Introduction to Artificial Intelligence

Game Playing

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

Summer Term 2003
Outline

- Perfect play
- Resource limits
- $\alpha$-$\beta$ pruning
- Games of chance
- Games of imperfect information
Games vs. Search Problems

Game playing is a search problem

Defined by

– Initial state
– Successor function
– Goal test
– Path cost / utility / payoff function

Characteristics of game playing

“Unpredictable” opponent:
Solution is a strategy specifying a move for every possible opponent reply

Time limits:
Unlikely to find goal, must approximate
Game Playing

Plan of attack

- Computer considers possible lines of play [Babbage, 1846]
- Algorithm for perfect play [Zermelo, 1912; Von Neumann, 1944]
- Finite horizon, approximate evaluation [Zuse, 1945; Wiener, 1948; Shannon, 1950]
- First chess program [Turing, 1951]
- Machine learning to improve evaluation accuracy [Samuel, 1952–57]
- Pruning to allow deeper search [McCarthy, 1956]
# Types of Games

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Chance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect Information</td>
<td>Chess, checkers, go, othello</td>
<td>Backgammon, monopoly</td>
</tr>
<tr>
<td>Imperfect Information</td>
<td></td>
<td>Bridge, poker, scrabble, nuclear war</td>
</tr>
</tbody>
</table>

B. Beckert: Einführung in die KI / KI für IM – p.5
Game Tree: 2-Player / Deterministic / Turns

MAX (X)

MIN (O)

MAX (X)

MIN (O)

TERMINAL

Utility

−1
0
+1

B. Beckert: Einführung in die KI / KI für IM – p.6
Minimax

Perfect play for deterministic, perfect-information games

Idea

Choose move to position with highest minimax value, i.e., best achievable payoff against best play
Minimax: Example

2-ply game

MAX

MIN
Minimax Algorithm

```
function MINIMAX-DECISION(game) returns an operator

    for each op in OPERATORS[game] do
        VALUE[op] ← MINIMAX-VALUE(APPLY(op, game), game)
    end

    return the op with the highest VALUE[op]

```

```
function MINIMAX-VALUE(state, game) returns a utility value

    if TERMINAL-TEST[game](state) then
        return UTILITY[game](state)
    else if MAX is to move in state then
        return the highest MINIMAX-VALUE of SUCCESSORS(state)
    else
        return the lowest MINIMAX-VALUE of SUCCESSORS(state)

```
## Properties of Minimax

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>Yes, if tree is finite (chess has specific rules for this)</td>
</tr>
<tr>
<td><strong>Optimal</strong></td>
<td>Yes, against an optimal opponent. Otherwise??</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(b^m)$ (depth-first exploration)</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(bm)$ (depth-first exploration)</td>
</tr>
</tbody>
</table>

**Note**

Finite strategy can exist even in an infinite tree
Resource Limits

Complexity of chess

\( b \approx 35, \ m \approx 100 \) for “reasonable” games

Exact solution completely infeasible

Standard approach

- Cutoff test
  - e.g., depth limit (perhaps add quiescence search)

- Evaluation function
  - Estimates desirability of position
Evaluation Functions

Estimate desirability of position

Black to move
White slightly better

White to move
Black winning
Evaluation Functions

**Typical evaluation function for chess**

**Weighted sum of features**

\[
	ext{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s)
\]

**Example**

\[
\begin{align*}
  w_1 &= 9 \\
  f_1(s) &= \text{(number of white queens)} - \text{(number of black queens)}
\end{align*}
\]
Cutting Off Search

Does it work in practice?

\[ b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4 \]

Not really, because …

- 4-ply \( \approx \) human novice (hopeless chess player)
- 8-ply \( \approx \) typical PC, human master
- 12-ply \( \approx \) Deep Blue, Kasparov
\(\alpha - \beta\) Pruning Example

