Playing Chess

A dive into algorithmic game playing

René Mellema

WASP: Mathematical foundations of AI cluter

UMEÅ UNIVERSITY

Historical positioning
- AI
- Chess computers

Intro to Game Theory
- Extended form games
- But what about Chess?

Game Theory Algorithms

Chess computer

Conclusion

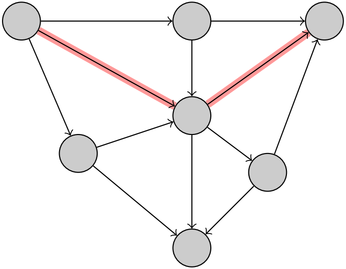
The early days

- AI “started” in 1956
  - Dartmouth Conference
- High level reasoning seen as the goal
  - Initial success of Logic Theorist
  - Focus on symbolic systems
- Lots of sceptics
  - “A machine can never do X!”
  - Focus on showing that machines can do X

    $\Rightarrow$ Microworlds/Toy problems

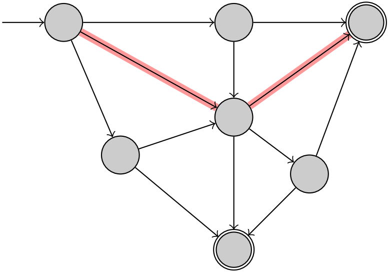
Reasoning as search

- Every problem occurs in some state space
- There are starting states and ending states
- States can be moved between (linked)
- Find a route from current state to an end state



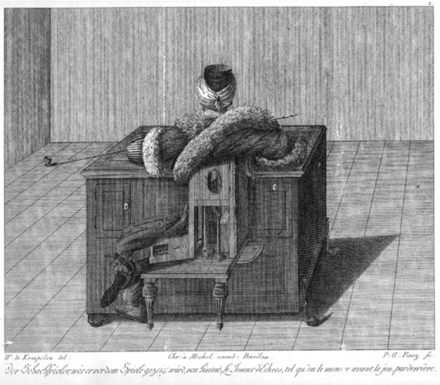
Reasoning as search

- General problem solving approach
  - Proofs
  - Route planning
  - Action selection
- Paths/states can lead to combinatorial explosion
- Pruning of the search tree necessary



Why Chess?

- Long interest in “solving” chess
  - Mechanical Turk (1769)
  - El Ajedrecista (1912)
  - Interest from many scholars
    - Norbert Wiener
    - Claude Shannon
    - Alan Turing
    - John McCarthy
    - ...
- Seen as a “cognitive”/hard game
  - Only smart people allowed



Game Theory

- Mathematical study of interaction among agents
- Agents are independent and self-interested
- Interactions are studied as “games”
  - Agents pick actions
  - Get pay-off (utility) based on all actions chosen.

Normal form game

- Games are defined by their pay-off matrices

$$A_i \,:\, \text{Actions for player } i$$
$$A = A_i \times \cdots \times A_n$$
$$u_i \,:\, A \longrightarrow \mathbb{R}$$

- A strategy is an action for a player

	S	H
S	3 / 3	0 / 1
H	1 / 0	1 / 1

Stag and Hare game

	C	D
C	-1 / -1	-5 / 0
D	0 / 5	-3 / -3

Prisoners dilemma

Zero sum games

- Games with one winner and one loser

$$\text{for all } a \in A_1 \times A_2 : u_1(a) = -u_2(a)$$

- Solution concept:
  - Pick the action that maximises utility
  - Whatever the other agent does!

$$\arg\max_{a_i \in A_i} \min_{a_j \in A_j} u_i(a_i, a_j)$$

  - Called the maxmin value
  - Coincides with minmax value

$$\arg\min_{a_i \in A_i} \max_{a_j \in A_j} u_j(a_i, a_j)$$

	H	T
H	1, -1	-1, 1
T	-1, 1	1, -1

Matching pennies

Rock, paper, scissors

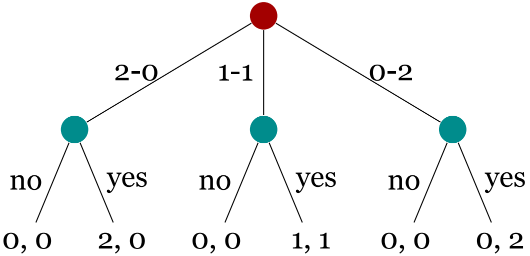
	R	P	S
R	0 / 0	1 / -1	-1 / 1
P	-1 / 1	0 / 0	1 / -1
S	1 / -1	-1 / 1	0 / 0

Rock, paper, scissors

Rock, paper, scissors, lizard, Spock is also a zero-sum game

Extended form games

- What if there are more actions?
- Extensive form games
- Strategies give an action for each node
- Each action is called a ply
- Can be translated to normal form games
- For zero sum: can calculate maxmin/minmax values



What about Chess?

