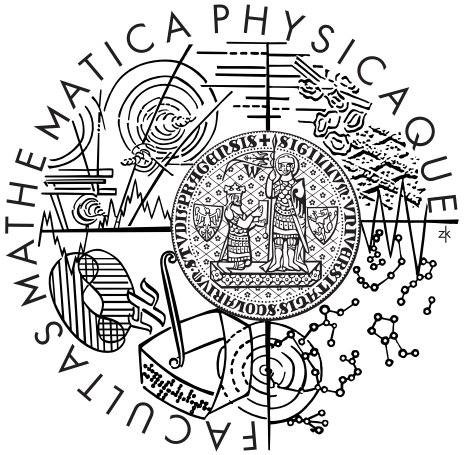
Charles University in Prague
Faculty of Mathematics and Physics

**MASTER THESIS**




Matej Moravčík

# Evaluating public state space abstractions in extensive form games with an application in poker

Department of Applied Mathematics

Supervisor of the master thesis: Milan Hladík
Study programme: Informatics
Specialization: Discrete models and algorithms

Prague 2014

I would like to thank my supervisor, Mgr. Milan Hladík, Ph.D. He supported me and provided helpful comments. He was also very understanding as for the homework duties for his classes, especially when me and my roommate were working late hours to make our agent for the competition before the submission deadline.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date .....  signature of the author

Název práce: Vyhodnotenie abstrakcií určených pre hry s neúplnou informáciou s využitím v pokeri

Autor: Matej Moravčík

Katedra: Katedra aplikované matematiky

Vedoucí diplomové práce: Mgr. Milan Hladík, Ph.D, Katedra aplikované matematiky

Abstrakt: Poznáme efektívne techniky na výpočet optimálnej stratégie pre hry v rozšírenej forme. Niektoré problémy, napríklad poker, sú stále omnoho väčšie, ako sú tieto techniky schopné zvládnuť. Riešením je vytvoriť abstrakciu hry, ktorá je menšia ako pôvodná hra. V tejto abstrakcii už dokážeme nájsť optimálnu stratégiu. Túto stratégiu môžeme potom využiť v originálnej hre. V tejto práci opisujeme techniky, ktoré sa na tvorbu abstrakcií aktuálne používajú. Väčšina z nich neberie osobite v úvahu informácie, ktoré sú viditeľné pre všetkých hráčov v hre. My sme na tento účel vyvinuli vlastnú techniku a otestovali sme ju v pokeri. Naše experimentálne výsledky ukázali, že nová technika priniesla značné zlepšenie oproti doteraz používaným technikám.

Klíčová slova: Hry v rozšírenej forme, Poker, Abstrakcia hry, Nashovo equilibrium

Title: Evaluating public state space abstractions in extensive form games with an application in poker

Author: Matej Moravčík

Department: Department of Applied Mathematics

Supervisor: Mgr. Milan Hladík, Ph.D, Department of Applied Mathematics

Abstract: Efficient algorithms exist for finding optimal strategies in extensive-form games. However human scale problems, such as poker, are typically so large that computation of these strategies remain infeasible with current technology. State space abstraction techniques allow us to derive a smaller abstract game, in which an optimal strategy can be computed and then used in the real game. This thesis introduces state of the art abstraction techniques. Most of these techniques do not deal with public information. We present a new automatic public state space abstraction technique. We examine the quality of this technique in the domain of poker. Our experimental results show that the new technique brings significant performance improvement.

Keywords: Extensive form games, State space abstraction, Public information, Nash equilibrium, Automatic abstraction technique

# Contents

# Introduction

## 0.1 Extensive games with imperfect information as a research topic

Game theory models games such as chess or card games, as well as real life scenarios from field of economics, political science, information security, and from many other fields.

One of the main goals of artificial intelligence research are autonomous computer agents capable of decision making in the real world environment such as the ones we have already mentioned.

Historically, computer chess has been a traditional domain for evaluation of artificial intelligence progress for many years. Big progress has been achieved in this area, but deterministic games without hidden information, such as chess or go, are often poor models for real world situations.

Many assumptions that are true in these games are rare in real world settings, mainly an assumption of perfect information and deterministic nature of the game.

There has been big progress in the area of imperfect information and stochastic games in last few years. This field of research is gaining more and more attention of artificial intelligence community.

## 0.2 Computer poker

Computer poker is a very good testbed for AI, designed to deal with imperfect information. It models several properties that are common for the real world problems:

- Imperfect information -in poker, a player does not know what card an opponent holds. Prefect information is also rare in the real word problems.
- Stochastic events - in poker, cards are dealt randomly from the deck. Random events are also very common in the real word problems.
- Quantification of winnings - in chess, player's goal is just to win the game. In contrast, a player always wants to win as much chips as possible in poker. Expected value of the profit is also very important in the real world problems, especially in those where money is involved.

There are also practical reasons, why computer poker is a good testbed:

- There is a number of strong computer players available to play against, and there is also a number of skilled human experts. This allows us to evaluate new game-play and opponent-modelling techniques.
- The game is too complex to be solved by brute force. Therefore, many smart AI techniques have been developed for poker domain, and there is still a lot of space for new ones.

- Computer chess programs can beat top human players by a large margin. In contrast, computer poker agents have beaten human players only in heads-up poker with limited bets [11]. Humans are considered to be stronger players in all other commonly played types of poker. Therefore, there is still a lot of room for improvement in computer poker.

## 0.3 The Annual Computer Poker Competition

In order to compare word best computer poker agents, the **Annual Computer Poker Competition (ACPC)** [1] has run each year since 2006. The event attracts amateur competitors, as well as academic researchers from countries all around the world.

Currently, in 2014, 34 agents from 20 teams were submitted to the competition, and four types of poker games were played: Heads-Up Limit Texas Hold'em, Heads-Up No-Limit Texas Hold'em, 3-player Limit Texas Hold'em and 3-player Kuhn.

