pioneering analysis of equilibria in the theory of non-cooperative games"

John F. Nash Jr.
USA
Princeton University
Princeton, NJ, USA
b. 1928
# Types of games

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perfect info</strong></td>
<td>chess, checkers, go, othello</td>
<td>backgammon monopoly</td>
</tr>
<tr>
<td><strong>Imperfect info</strong></td>
<td>battleships, blind tic-tactoe</td>
<td>bridge, poker, scrabble nuclear war</td>
</tr>
</tbody>
</table>
Game tree (2-player, deterministic, turns, zero-sum)
Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest minimax value
= best achievable payoff against best play

E.g., 2-ply game:
Minimax algorithm

**function** `MINIMAX-DECISION(state)` **returns** an action

**inputs** `state`, current state in game

\[ v \leftarrow \text{MAX-VALUE}(state) \]

**return** the action in `SUCCESSORS(state)` with value \( v \)

**function** `MAX-VALUE(state)` **returns** a utility value

**if** `TERMINAL-TEST(state)` **then return** `UTILITY(state)`

\[ v \leftarrow -\infty \]

**foreach** `a,s in SUCCESSORS(state)` **do** \( v \leftarrow \text{MAX}(v,\text{MIN-VALUE}(s)) \)

**return** \( v \)

**function** `MIN-VALUE(state)` **returns** a utility value

**if** `TERMINAL-TEST(state)` **then return** `UTILITY(state)`

\[ v \leftarrow \infty \]

**foreach** `a,s in SUCCESSORS(state)` **do** \( v \leftarrow \text{MIN}(v,\text{MAX-VALUE}(s)) \)

**return** \( v \)
Properties of minimax

Complete??
Properties of minimax

**Complete** - Only if tree is finite (chess has specific rules for this).

NB: a finite strategy can exist even in an infinite tree!

Time??
Properties of minimax

Complete - Only if tree is finite (chess has specific rules for this). NB: a finite strategy can exist even in an infinite tree!

Time - $O(b^m)$

Space??
Properties of minimax

**Complete** - Only if tree is finite (chess has specific rules for this). NB: a finite strategy can exist even in an infinite tree!

**Time** - $O(b^m)$

**Space** - $O(bm)$ (depth-first exploration)

Optimal??
Properties of minimax

Complete - Only if tree is finite (chess has specific rules for this). NB: a finite strategy can exist even in an infinite tree!

Time - $O(b^m)$

Space - $O(bm)$ (depth-first exploration)

Optimal - Yes, against an optimal opponent. Otherwise??
Properties of minimax

**Complete** - Only if tree is finite (chess has specific rules for this). NB: a finite strategy can exist even in an infinite tree!

**Time** - $O(b^m)$

**Space** - $O(bm)$ (depth-first exploration)

**Optimal** - Yes, against an optimal opponent. Otherwise??

For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games $\Rightarrow$ exact solution completely infeasible

But do we need to explore every path?
$\alpha - \beta$ pruning example

The diagram illustrates the $\alpha - \beta$ pruning technique in a game tree. The MAX node prunes branches below a certain value, represented by the triangle indicating $\geq 3$. The MIN node evaluates the values 3, 12, and 8, and the pruning occurs when the MAX node prunes the branches to 12 and 8, as they are below the threshold.
$\alpha-\beta$ pruning example
\( \alpha - \beta \) pruning example

\[
\begin{array}{c}
\text{MAX} \\
\begin{array}{c}
\text{MIN} \\
3 \\
12 \\
8 \\
2 \\
14
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\geq 3 \\
\leq 2 \\
\leq 14
\end{array}
\]

X X
\( \alpha - \beta \) pruning example

Diagram:

```
\[
\begin{array}{c}
\text{MAX} \\
3 \\
3 \\
12 \\
8 \\
2 \\
\text{MIN} \\
\leq 3 \\
\leq 2 \\
\leq 5 \\
14 \\
5 \\
\end{array}
\]
```
\(\alpha-\beta\) pruning example
Why is it called $\alpha-\beta$?

$\alpha$ is the best value (to $\text{MAX}$) found so far off the current path.
If $V$ is worse than $\alpha$, $\text{MAX}$ will avoid it $\Rightarrow$ prune that branch.
Define $\beta$ similarly for $\text{MIN}$. 

$\alpha$ is the best value (to $\text{MAX}$) found so far off the current path. If $V$ is worse than $\alpha$, $\text{MAX}$ will avoid it $\Rightarrow$ prune that branch. Define $\beta$ similarly for $\text{MIN}$. 