- Chess is zero sum
- Chess is extended form
- Using reasoning as search:
  - Build extended form tree
  - Calculate minmax strategy

Minimax algorithm

- Calculates minmax values/strategies
- Works on extended form games
  - Tree can be either given or generated
- Basis for all zero-sum game algorithms
  - Also has non-deterministic extensions
- General form is called Backward Induction

Minimax Algorithm

Function Minimax-Decision(state)

return $\arg\max_{a \in \texttt{Actions}(state)} \texttt{Min-value}(\texttt{Result}(state, a))$

Function Min-value(state)

if $\texttt{Terminal-test}(state)$ **then return** $\texttt{Utility}(state)$;
$\texttt{v} \leftarrow \infty$;
for $a \in \texttt{Actions}(state)$ **do**
  $\texttt{v} \leftarrow \min(\texttt{v}, \texttt{Max-value}(\texttt{Result}(state, a)))$
return $\texttt{v}$

Function Max-value(state)

if $\texttt{Terminal-test}(state)$ **then return** $\texttt{Utility}(state)$;
$\texttt{v} \leftarrow -\infty$;
for $a \in \texttt{Actions}(state)$ **do**
  $\texttt{v} \leftarrow \max(\texttt{v}, \texttt{Min-value}(\texttt{Result}(state, a)))$
return $\texttt{v}$

Alpha-Beta pruning

- Minimax is not very efficient
- Can we do better?
- Alpha-beta pruning can cut away half the tree
- Opponent will never go to nodes better for you
- If value opponent can force you to is lower than best value you can get so far, ignore whole branch
- Order of actions matters
  - Can use iterative deeping to order moves[1]

[1] Requires a heuristic over how good an action is

Alpha-Beta pruning

Function Alpha-Beta-Decision(state)

```
v ← Max-value(state, -∞, ∞);
return a ∈ Actions(state) with value v;
```

Function Max-value(state, α, β)

```
if Terminal-test(state) then return Utility(state);
v ← −∞;
for a ∈ Actions(state) do
    v ← max(v, Min-value(Result(state, a), α, β));
    if v ≥ β then return v;
    α ← max(α, v);
return v
```

What if the tree is to deep?

- Evaluating whole tree might not be feasible
- Cut off search at a certain depth
- Instead of utility, use a heuristic
  - Heuristic calculates expected pay-off for a state
  - Heuristic depends on game
- Do iterative deeping until turn time runs out
- No longer guaranteed to give the right move!
  - Horizon effect

Other improvements

- States can occur multiple times
  - Transposition table
  - Hash function maps similar states on each other
  - Hash map stores v for state, action pairs
  - Can again double performance
- What if we don't *need* to be correct?
  - Prune branches likely not between α and β
  - Logistello using ProbCut beats regular 64% of time
- Only look at a clearly better option, the Singular Extension
  - When depth limit is reached, try singular extension

Can we now play chess?

Chess is hard

- Algorithms are general
- Chess specific alterations are needed
- Chess openings and endgames are well understood
  - Simply store the best actions for begin/end game states
- Good evaluation function for states needed
- Deep Blue et al run on specialized hardware

Deep Blue

- Defeated Garry Kasparov in 1997
- Ran parallel Alpha-Beta Search on:
  - 30 IBM RS/6000 processors
  - 480 custom VLSI chess processors
- Evaluated 30 billion positions per move
- Regularly reached depth 14
  - Could hit 40
- 8000 features in the evaluation heuristic
- "Opening book" with 4000 positions
- End game table for up to 6 pieces

Take home message

- Reasoning as search is a powerful paradigm
  - Works well for “cognitive” problems
- Needs specific alteration tailored to problem domain
- Chess is not yet solved, but hardware can change that

References

- Essential of Game Theory, Leyton-Brown and Shoham (2008)
- Adversarial search, in Artificial Intelligence: A Modern Approach, Russel and Norvig (2010)
- Wikipedia