Most agents were submitted to Heads-Up No-Limit competition. This competition is also the most interesting one from the point of game abstraction, and we have focused on it.

As a result of techniques introduced by this thesis, we created our agent for 2014 competition.

## 0.4 Game abstraction

New game solution techniques, capable of finding optimal strategies in large games, were introduced in recent years [27, 7].

However, real human scale problems are often several magnitudes larger that problems solvable with state of the art techniques. For example, No-limit Texas Hold'em Poker, played at ACPC competition has approximately $6.3 \times 10^{164}$ game states [12]. It is very unlikely that it will ever be possible to solve such a big games directly.

The technique of game abstraction was introduced to deal with large games. Firstly, new smaller abstract game is created. Then some solution technique is used to find good strategy in this new game. Finally, there have to be translation techniques to translate information from real game to the abstract game, and to translate action made in the abstract game back to the original game.

Game abstraction should have strategical properties as similar with the original game as possible.

In two players zero sum games, such as No-limit Texas Hold'em Poker played at ACPC, solution techniques are good explored, and the main difference of the top agents performance is caused by different quality of the game abstractions.

### 0.4.1 Lossy and lossless abstractions

In an ideal case, we can create an abstract game with a property that any optimal strategy in this game corresponds to optimal strategy in original, unabstracted

game. This type of abstraction is called losseless. With this approach Rhode Island Hold'em Poker - large, but still artificial game, was solved [8].

Loosless game abstractions are still too large for most human scale problems. Therefore, lossy abstractions are used. Strategy in these abstractions can have different performance as the equivalent strategy in the original game. When we are creating such an abstraction, our main goal should be to minimize this difference.

### 0.4.2 Action space abstraction

To create small abstract game from game with many possible player actions, we can use the action space abstraction. In this type of abstraction, players are allowed to play only subset of the actions available in the original game.

For example, in abstracted No-limit Texas Hold'em Poker, players are allowed to choose only from few distinct bet sizes instead of betting any legal amount of chips.

### 0.4.3 State space abstraction

Another approach how to create a game abstraction is to force players to play in the same way when the information about the game state differs only slightly.

For example, in Texas Hold'em Poker abstraction, a player wold not be allowed to distinguish between A♠A♡ and K♠K♡, he would be forced to play both combinations in the same way.

For smaller games, like a limit versions of poker, state space abstraction is often sufficient. For large games with many possible actions, like a No-limit Texas Hold'em Poker, combination of the state space and action space abstraction has to be used.

### 0.4.4 Public information

Every information about the game state is available only to certain subset of players. Information available only to one player is called private information, information available to all players public information.

In Texas Hold'em Poker, information about player's hand is private, and information about cards dealt on the board is public.

Traditional computer poker agents used abstractions that capture only information about player's hand, and did not deal with information about board cards [11, 7].

Current state of the art agents use abstractions capable to capture also information about the board cards [1]. Some of the public state space abstractions have to be used for this purpose.

These public state space abstractions were handcrafted by human experts what brings several drawbacks:

- Quality of abstraction is very dependent on quality and judgment of human expert. Therefore, this approach is very domain specific.

- Creating any new abstraction take a lot of effort. This makes changing the size of the abstraction very problematic.
- Modern clustering algorithms can handle much more data than human brain. Therefore, we believe it its possible to create higher quality abstractions for complex games automatically.

## 0.5 Our goals

Our main goal was to create an algorithm for automatic public state space abstraction. Then, we used this algorithm as one of the key components for the whole game abstraction.

Domain of No-limit Texas Hold'em Poker was used to evaluate our new approach. We compared it's performance with older abstraction techniques.

Finally, we used our new abstraction for the ACPC 2014 competitor, Nyx 2014.

# 1. Background

## 1.1 Texas Hold'em Poker

Poker games are a large family of card games, involving betting. The winner of the game is determined by his card combination. **No-limit Texas Hold'em Poker** is the most often played version of poker worldwide. We focused on two player version of this game. This version is also played at the ACPC competition.

## 1.2 Rules of No-limit Texas Hold'em Poker

Single play of poker game with single deal of cards to players is called hand. Poker match typically consists of a large number of hands. For example, one match consists of 3000 hands at ACPC competition, and multiple matches are played to determine the winner.

At the beginning of the hand, each player has some amount of chips called **stack**. To **bet** some amount in poker means to move chips from player stack to pile of chips in the middle of the table called the **pot**.

### 1.2.1 Play of single hand

#### Start of the hand

Before the hand starts, the players have to post mandatory bets called blinds. In two player version of poker, the player on the dealer position posts the **small blind** and the other player posts the **big blind.** After that, each player obtains two cards from standard 52-card deck. These cards are known only to the player who is holding them, and they are called the **private hand**.

#### Betting rounds

A hand consists of four betting rounds. At the begging of each betting round, except the first one, cards visible for all players are dealt from the deck. Three cards are dealt for the second betting round and one card for the third and the fourth. These cards are called the **board cards**.

Players alternate in their actions until the betting round ends. They can place **bet** if there was not previous bet during actual round, they can **raise** the opponent bet if the bet is not larger than their stack, they can **call** the opponent bet, or they can give up and **fold** their hand.

The player on the button plays first on the first round, the order is then switched on later rounds.

Betting round continues until at least one of the players folds, or until both players call opponent bet.

#### End of the game

When one player folds, hand ends and the other player wins all chips in the pot. If neither player folds his cards until the end of the fourth betting round, there

is the **showdown**.

Players can use all five cards dealt on the table and combine them with two private cards to make the best possible five card combination called the **hand** .

Player with stronger hand wins the pot. When both players have combinations with equal strength, there is a tie, and the pot is split equally between them.

The goal of the game is to win as much chips as possible. Lost chips are removed from stacks at the end of the hand in common settings, but at ACPC stacks remain the same during the whole match.

