Vorlesung
Grundlagen der Künstlichen Intelligenz

Reinhard Lafrenz / Prof. A. Knoll

Robotics and Embedded Systems
Department of Informatics – I6
Technische Universität München

www6.in.tum.de
lafrenz@in.tum.de
089-289-18136
Room 03.07.055

Wintersemester 2012/13

16.11.2012
Chapter 5 (3rd ed.)

Adversarial Search
Question:

- What is the difference between the 8-queen problem and playing chess?

- Reconsider the properties of different environments
  - Observability
  - Number of agents
  - Deterministic
  - etc.
Adversarial search

- Up to now: “Single-agent” problems
- Now: competitive environments, e.g. 2 agents competing
- Closely related to Game Theory

**Adversarial search:**
Typically 2-player zero-sum games

- Assumption: fully observable environments
- Examples: Chess, checkers

- Goal: Find optimal move (often with given time constraints)
Formal description of a game

- $S_0$: the initial state
- $\text{PLAYER}(s)$: player with next turn in a given state $s$
- $\text{ACTIONS}(s)$: set of possible moves
- $\text{RESULT}(s, a)$: transition model (result of a move)
- $\text{TERMINAL-TEST}(s)$: true, if “game over“, false otherwise
- $\text{UTILITY}(s, p)$: utility for a player $p$ if game ends in state $s$

$S_0$, $\text{ACTIONS}(s)$, $\text{RESULT}(s, a)$ define a search tree

Simple example: Tic Tac Toe

\[
\begin{array}{ccc}
X & O & X \\
O & O & X \\
X & X & O \\
\end{array}
\]
Tic Tac Toe search tree: 2 players, MIN and MAX

MAX (x)

MIN (o)

MAX (x)

MIN (o)

TERMINAL

Utility: -1 0 +1
Optimal decisions

Difference to classical search:
- Find a strategy, i.e. a sequence of optimal moves, taking the possible moves by the opponent into account

Definition of Minimax-function

\[
\text{MINIMAX}(s) = \begin{cases} 
\text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) = \text{true} \\
\max_{a \in \text{ACTIONS}(s)} \text{MINIMAX}((\text{RESULT}(s,a)), & \text{if PLAYER}(s)=\text{MAX} \\
\min_{a \in \text{ACTIONS}(s)} \text{MINIMAX}((\text{RESULT}(s,a)), & \text{if PLAYER}(s)=\text{MIN} 
\end{cases}
\]

In a leaf (=final state of the game), Minimax(s) is the utility, otherwise the best value for the given player
Optimal decisions

Propagation of $\text{MAXIMAX}$ values from the leaves to the root
**Alpha-Beta Pruning**

B tries to minimize, therefore set value 3 as the maximum

Subtree expanded, No lower value, set minimum

Last node on next level expanded, able to set max

12 > 3, therefore the value remains

2 < 3, therefore the A would not choose this move
**Alpha-Beta Pruning**

The **Alpha-Beta Pruning** algorithm is an optimization technique for the **MINIMAX** decision function.

**function** \( \text{MINIMAX-DECISION}(\text{state}) \) \text{ returns } \text{an action} \\
\text{return } \text{argmax}_{a \in \text{ACTIONS}(s)} \text{ MIN-VALUE(RESULT(\text{state}, a))} \\

**function** \( \text{MAX-VALUE}(\text{state}) \) \text{ returns } \text{an utility-value} \\
\text{if } \text{TERMINAL-TEST}(\text{state}) \text{ then return } \text{UTILITY}(\text{state}) \\
\quad v \leftarrow -\infty \\
\text{foreach } a \text{ in } \text{ACTIONS}(\text{state}) \text{ do} \\
\quad v \leftarrow \text{MAX}(v, \text{MIN-VALUE(RESULT(\text{state}, a)))} \\
\text{return } v \\

**function** \( \text{MIN-VALUE}(\text{state}) \) \langle \text{same structure} \rangle
Incomplete real-time decisions

(a) White to move

(b) White to move
Incomplete real-time decisions

\[ H{\text{-MINIMAX}}(s,d) = \begin{cases} 
  \text{EVAL}(s) & \text{if CUTOFF-TEST}(s,d) = \text{true} \\
  \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s,a),d+1), & \text{if PLAYER}(s) = \text{MAX} \\
  \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s,a),d+1), & \text{if PLAYER}(s) = \text{MIN} 
\end{cases} \]

Evaluation function:

- Expectation values for UTILITY based on statistical data
  - Too many categories, not manageable
- Assumptions, e.g. „value“ of chessmen
  - Pawn: 1, Knight/bishop: 3, rook: 5, queen: 9
- Weighted linear function
Summary

- Adversarial search needs to take all possible moves of the opponent into account.
- Maximise your strategy based on the assumption that the opponent acts optimally.
- Alpha-beta pruning can reduce the search space.
- Incomplete real-time decisions need evaluation functions.
Questions

Consider the travelling salesman problem (TSP)

- Minimal spanning tree (MST) heuristic: If partial tour already exists, MST of a subset of cities is the minimal sum of the connection cost of any tree connecting these cities

a) Show that this heuristic can be derived from a relaxed version of the TSP

b) Show that this heuristic dominates the straight-line distance
c) Are there efficient ways to solve the MST problem?
Questions

Consider the following game

A moves first
- Each player must move to empty field in either direction
- If field occupied, the player can jump over the opponent, if there is a field
- Goal: reach the start field of the opponent
- Utility function for A: +1, if A reaches field 4 first
  - 1, if B reaches field 1 first
Questions

a) Draw the game tree using state (field for A, field for B)
   - terminal states drawn as squares, utility values in a circle
   - loop states drawn as double squares, their values are “?”

b) Mark the nodes with their MINIMAX values
   How are the “?” handled? Why?

c) Why does the Minimax-algorithms fail for that tree?
   How can the tree be repaired?
Questions

a) How many Tic Tac Toe games are possible?

b) Draw a tree with 2 levels and show the utility values at level 2

c) Use the minimax algorithm to choose the best initial move

d) Show the nodes *not* expanded because of pruning-