Algorithmic Game Theory

Henrik Tscherny

23. Juli 2023

Inhaltsverzeichnis

1 Random Definitions

- **weakly solved**: outcome if both players play perfect
- **strongly solved**: entire game tree/graph is known

2 Non-Cooperative-Games

Normalform $G = (P, S, u)$

- Players: $P = \{1, 2, ..., n\}$

- Strategies: $S = (S_1, S_2, S_n)$
- Utility functions: $u = (u_1, u_2, ..., u_n)$ with $u_i : S \to \mathbb{R}$

Note: Only **one decision** is made (choice of strategy) and moves are **simultaneous**
Strategy Profile $s = (s_1, s_2, ...s_n) \in S_1 \times S_2 \times ... \times S_n = S$
Weakly Dominant Strategies ($\leq$**)**: The strategy has the best payoff regardless of the choices of the other players.
Strictly Dominant Strategies ($<$**)**: The strategy has the greatest payoff among all other strategies. (there can only be at most one such strategy)

2.1 pure and mixed strategies

pure strategies:
every player chooses the same action in a given situation (like a spreadsheet on what to do). Each move is deterministic. Example: Play Paper every time in Rock-Paper-Scissors

mixed strategies:
use probabilities to determine a pure strategy for each situation. Each move is non-deterministic. Example: choose Rock, Paper, Scissors with p=1/3 each.

- Models that players confuse each other by switching
- adds uncertainty about actions of others
- models repeated play and population dynamic

2.2 Pareto

- **s weakly Parteo-dominates t**: $\forall i \in I : u_i(s) \geq u_i(t)$.
  payoff of s is at least as good as for t for each utility function
- **s Pareto-dominates t**: $\exists j \in I : u_j(s) > u_j(t)$.
  payoff of s is better than for t for a given utility function
- **s strongly Pareto-dominates t**: $\forall i \in I : u_i(s) > u_i(t)$.
  payof of s is better than for t for all utility functions
- **t is Pareto-optimal**: $\neg \exists s \in S :$ s Pareto-dominates t
- **t is weakly Pareto-optimal**: $\neg \exists s \in S :$ s stongly Pareto-dominates t

In a Pareto optimum no Player can gain switch strategies without another player being worse off
Example: (S, S), (S, C), (C, S) are Pareto-optimas for the prisoners dilemma.
(S, S) strongly dominates (C, C).
All strategies in rock paper scissors are pareto-optimal
Best Response: A Strategy is a best response to another strategy iff it produces the best payoff for the player given the strategies of the other players
Note: dominant strategy $\rightarrow$ best response but not $\leftarrow$

2.3 Nash

pure:

- **Nash equilibrium in pure strategies**: $\forall i \in I : s_i$ is a best response to s_{-1}
- **strict Nash equilibrium in pure strategies**: $\exists! s : s$ is a best response to s_{-1}
  s_{-i} is the strategy profile s without the strategy of player i

The prisoners dilemma has (C,C) as the single pure Nash equilibrium where every player plays his dominant strategies.
There can be none, one or multiple such equilibria
Note: The pure Nash equilibria can be computed in **PTIME** by exhaustive search for best response

2.3.1 Nashs Theorem

For a non-cooperative game G with finite Players, finite Strategies there exists a Nash equilibrium in mixed strategies
Proof sketch:

- Model strategies as unit vectors in $\mathbb{R}^{|S|}$
- Mixed strategies are points on a simplex in this vector space
- define some function $f : \Pi \rightarrow \Pi$ (Π is the set of all mixed strategy profiles)
- use Brouwers fixpoint theorem to show that f has at least one fixpoint

2.3.2 Computation

- translate problem intro mixed integer programming
- solve MIP (NP-complete)

- Given a Nash equilibrium for a game, finding the next equilibrium is FNP-compete
  - Functional Complexity: Functions that given input x can be compute some y in some Time/Space
  - FP and FNP are functional P and NP