## 1.3 Extensive game

Traditional representation of a game in game theory is the **normal form**, where players act simultaneously. This assumption does not hold for most of the games played by humans, such as chess, go or poker. Better model for games with sequential moves is the **extensive game**. It models single game play like a sequence of chance and players' actions called history. The whole game is a set of all possible histories.

Formally, an extensive form game consists of[19, p. 200]:

- A finite set $N$ (the set of **players**).
- A finite set $H$ of sequences. Each member of $H$ is a **history**, each component of a history is an **action**. The empty sequence is in $H$, and every prefix of a history is also a history $((h, a) \in H \implies (h \in H))$. $h \sqsubseteq h'$ denotes that $h$ is a prefix of $h'$. $Z \subseteq H$ are the terminal histories (they are not a prefix of any other history).
- The set of actions available after every non-terminal history $A(h) = \{a : (h, a) \in H\}$.
- A function $p$ that assigns to each non-terminal history an **acting player** (member of $N \cup c$, where $c$ stands for chance).
- A function $f_c$ that associates with every history for which $p(h) = c$ a probability measure on $A(h)$. Each such probability measure is independent of every other such measure.
- For each player $i \in N$, a partition $\mathcal{I}_i$ of $h \in H : p(h) = i$. $\mathcal{I}_i$ is the **information partition** of player i. A set $I_i \in \mathcal{I}_i$ is an **information set** of player $i$.
- For each player $i \in N$ an **utility function** $u_i : Z \to \mathbb{R}$.

Standard information partition of the game is to have each information set for a player containing only histories which he cannot differ by events in the game that are known to him. During the game, a player knows only information set, he is currently in, and not particular history within the information set. Therefore, he has to play in the same way, no matter history he is currently in.

### 1.3.1 Poker example

In poker, history is determined by betting sequence, the cards dealt to the players and the board cards.

Information set for the player consists of all histories distinct only in opponent's private cards. All histories in the player's information set share betting sequence and his hole cards.

Actions available to a player in his information set are all legal betting actions determined by betting sequence that leads to the set. These can be to fold, to call, or to bet/raise any legal amount.

Terminal histories are those, where the game ends, that means one player folds or there is a showdown. Utility for a player in some terminal history is the amount of chips won after a game play determined by this history.

### 1.3.2 Constant sum games

Game is **constant sum**, when for each terminal history, sum of utilities of all players in this history is some constant $c$, i.e for all $h \in Z$ holds equation $\sum_{i=1}^{n} u_i(h) = c$.

The **zero sum game** is a special case of constant sum game, when $c = 0$. When we slightly modify the constant sum game by defining new utility function $u_i^{'}$ with property that $u_i^{'} = u_i - c/N$ we will obtain zero sum game. Most games played by humans, like chess, roshambo or poker, are zero sum games.

## 1.4 Strategy

Strategy of a player defines how he acts in his turn. In perfect information games like chess, player should make decision in order to take best possible single action. In contrast, when there is an imperfect information involved, a player could reveal additional information to the opponent by choosing the same action every time he is in the same history, thus his play would not be optimal any more. Good example of this phenomenon is a game of roshambo. If, for example, one player always plays rock, another player can take advantage of this additional information and always wins by playing paper. Therefore, players have to choose actions stochastically according to some probability distribution.

The behavior strategy of a player is defined as set of such distributions, where one distribution belongs to each his information set. The strategy profile of the game is a vector of strategies for all players.

Formally:

- A **strategy** for player $i$, $\sigma_i$, is a function that maps $I \in \mathcal{I}_i$ to a probability distribution over $A(I)$ and $\pi^{\sigma}(I, a)$ is the probability of action $a$. $\Sigma_i$ denotes the set of all strategies of player $i$.
- A **strategy profile** is a vector of strategies of all players, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{|N|})$. $\Sigma$ denotes the set of all strategy profiles.
- We denote $\pi^{\sigma}(h)$ as the probability of history $h$ occurring given the strategy profile $\sigma$.
- Let $\pi_i^{\sigma}(h)$ be the contribution of player $i$ to that probability. Then we can decompose $\pi^{\sigma}(h)$ as

$$\pi^{\sigma}(h) = \prod_{i \in N \cup c} \pi_i^{\sigma}(h)$$

- Let $\pi_{-i}(h)$ be the product of all players' contribution (including chance), except that of player $i$.
- Define $\sigma|_{I \to a}$ to be the same strategy profile as $\sigma$, except that a player always plays the action $a$ in the information set $I$.
- Define $u_i(\sigma)$ to be the expected utility for player i, given the strategic profile $\sigma$.

## 1.5 Perfect recall and imperfect recall

If a game satisfies the **perfect recall** , it guarantees that players neither forget any information revealed to them, nor the order in which the information was revealed. Otherwise we say that a game has **imperfect recall**.

Perfect recall is more natural model for games like poker, and has nice theoretical properties like an existence of optimal strategy. However, most of the state of the art abstractions in poker use imperfect recall, which allows much more compact game representation.

## 1.6 Solution techniques

Goal of the solution techniques is to find some good game strategy. There are basically two different approaches. One option is to model an opponent strategy and then use the best response to this strategy. The other approach is to try to find some approximation of the game theoretic optimal strategy.

### 1.6.1 Best response

If we know strategy of all opponents exactly, we can easily compute the best counter strategy. However, this is not typical setting for most of games. Instead, we can try to model opponent strategy according to actions he takes, and compute counter strategy to this model.

This approach is still problematic when we play large extensive games such as poker. Firstly, we usually have to play some strategy until the model is created, and we do not want to loose too much during this period. The other issue is that very large number of observations is needed to create an accurate model of opponents. Best response technique is very sensitive to model inaccuracy.

To counter these problems, multiple techniques, such as **restricted Nash response** [18], **data biased response** [15] or **implicit agent modelling** [3], were introduced.

