What is this lecture about?

- Games - problem definition
- The Minimax algorithm
- Alpha-beta search
- Heuristic extensions
- State of the art game programs

Games Defined as Search Problems for Agents

Difference to previous lecture: now we have more than one agent that have different goals.

- All possible game sequences are represented in a game tree.
- The nodes are the states of the game, e.g. board position in chess.
- Initial state and terminal nodes.
- States are connected if there is a legal move/ply.
- Utility function (payoff function)
- Terminal nodes have utility values 0, 1 or -1.

Example: Tic-tac-toe, Russel-Norvig p163, figure 5.1.

Which Kind of Games?

Deterministic two-player zero-sum game with perfect information, i.e. the current state (position) of the game is completely known to both players. The players act always rational.

Examples: Chess, Go, 4-in-a-row, etc.
**Properties of Alpha-Beta Search**

Good news:
- The number of nodes/game states investigated until a solution is found is $O(b^{m/2})$ instead of $O(b^m)$ for a suitable ordering.
- Effective branching factor is $\sqrt{b}$, i.e. now we can search twice as many plies!

Not so good news:
- Still far from grandmaster level . . .

What if we skip uninteresting branches? How to identify those?

**Strategies for Two-Player Games**

Given two players called MAX and MIN, MAX wants to maximize the utility value.
Since MIN wants to minimize the same value, MAX should choose the alternative that maximizes given that MIN minimized.

**MINIMAX-algorithm**

$$
\text{MINIMAX}(s) =
\begin{cases}
\text{Utility}(s) & \text{if Terminal-Test}(s) \\
\max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{Result}(s,a)) & \text{if Player is MAX} \\
\min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{Result}(s,a)) & \text{if Player is MIN}
\end{cases}
$$

Example: Russel-Norvig p164, fig 5.2; algorithm p166, fig 5.3.

**Alpha-Beta Search**

Suppose, we reach a node $t$ in the game tree which has leaves $t_1, \ldots, t_k$ corresponding to moves of player MIN.
Let $\alpha$ be the best value of a position on a path from the root node to $t$.
Then, if any of the leaves evaluates to $f(t_i) \leq \alpha$, then we can discard $t$, because any further evaluation will not improve the value of $t$.
Analogously, define $\beta$ values for evaluating response moves of MAX.

Example: Russel-Norvig p168, fig 5.5.
An Evaluation Function for Chess

\[ f(s) = \min \{ p_1 M(s), p_2 C(s), p_3 P(s) \} \]
\[ f(s) = p_1 M(s) + p_2 C(s) + p_3 P(s) \]

where \( p_1, \ldots, p_3 \) are some weights, and

- \( M(s) \) is the material value over all pieces, e.g. pawn=1, knight=3, bishop=3, rook=5, queen=9, negative values for black pieces,
- \( C(s) \) is weighted sum over all controlled fields (high weight for important fields), minus the corresponding value for black,
- \( P(s) \) is a value for the pawn structure of position \( s \) minus the corresponding value for black.

One white move in state \( s \)

- Search depth = 3 plies:
  \[ \text{move} = \arg \max_{i=1}^{k} \min_{j=1}^{k_i} \max_{l=1}^{k_{ij}} f(s_{i_1}) \]

...  

- Search depth = \( d \) plies:
  \[ \text{move} = \arg \max_{i=1}^{k} \min_{j=1}^{k_i} \max_{w=1}^{k_{iw}} f(s_{k_1 \ldots k_w}) \]
**A Stochastic Game: Backgammon**

In Backgammon, the dice rolls introduce uncertainty. This can be modelled by *chance nodes* in the game tree.

Instead of minimax (or alpha-beta) values, use *expected values* according to the probability of the dice rolls.

Fig 5.10, Russel-Norvig p177.

Example: Game tree for backgammon, Russel-Norvig p163, fig 5.1.

---

**Further Questions**

- What happens when initial conditions like 'the players always act rational' is dropped?
- Extensions to games with more than 2 players.
- Humans often use goal directed planning.
- See lecture about Automated Planning (AP), a combination of search and reasoning with logic.

---

**State of the Art Game Programs**

- DEEP BLUE (IBM) 1997 beats chess world champion Garry Kasparov. Modern chess programs: HYDRA and RYBKA.
- LOGISTELLO (Othello/Reversi) beats world champion in 1997.
- CHINNOK 2007 plays checkers perfectly (endgame table in combination with forward alpha-beta search).
- TD-GAMMON plays well against expert backgammon players.
- BridgeBaron based on e.g. planning and Monte Carlo simulation play well against humans (no bidding)
- MoGo plays Go at amateur level (main problem: branching factor ~ 360)
- ...

---

**Further Tricks**

- Find a good *move-ordering*.
- Limiting the search depth: cut off the search tree at a certain depth $d$ and apply a heuristic *evaluation function*. Maybe in combination with iterative deepening.
- *Cut-off test*: stop searching when $f(s)$ stops fluctuating heavily (*quiescence*).
- Handle the *horizon problem*: Tendency of computer players to move bad states behind the search depth (horizon) by the use of effect-less moves.
- Opening books and end game tables.