2.4 Prisoners Dilemma

- $P = \{1, 2\}$
- $S_1 = S_2 = \{S, C\}$
- $S = \{(S, S), (S, C), (C, S), (C, C)\}$
- $u_1 = \{(S, S) \mapsto 4, (S, C) \mapsto 0, (C, S) \mapsto 5, (C, C) \mapsto 3\}$
- $u_2 = \{(S, S) \mapsto 4, (S, C) \mapsto 5, (C, S) \mapsto 0, (C, C) \mapsto 3\}$

3 Sequential Games

- Several decisions
- Strategies can be seen as advice
- Players move sequentially
- can be represented using a Graph (maybe even Tree) (e.g. in Chess board state diagram)

Complete Information: All players know possible moves and utilities of all players

3.1 Zermelos Theorem

Every finite sequential game tree has a solution (strategy profile and utility) that can be obtained by backward induction
Backwards Induction

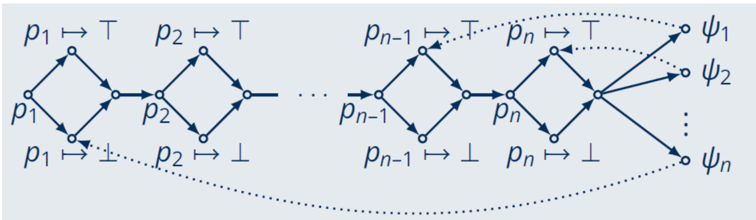
- start at the last round of the game (only one decision is left)
- determine for each of those nodes (in this depth) the best choice for the current player (and delete the rest)
- move one layer up and repeat till you reach root (only one path is left)

4 Combinatorial Games

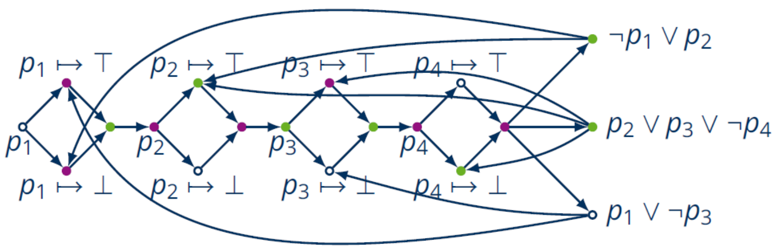
Can be solved using backwards induction

- Two Players
- Either One Player wins (1) and the other loses (-1) or its a draw (0)
- zero-sum-game

Geography is PSPace-complete

- $\in PSapce$: use DFS on game tree and only keep current path in memory (depth is poly)
- $hardness$: by reduction to TrueQBF:
  - $\varphi = \exists p_1 \forall p_2 \cdots .\psi$ with $\psi = \psi_1 \wedge \cdots \wedge \psi_m$
  - 

  - 'diamonds' select for each prop-var if its true/false and in the end formula gets checked by edges to diamond nodes
  - Example:

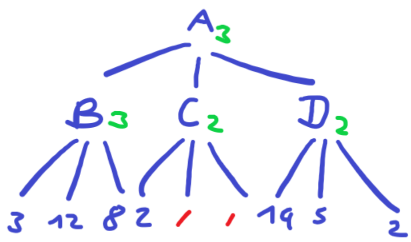


5 Minimax Tree Search

used to solve zero-sum games

- there is a min and a max player
- if min players turn select child node with min value (analogous for max)

- min and max switch in each layer
- Complexity: $O(b^d)$ (b: branching factor, d: depth), hence its impractical for larger games
- $\rightarrow$ solution: alpha beta pruning:
  - stop search when when we know that a nodes does not gets visited anyway
  - Example:



  - because B is already 3 we can stop at C=2 because Max does not care anyway
  - we have to search all of D because D=2 is the right most element (unlucky)
  - better heuristic helps
  - Worst-case complexity: $O(b^d)$ (all nodes have to be searched (because they are on the right))
  - Best-case complexity: $O(\sqrt{b}^d)$ (left nodes terminates search already)