Although these new techniques exhibit good results in the limit poker competitions, this does not hold for no-limit competitions, and top agents in these competitions try to play approximate Nash equilibrium strategy.

## 1.7 Nash equilibrium

Strategy profile, in which no player can improve by changing his strategy, i.e each player plays best response to other players' strategies, is called the Nash equilibrium. Formally:

A **Nash equilibrium** is a strategy profile $\sigma$ such that for any player $i \in N$,

$$u_i(\sigma) \geq \max_{\sigma_i^* \in \Sigma_i} u_i((\sigma_i^*, \sigma_{-i}))$$

It has been proved by Nash, that every finite game in normal form has such a strategy profile [19]. This theorem holds also for extensive games with perfect recall, and there is often more than one equilibrium.

### 1.7.1 Properties of Nash equilibrium in two player zero sum game

Nash equilibrium approximations are most often used in two players zero-sum games. Equilibrium profile has some nice properties in these games:

- Equilibria form a convex set, i.e any convex combination of two equilibria is also an equilibrium.
- If players $i$ plays optimally, i.e his strategy $\sigma_i$ is a part of some equilibrium strategy profile $\sigma$, and there is another equilibrium strategy profile $\sigma^{'}$, then also $(\sigma_i, \sigma^{'}_{-i})$ is the Nash equilibrium of the game. That means the player's strategy is optimal regardless to the opponent's strategy.

  This is very important property. When we want to find optimal strategy for some player, it is sufficient to compute any Nash equilibrium of the game and then pick player's strategy from that profile.
- Utility of first player's players optimal strategy is the same constant in any equilibrium profile. That means if we have two Nash equilibira $\sigma$, $\sigma^{'}$, then $u_1(\sigma) = u_1(\sigma^{'}) = v$ and consequently $u_2(\sigma) = u_2(\sigma^{'}) = -v$, where $v$ is some constant called **game value**.

  Moreover, if player strategy $\sigma_1$ is optimal ( there is some $\sigma_2$ with property that $(\sigma_1, \sigma_2)$ is an equilibrium), the player 1 has a guarantee, that he will obtain at least game value against any opponent's strategy. Any optimal strategy of player 2 also ensures that player 1 does not win more that game value with any strategy.

  Therefore, the first player can ensure that he will win at least game value, and the other player can ensure that he won't win more, and vice versa. Thus playing an equilibrium strategy is rational choice for both players.
- We can find equilibrium strategic profile in polynomial time.

These properties do not hold for the general sum games, or for the games with more than two players. However, our main area of interest is Heads-Up No-limit Texas Hold'em Poker, which is an instance of two players zero sum game.

### 1.7.2 Computation of Nash equilibrium

Generally, computation of the Nash Equilibrium belongs to class of problems called PPAD-complete, and it is believed that there is no polynomial time algorithm for this class [20], but there are polynomial algorithms for two players zero sum case.

### 1.7.3 Linear programming

One way how to compute optimal strategy for extensive game is by linear programming. Firstly, new representation of the game called sequence form is created. Sequence form represents game and players' strategies as a set of linear equations. Then, the linear program is created and we can find Nash equilibrium by solving this program.

If we use some efficient solution technique, such as the interior point method, we can obtain solution in polynomial time. It is easy to convert the result back to the extensive form.

Both, the linear program and the sequence form representation of game, need memory linear to the count of possible histories. This could seem efficient, but for large games such as poker, this bound is still too big.

We can exploit the fact, that the count of the information sets if often much smaller than the count of possible histories. There are strong approximation algorithms that use only memory linear to the count of information sets in the game.

### 1.7.4 Epsilon-equilibrium

If the strategy profile is a Nash equilibrium, no player can improve his utility by changing his strategy. In epsilon equilibrium, no player can improve his utility for more than some small constant $\epsilon$.

Formally:

The strategy profile $\sigma^*$ is an **$\epsilon$-equilibrium** iff the following inequality holds: $u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i^*, \sigma_{-i}) - \epsilon \ \forall i, \forall \sigma_{-i}$

### 1.7.5 Counterfactual regret minimization

The **Counterfactual Regret Minimization (CFR)** is an iterative algorithm used to find an $\epsilon$-equilibrium in the player zero sum games[27]. It computes strategy for both players in each iteration. Average of these strategies converges to a Nash equilibrium. Convergence rate of the algorithm is $O(1/\sqrt{T})$, where $T$ is count of the iterations.

The main advantage of this algorithm is that it has low memory requirements. It needs only memory linear to the count of the information sets.

Recently, many variants of this technique have been developed . Some of them use sampling to avoid of traversal entire game tree [6], other can also exploit structure of the game [14]. In 2013 ACPC competition, all top participants of No-limit competition used some version of counterfactual regret minimization algorithm.

## 1.8 Game Abstraction

Although we introduced strong game solving techniques, such as those described above, most poker games are still way too large to be solved directly by those techniques. For example, Heads-Up No-limit Texas Hold'em Poker, played at 2013 ACPC had $6.31 \times 10^{164}$ game states and $6.37 \times 10^{161}$ information sets [12]. Even all storage capacity on the world would not be sufficient to store the whole strategy for such game.

Therefore, some sort of a game abstraction is needed. To do that, we can merge different information sets together, restrict players to play only certain actions, or do combinations of these two approaches. More formally, a game abstraction is defined as follows [25]:

An **abstraction** for player $i$ is a pair $\alpha_i = \langle \alpha_i^{\mathcal{I}}, \alpha_i^A \rangle$, where:

- $\alpha_i^{\mathcal{I}}$ is partitioning of $H_i$, set of all histories where player $i$ acts. It defines set of abstract information sets that must be coarser than $\mathcal{I}_i$.
- $\alpha_i^A$ is a function on histories, that determines actions legal in the abstraction. It must satisfy $\alpha_i^A(h) \subseteq A(h)$ and $\alpha_i^A(h) = \alpha_i^A(h')$ for all histories $h$ and $h'$ in the same abstract information set. We will call this the **abstract action set**.