Diagram:

```
\text{MAX} \\
\text{MIN} \\
\vdots \\
\vdots \\
\text{MAX} \\
\text{MIN} \\
\alpha \\
\Rightarrow \\
v
```
The $\alpha$–$\beta$ algorithm

function $\text{ALPHA-BETA-SEARCH}(\text{state})$ returns an action
  inputs state, current state in game
  
  $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
  return the action in $\text{SUCCESSORS}(\text{state})$ with value $v$

function $\text{MAX-VALUE}(\text{state}, \alpha, \beta)$ returns a utility value
  inputs state, current state in game
  $\alpha$, the value of the best alternative for MAX along the path to state
  $\beta$, the value of the best alternative for MIN along the path to state
  
  if $\text{TERMINAL-TEST}(\text{state})$ then return $\text{UTILITY}(\text{state})$
  $v \leftarrow -\infty$
  foreach $a,s$ in $\text{SUCCESSORS}(\text{state})$ do
    $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$
    if $v \geq \beta$ then return $v$
    $\alpha \leftarrow \text{MAX}(\alpha, v)$
  
  return $v$
The $\alpha$–$\beta$ algorithm

function $\text{MIN-VALUE}(state, \alpha, \beta)$ returns a utility value

inputs $state$, current state in game
- $\alpha$, the value of the best alternative for MAX along the path to state
- $\beta$, the value of the best alternative for MIN along the path to state

if $\text{TERMINAL-TEST}(state)$ then return $\text{UTILITY}(state)$

$v \leftarrow +\infty$

foreach $a,s$ in $\text{SUCCESSORS}(state)$ do

$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s, \alpha, \beta))$

if $v \leq \alpha$ then return $v$

$\beta \leftarrow \text{MIN}(\beta, v)$

return $v$
Animations of $\alpha-\beta$

- http://proof.github.io/minimax/
Properties of $\alpha-\beta$

Pruning does not affect final result

Good move ordering improves effectiveness of pruning

With “perfect ordering,” time complexity $= O(b^{m/2})$
  $\Rightarrow$ doubles solvable depth

A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

Unfortunately, $35^{50}$ is still impossible!
Resource limits

Standard approach (Shannon, 1950, “Programming a computer for playing chess”):

- Use **Cutoff-Test** instead of **Terminal-Test**
  e.g., depth limit (perhaps add quiescence search)
- Use **Eval** instead of **Utility**
  i.e., evaluation function that estimates desirability of position

Suppose we have 100 seconds, explore $10^4$ nodes/second

$$\Rightarrow 10^6 \text{ nodes per move } \approx 35^{8/2}$$
$$\Rightarrow \alpha-\beta \text{ reaches depth 8 } \Rightarrow \text{ pretty good chess program}$$
Summary

1950
1956
1974-1980
1987-1993
2008
2011/2012
2013
2015
2016

Alan Turing creates the "Turing Test" to determine a machine's human ability.

Stanford Research Institute creates Shakey, the first robot with self-reasoning.

Symbols Lisp Machines commercialized - AI renaissance.

IBM Deep Blue defeats Garry Kasparov at Chess.

IBM Watson defeats Ken Jennings on Jeopardy.

Apple introduces Siri, intelligent personal assistant.

Facebook develops DeepFace, near-human accuracy.

Google DeepMind's AlphaGo defeats Lee Sedol at Go.

Uber pilots self-driving car program in Pittsburgh, PA.

Google Brain developed 2% accuracy identifying cats on YouTube.

China's Tianhe-2 doubles world's Top Supercomputing Speed (33.86 petaflops).

Google open-sources TensorFlow software library.

World's Top Supercomputing Speed (93 petaflops).

Mas, University of Genoa, DIBRIS
Did Elon Musk’s AI champ destroy humans at video games? It’s complicated

By James Vincent | @jjvincent | Aug 14, 2017, 1:54pm EDT

You might not have noticed, but over the weekend a little coup took place. On Friday night, in front of a crowd of thousands, an AI bot beat a professional human player at Dota 2 — one of the world’s most popular video games. The human


MAS, University of Genoa, DIBRIS
https://www.wired.com/story/googles-ai-declares-galactic-war-on-starcraft-/
Summary

GOOGLE'S AI DECLARES
GALACTIC WAR ON STARCRAFT