MAX

MIN

\[\geq 3\]
Pruning Example

\( \alpha - \beta \) Pruning Example

MAX

MIN

B. Beckert: Einführung in die KI / KI für IM – p.16
**α-β Pruning Example**

Max

MIN

B. Beckert: Einführung in die KI / KI für IM – p. 16
\( \alpha - \beta \) Pruning Example

MAX

MIN

3
12
8

3
2

14
5

\( \geq 3 \)

\( \leq 2 \)

\( \times \)
Pruning Example

MAX

MIN

B. Beckert: Einführung in die KI / KI für IM – p.16
Properties of $\alpha$-$\beta$

Effects of pruning

- Reduces the search space
- Does not affect final result

Effectiveness

Good move ordering improves effectiveness

Time complexity with “perfect ordering”: $O(b^{m/2})$

Doubles depth of search

For chess:
Can easily reach depth 8 and play good chess
The Idea of $\alpha$-$\beta$

$\alpha$ is the best value (to MAX) found so far off the current path

If value $x$ of some node below $V$ is known to be less than $\alpha$,

then value of $V$ is known to be at most $x$, i.e., less than $\alpha$,

therefore MAX will avoid node $V$

Consequence

No need to expand further nodes below $V$
The $\alpha$-$\beta$ Algorithm

function $\max$-$\text{VALUE}(\text{state}, \text{game}, \alpha, \beta)$ returns the minimax value of $\text{state}$

inputs: $\text{state}$ /* current state in game */
         $\text{game}$ /* game description */
         $\alpha$ /* the best score for $\text{MAX}$ along the path to $\text{state}$ */
         $\beta$ /* the best score for $\text{MIN}$ along the path to $\text{state}$ */

if $\text{CUTOFF-TEST}(\text{state})$ then return $\text{EVAL}(\text{state})$

for each $s$ in $\text{SUCCESSORS}(\text{state})$ do
    $\alpha \leftarrow \max(\alpha, \min$-$\text{VALUE}(s, \text{game}, \alpha, \beta))$
    if $\alpha \geq \beta$ then return $\beta$
end

return $\alpha$
The $\alpha$-$\beta$ Algorithm

```plaintext
function \textsc{Min-Value}(state, game, \alpha, \beta) returns the minimax value of \textit{state}

inputs: state /* current state in game */
        game /* game description */
        \alpha /* the best score for MAX along the path to state */
        \beta /* the best score for MIN along the path to state */

if \textsc{Cutoff-Test}(state) then return \textsc{Eval}(state)

for each \textit{s} in \textsc{Successors}(state) do
    \beta \leftarrow \textsc{Min}((\beta, \textsc{Max-Value}(s, game, \alpha, \beta))
    if $\beta \leq \alpha$ then return $\alpha$

end

return $\beta$
```

B. Beckert: Einführung in die KI / KI für IM – p.20
Deterministic Games in Practice

Checkers

Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

Chess

Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

Go

Human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.
Nondeterministic Games: Backgammon
Nondeterministic Games in General

Chance introduced by dice, card-shuffling, etc.

Simplified example with coin-flipping

```
+---+---+---+---+
|   | 3 |   |   |
+---+---+---+---+
| 0.5 | 0.5 |
+---+---+
| MIN 2 | 4 |
+---+---+---+---+
| 2 | 4 | 7 | 4 |
+---+---+---+---+
| MAX |   |   |   |
+---+---+---+---+
| 0.5 | 0.5 | 0.5 | 0.5 |
+---+---+---+---+
| CHANCE | 3 |
+---+---+
| 0.5 | 0.5 |
+---+---+
| MIN | 2 | 4 |
| 2 | 4 |
```