The **null abstraction** for player $i$, is $\phi_i = \langle \mathcal{I}_i, A \rangle$. An **abstraction** $\alpha$ is a set of abstraction $\alpha_i$ for all players $i \in N$. The **abstract game**, $\Gamma^\alpha$, is the extensive game obtained from the original game, $\Gamma$, by replacing the set of possible actions $A(h)$ with $\alpha_i^A(h)$, when $P(h) = i$, and replacing the information partition $\mathcal{I}_i$ with $\alpha_i^{\mathcal{I}}$, foll all $i \in N$.

### 1.8.1 Strategy in the abstraction

Strategy in an abstract game is defined in the same way as a strategy for an original game, but it is allowed to assign non-zero probability only to actions from abstract action set, and it must have same probability distribution in all histories from the same abstract information set.

### 1.8.2 State abstraction

If $|\alpha_i^{\mathcal{I}}| < |\mathcal{I}_i|$, i.e some information sets form the original game are merged, then the resulted abstraction is called a **state abstraction**. In large games with many information sets, like Texas Hold'em Poker, some form of state abstraction usually must be performed to keep the memory size required for game representation manageable.

### 1.8.3 Action abstraction

If $|\alpha_i^A(h)| < A(h)$ for some h, then the resulted abstraction is called **action abstraction**. In games with small count of actions, like limit poker games, there is often no action abstraction performed, but this is not possible in games with large number of actions like no-limit poker games. In these games, some combination of action abstraction and state abstraction is usually used.

### 1.8.4 Lossless abstractions

The abstraction of a game is **lossless** iff any strategy optimal in the abstraction is optimal also in the original game. Lossless abstractions exploit game isomorphism and merge isomorphic information sets together. For that purpose, optimal algorithm GameShrink, which can use all isomorphisms automatically, was developed [8]. Although this algorithm achieved some notable results in small artificial version of poker [8], lossless abstractions are still way too large to be used in many human scale games, like Texas Hold'em Poker.

### 1.8.5 Lossy abstractions

If an abstractions is **lossy** there is no restriction which information sets could be merged and which actions will be allowed for players to play. Consequently, there is no any guarantee that a strategy optimal in the abstraction will also perform well in the original game.

Because we have no restrictions on merging information sets or on restricting players' actions, we can create abstraction of any desired size, and therefore fit memory and time requirements of available computational resources. All recent Texas Hold'em Poker agents use lossy abstraction.

### 1.8.6 Perfect recall

If abstraction satisfies the **prefect recall condition**, i.e. players neither forget any information revealed to them, nor the order in which the information was revealed, this abstraction is called **perfect recall abstraction**. Early poker agents used only this type of abstraction.

### 1.8.7 Imperfect recall

In **imperfect recall abstractions**, players are allowed to forget information revealed to them. We can create imperfect recall abstractions also from original game with perfect recall.

If there is imperfect recall in abstraction, many theoretical guarantees, like existence of a Nash equilibrium, are lost. Moreover, many game solving algorithms, such as linear programming are not well defined for such game. On the other hand, in practice imperfect recall abstractions often perform better than perfect recall abstractions of the same size [26].

All top agents used some sort of imperfect recall abstraction in 2013 ACPC No-limit Texas hold'em competition[1].

### 1.8.8 Private and public information

If information about the game state is **public**, it is known to all players. **Private** information is known only to one player.

The **private state space abstraction** groups together information sets with similar private information. The **public state space abstraction** groups together sets with similar public information.

Traditionally, poker abstractions merged all information sets with similar private information, and differences in public information were ignored.

Despite of very good performance of public state space abstraction, only very limited attention has been devoted to it.

Recently, on 2013 ACPC no-limit competition, all strongest agents, except ours, used some form of public state abstraction[1].

Most of the agents use public state space abstraction handcrafted by human experts.

## 1.9 No-limit Texas Holdem Poker example

### 1.9.1 Action abstraction in poker

In No-limit Texas Holdem Poker, we can perform action abstraction by restricting allowed bets to only few possible chip counts. Early agents had abstractions with very limited types of possible bets. For example, only pot sized bet and all-in bet were allowed [23].

### 1.9.2 Private state abstraction in poker

In poker, private state abstraction considers strategic properties of the private hand hold by a player. For example, a player can be forced to play strong starting combinations, such as a pair of aces and a pair of kings, in the same way in the abstraction.

### 1.9.3 Public state abstraction in poker

Public state abstraction considers properties of board cards. For example, public flop card combinations 3♠3♡3♢ and 4♠4♣4♡ would be considered to be the same combination in the abstraction.

# 2. Properties and evaluation of game abstraction

## 2.1 Existence of Nash equilibrium in abstract game

### 2.1.1 Perfect recall

If abstracted game is a perfect recall game, all conditions for Nash theorem are satisfied and there exits at least one. But, unless the abstraction is lossless, there is not any guarantee that this strategy would perform well also in the original game.

### 2.1.2 Imperfect recall

For imperfect recall abstractions, there is no guarantee that some Nash equilibrium exits. Even worse, some of the algorithms for finding an optimal strategy are ill-defined and they do not give us any strategy at all.

Fortunately, we can use some sampling variants of counterfactual regret minimization algorithm. Strategies obtained by this algorithm does not have to converge to a Nash equilibrium (since there could be none), but in practice, resulting strategy is usually some $\epsilon$-equilibrium with sufficiently low value of $\epsilon$.

## 2.2 Abstract strategy in real game

### 2.2.1 Strategy development process

Most human scale problems, such as no-limit poker, need some sort of game abstraction to become solvable. Therefore, first step in development of a strategy is to create an abstraction.

Then, some game solving algorithm, like CFR, is used to obtain an $\epsilon$-equilibrium for the abstraction. It is typically impossible to hold the whole strategy for original game in memory, but we must be still able to play in it.

When it is player's turn, information set from original game has to be translated to information set in abstracted game. Then, an action is chosen according to probability distribution of our abstract strategy. The last step is to translate the action chosen in abstract game into action played in real game. This technique is called the **game translation**.

### 2.2.2 Abstraction pathology

Intuitively, strategy from bigger abstraction, that merges fewer information sets together, should perform better in original game that strategy from smaller abstraction. However, this is not always the case, even if the bigger abstraction is refinement of the smaller one. Counter examples have been shown in small artificial games [25], and most recently also in asymmetric poker abstractions [2].

### 2.2.3 Over-fitting

When we use iterative algorithm like CFR, the strategy obtained by this algorithm converges to Nash equilibrium in abstract game. But, this is not the case of performance of strategy in original game. Empirically, strategy is improving only during some number of iterations, and after that, performance of the strategy starts to decrease slightly. This phenomenon is called **over-fitting**.

### 2.2.4 Empirical performance

Abstraction pathology occurs very rarely, and bigger abstractions have the tendency to be better [17]. When all players use some game abstraction, over-fitting is not such a big problem, because the strategy has a tendency to improve itself in abstracted game.

It is possible to deal with both problems by using special version of the CFR algorithm called CFR-BR, which will be described later in this chapter.

## 2.3 Abstraction evaluation

Because there are many ways how to create abstractions, it is natural to try to find the best one. There are several ways how to compare abstraction techniques.

### 2.3.1 One-on-One performance

The simplest, and the most natural way of comparing two abstractions is to find optimal strategies in each of them and then just to let resulting agents play against each other in the original game.

Statistically significant amount of games has to be played to obtain meaningful results. In poker, several millions games are usually required.

We used this method for our experiments.

### 2.3.2 Performance against best response

Another option is to find an optimal strategy in the abstraction and then compute it's performance against the best response in the original game. This approach has two drawbacks.

First problem is that the computation of the best response can be intractable in the original game. Recently, new algorithm exploiting game structure was introduced [16]. We can compute the best response in domain of Limit Texas Hold'em Poker with this algorithm, but domain of No-limit Texas Hold'em Poker is still intractable with current technology.

The other drawback is that different abstract game equilibria can have a wide range of exploitability in the original game [25]. Therefore, obtained results are dependent of used solution technique, and this approach is very unreliable.

### 2.3.3 CFR-BR

The **CFR-BR**, is a variant of CFR algorithm capable of finding strategies in the abstraction that has minimal exploitability in the original game [13].

In order to evaluate the game abstraction, we will find strategy within this abstraction using CFR-BR and then we will compute it's performance against the best response in the original game.

This approach has been used successfully for evaluation of Limit Texas Hold'em Poker abstractions[13]. However, No-limit Texas Hold'em Poker is still too lager for this technique.

Luckily, One-to-One performance of two abstractions gives us results very similar to this technique, in poker [13].

# 3. Prior abstraction techniques

Arguably, Texas Hold'em Poker is the most studied testbed for game abstraction. We will describe state space abstraction techniques, that have been used and evaluated in this domain. Most of them take only the private information about game state into account.

## 3.1 Bucketing

**Bucketing** is a technique for state space reductions used in the card games. It will partition possible card combinations to certain fixed number of **buckets**. Only strategically similar cards should be in the same bucket. There are many ways, how to compute the similarity distance between the cards.

## 3.2 Expected hand strength

Probably the first measure of cards similarity was based on the concept of "strength" of the cards. The **expected hand strength (E[HS])** is defined as a probability of winning against random opponent hand [27]. For example, probability of winning against random hand during first round is 0.8520 for A♠ A♣, and 0.8240 for K♡K♠. So, these hands will be very likely merged into the same bucket.

## 3.3 Expected hand strength squared

**Expected hand strength squared (E[HS$^2$])** is another, slightly different version of the hand strength measure [27]. When we want to compute E[HS], we firstly compute probability that our hand wins against random opponent hand for each possible combination of the river board cards. E[HS] is then just an arithmetic mean of these probabilities. To compute E[HS$^2$], we use a mean of squares of these probabilities instead.

The main idea behind this measure is to better capture the potential of the hand to improve on the futre rounds of the game. In practice, abstractions created by E[HS$^2$] perform better than those created by E[HS] [27].

### 3.3.1 Percentile Bucketing

The **Percentile bucketing** is an approach used to divide hands into buckets using E[HS] or E[HS$^2$] measure. Firstly, the strength for each hand is computed. Then, hands are sorted according to their strength. When we want to divide hands to N buckets, we assign bottom 100/N percent of hands to the first bucket, the next 100/N percent of hands to the second bucket and so on.

## 3.4 Nested bucketing

**Nested bucketing** uses multiple levels of bucketing. Firstly, all card combinations are dived into top level buckets. Then, combinations from every top level bucket are divided into several lower level buckets. The final bucket for the hand is determined by vector of bucket numbers, one for each level. Typically, two levels are used.

In the first application of this technique, E[HS$^2$] bucketing was used for the top level and E[HS] for the bottom one [27].

## 3.5 History bucketing

The **bucket sequence** is the sequence of buckets that the player's card combination was placed into history. If we want to ensure perfect recall of abstraction, we can use the **history bucketing**, where only card combinations with the same bucket sequence from previous rounds are allowed to be in the same bucket.

## 3.6 Potential of the hand

When a hand is considered to be weak, i.e. it has low probability of winning, it can have different potential to improve itself and to become a strong hand on the future rounds of the game. Hands that have weak expectation of winning initially, but can improve to hands with very high expectation are called **drawing hands**. When we use expectation based bucketing, such as E[HS] with percentile bucketing, these hands are grouped together with hands that have also low probability of winning, but can not improve on later rounds. It was shown that the difference in potential is strategically important [9].