B. Beckert: Einführung in die KI / KI für IM – p.23
Algorithm for Nondeterministic Games

**EXPECTMINIMAX** gives perfect play

if *state* is a **Max** node then
return the highest *EXPECTMINIMAX* value of *SUCCESSORS*(*state*)

if *state* is a **Min** node then
return the lowest *EXPECTMINIMAX* value of *SUCCESSORS*(*state*)

if *state* is a chance node then
return average of *EXPECTMINIMAX* value of *SUCCESSORS*(*state*)
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible

\[
\begin{align*}
\text{Node} & \quad \text{Value} \\
\text{Top} & \quad [\,-\infty, +\infty\,] \\
\text{Left} & \quad [\,-\infty, +\infty\,] \quad \text{Weight: 0.5} \\
\text{Right} & \quad [\,-\infty, +\infty\,] \quad \text{Weight: 0.5}
\end{align*}
\]
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible
A version of $\alpha$-$\beta$ pruning is possible
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible

B. Beckert: Einführung in die KI / KI für IM – p.25
Pruning in Nondeterministic Game Trees

A version of $\alpha$-$\beta$ pruning is possible
A version of $\alpha$-$\beta$ pruning is possible
Pruning Continued

More pruning occurs if we can bound the leaf values
Pruning Continued

More pruning occurs if we can bound the leaf values.
Pruning Continued

More pruning occurs if we can bound the leaf values
Pruning Continued

More pruning occurs if we can bound the leaf values
More pruning occurs if we can bound the leaf values.
Pruning Continued

More pruning occurs if we can bound the leaf values

B. Beckert: Einführung in die KI / KI für IM – p.26
Nondeterministic Games in Practice

Problem

$\alpha$-$\beta$ pruning is much less effective

Dice rolls increase $b$

21 possible rolls with 2 dice

Backgammon

$\approx 20$ legal moves

\[
\text{depth 4} = 20^4 \times 21^3 \approx 1.2 \times 10^9
\]

TDGammon

Uses depth-2 search + very good EVAL $\approx$ world-champion level
Digression: Exact Values DO Matter

MAX

DICE

MIN

Behaviour is preserved only by positive linear transformation of \( \text{EVAL} \)

Hence \( \text{EVAL} \) should be proportional to the expected payoff
Games of Imperfect Information

Typical examples

Card games: Bridge, poker, skat, etc.

Note

Like having one big dice roll at the beginning of the game
Games of Imperfect Information

Idea for computing best action

Compute the minimax value of each action in each deal, then choose the action with highest expected value over all deals.

Requires information on probability the different deals.

Special case

If an action is optimal for all deals, it’s optimal.

Bridge

GIB, current best bridge program, approximates this idea by

– generating 100 deals consistent with bidding information
– picking the action that wins most tricks on average
Commonsense Example

**Day 1**

Road A leads to a small heap of gold pieces  
10 points

Road B leads to a fork:
- take the left fork and you’ll find a mound of jewels  
100 points
- take the right fork and you’ll be run over by a bus  
-1000 points

Best action: Take road B  (100 points)

**Day 2**

Road A leads to a small heap of gold pieces  
10 points

Road B leads to a fork:
- take the left fork and you’ll be run over by a bus  
-1000 points
- take the right fork and you’ll find a mound of jewels  
100 points

Best action: Take road B  (100 points)
Commonsense Example

Day 3

Road A leads to a small heap of gold pieces (10 points)

Road B leads to a fork:
– guess correctly and you’ll find a mound of jewels 100 points
– guess incorrectly and you’ll be run over by a bus −1000 points

Best action: Take road A (10 points)

NOT: Take road B \( \left( \frac{-1000 + 100}{2} = -450 \text{ points} \right) \)
Proper Analysis

Note

Value of an action is NOT the average of values for actual states computed with perfect information.

With partial observability, value of an action depends on the information state the agent is in.

Leads to rational behaviors such as:

- Acting to obtain information
- Signalling to one’s partner
- Acting randomly to minimize information disclosure
Summary

- Games are to AI as grand prix racing is to automobile design
- Games are fun to work on (and dangerous)
- They illustrate several important points about AI
  - perfection is unattainable, must approximate
  - it is a good idea to think about what to think about
  - uncertainty constrains the assignment of values to states