## 3.7 Abstraction as a clustering

Once we have distance function between the hands, we can automatically divide hands into buckets using some **clustering algorithm**. These algorithms try to divide elements (hands) into clusters (buckets), in such a way that the distances between the elements form one cluster are minimized. This problem is proved to be NP-complete, but good heuristic algorithms exist.

### 3.7.1 K-means clustering

**K-means clustering** aims to partition elements into clusters, in a way that each element belongs to one cluster. The number of clusters is some fixed constants $k$, which have to be specified before the algorithm starts. Each element belongs to a cluster with the nearest **mean**. The mean is computed as a mean value of all elements of the cluster. Because this problem is NP-complete, this algorithm finds only local optima. Multiple runs of algorithm are used to obtain better result.

## 3.8 Potential aware abstraction

To address problem with capturing potential of the hand, **potential aware abstraction** was introduced [9]. In this type of abstraction, multi-pass clustering is used. Firstly, hands on the last round are divided with k-means algorithm according to E[HS], distance. Then, for one round before the final, histograms showing probability of transitioning to the next round buckets are created for each hand. Distance between the hands is defined as L2 distance between these histograms. Using this distance, hands are again clustered with k-means algorithm. To create state abstraction for the whole game, we will proceed this way until we complete bucketing for the first round of the game. First version of this approach used perfect recall. Recently, new version of potential aware bucketing, which uses imperfect recall and **earth movers distance** between histograms, was introduced [5].

Earth mover's distance measures the "minimum work" we need to change one histogram into another. We will describe this distance in more detail, in the next chapter.

## 3.9 Distribution aware abstraction

### 3.9.1 Hand strength distribution

The E[HS] value o is the mean of all possible E[HS] values that a hand can have on the last round of the game. We can see that we lose a lot of information by forgetting these values and using only a mean instead.

Better representation of the hand is the **hand strength distribution**. It is defined as a histogram created from all possible E[HS] values that a hand can have during the last round of the game.

It represents the probability that a hand will have certain E[HS] for values from [0,1] interv.al. For an example of the hand strength distribution see figure 3.1;

### 3.9.2 K-means bucketing

In order to use hand strength distribution for bucketing, we can define distance between two hands as an earth mover's distance between their hand strength distributions. Then, some clustering algorithm, usually k-means, is used to cluster hands into buckets. This approach, known also as a **distribution aware abstraction**, is quite simple to implement and it has very good empirical results [17].

## 3.10 Opponent cluster hand strength

On the final round, there is not any potential for a hand to improve itself since there are none future rounds. Therefore, both, potential aware and distribution aware abstractions are equivalent with E[HS] on this round.

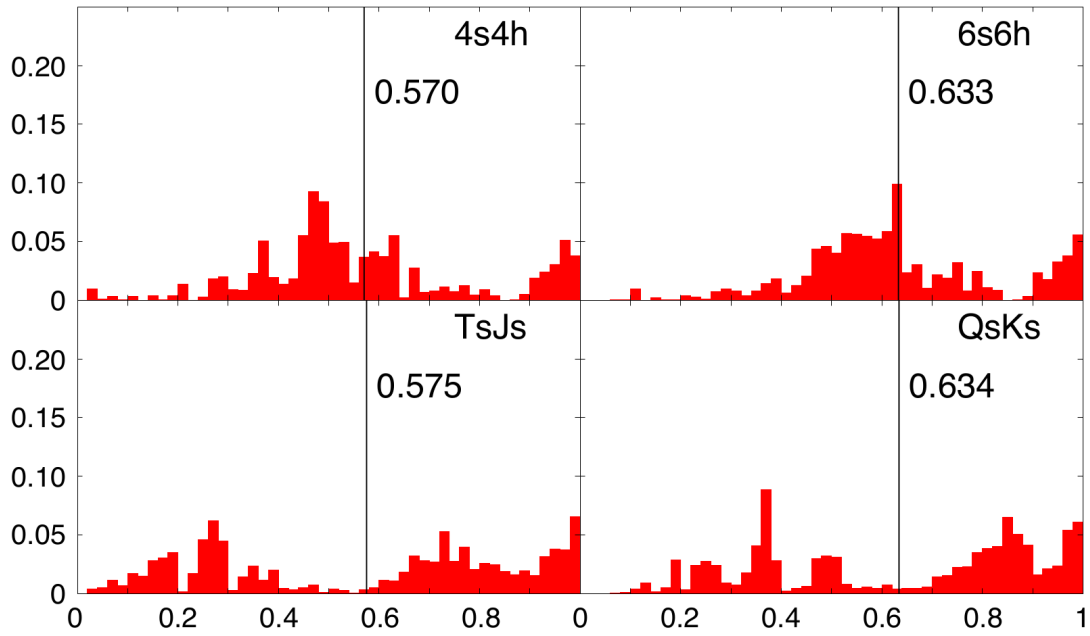



Figure 3.1: Hand strength distributions for hands on the first round of poker[17]. E[HS] of the hand is represented by vertical line. In each column, hands have very similar E[HS] value, but different hand strength distribution. On the contrary, hands have significantly different E[HS] value in each row, but similar hand strength distribution. Expectation aware abstraction would likely merge hands in the same column, while distribution aware abstraction would merge hands in the same row.

By definition, E[HS] gives our winning probability against opponent with uniform random distribution of hands. However, we can obtain even more information by looking at wining probabilities against other possible distributions of opponent hands.

The **opponent cluster hand strength (OCHS)** bucketing was designed for this purpose.

All possible starting hands are partitioned to several subsets called **opponent clusters** . Instead of computing single E[HS] value, we can now compute a vector of winning probabilities with one probability for each cluster. Distance between hands is defined as L2 distance between these vectors. Finally, some clustering algorithm, like k-means, is used to cluster hands into buckets.

## 3.11 Prior public state abstractions

With use of imperfect recall, we can handle more information about current game state. In addition to information about his own hand, a player can also distinguish between different public board cards. This has been shown to be significant advantage in the domain of computer poker [26].

### 3.11.1 Abstraction using transition table

First use of the public state abstraction in poker domain in literature was using a **transition table** [26]. This table captures change of hands expectations caused by public board.

To compute the table, prefect recall abstraction using E[HS] is created for first two rounds. Then, for the public board we are interested in, a vector of transition probabilities is created for each first round bucket. This vector has the same number of elements as the count of second round buckets, and the value of the $i$-th element is defined as a probability that random hand from the first round bucket will be in $i$-bucket on second round.

These vectors are rows of the transition table, one for each first round bucket.

We can now define the distance between two public boards as L2 distance between their transition tables. When the distance is defined, we can use clustering algorithm to cluster public boards into buckets.

Final abstraction uses nested bucketing with two levels. Top levels is used for the public information bucketing, and second level for bucketing of player hands.

### 3.11.2 Current public state abstraction techniques

There is only 1755 distinct, non-isomorphic public flop card combinations. Therefore, creation of a public bucketing is scalable by humans, and most of the current state of the art agents use public state abstraction hand crafted by expert poker players for the flop round.

This approach has several drawbacks:

- It is very domain specific - it would be difficult to apply it on another variants of poker or on another game types. For each game type, new, specialized human experts are needed.
- It has very low scalability - card combinations are typically assigned to buckets according to some system of rules defined by an expert, and the whole system of rules has to be changed to change number of buckets.
- We believe that the automatic solution with precisely described similarity between card combinations and with the support of a strong clustering algorithm can outperform expert approaches based only on human intuition.

## 3.12 Empirical performance of abstract strategies

As we can see, a lot of techniques for poker state space abstraction were developed. For each abstraction technique, there are also several parameters, that have big effect on it's performance. Therefore, a lot of effort is needed to find the best of them. Fortunately, there are several publications that focus on this problem.

### 3.12.1 Perfect recall, imperfect recall and public state abstraction

The publication [26] compares perfect recall and imperfect recall public state abstractions in poker.

It shows that for the expectation aware abstractions, imperfect recall is a great advantage in domain of no-limit poker, but this does not necessary hold for limit poker.

It came out for limit poker abstractions of the same size, that public bucketing along with imperfect recall has significantly better performance than perfect recall abstraction without public bucketing.

### 3.12.2 Comparison of distribution aware and expectation aware abstractions

The publication [17] introduces distribution aware bucketing and OHCS bucketing and compared them with an older expectation based abstractions. The combination of distribution aware bucketing and OHCS bucketing performed best from all of the evaluated abstractions.

### 3.12.3 Comparison of potential aware and distribution aware abstractions

The most recent work [5] shows that the potential aware abstraction along with imperfect recall and earth mover's distance slightly outperforms the distribution aware abstraction in the domain of no-limit poker.

## 3.13 Abstractions used by state of the art players

In limit poker, use of the imperfect recall allows abstractions to use large number of buckets in the first few rounds of the game. The top limit poker agents, like Hyperbolean [1] of Slumbot [10], use lossless abstraction for the first two or three rounds of the game and large number of buckets on later rounds.

For no-limit poker, state space abstraction has to be much smaller, therefore imperfect recall is used almost exclusively. Since there is only 169 distinct non-isomorphic hands at the first round of the game, all top agents use lossless abstraction for this round. Distribution aware or potential aware abstraction is used for private hands bucketing on later rounds

## 3.14 State space abstraction used by our agent in 2013 ACPC competition

Our agent Nyx participated in 2013 ACPC. We achieved 4th and 3/4th place in No-limit total bankroll and No-limit instant runoff competition [1].

Action space abstraction is very important in no-limit games. Because of that, we decided to restrict the maximum number of buckets to 1000 for our state space abstraction, so there is more space for action space abstraction.

For the first round of the game, our agent used lossless abstraction with 169 buckets, one for each possible non-isomorphic starting combination.

For next two rounds (flop and turn), we used distribution aware abstraction with 1000 buckets.

Finally, for the last round (river), we used OCHS bucketing with 1000 buckets and 8 clusters for the opponent's hands.

At the beginning of each round, all information about previous rounds buckets is forgotten.

# 4. Our new public state space abstraction

We designed new automatic method for creating public state space abstraction. Our approach is applicable on many extensive games, but we evaluated it in the domain of No-limit Texas Hold'em Poker.

## 4.1 Overview of our approach

Design of a state abstraction typically consists of two steps - definition of a distance measure between information sets and specification of clustering algorithm used for bucketing.

### 4.1.1 Distance measure

Basic idea behind our approach is to define the distance between two public states as the distance between two sets. Elements of these sets are all states sharing the same public information which corresponds to the public state represented by the set.

In poker, these sets consist of all hands which a player can have on the same public board.

We used the Earth Mover's Distance to measure distances between the sets. This measure can naturally extend notion of distance between elements to distance between whole sets.

There are many ways how to define distance measure between elements (in our case player's hands), and we examined two of them.

### 4.1.2 Clustering technique

Although most of the state space abstractions use k-means for clustering, our approach uses k-medoids. Big advantage of this approach is that it can use table of precomputed distances. This makes it much faster than k-means in our setting.

In practice, k-medoids often demonstrate better performance than k-means.

## 4.2 Overview of clustering algorithms

### 4.2.1 K-means clustering

In k-means clustering, we want to divide the data points into $k$ clusters, in a manner that each data point belongs to the cluster with the nearest mean. The **mean** is created as a mean value of all points in the cluster. The main objective is to find clustering with minimal sum of distances between means of a cluster and elements of that cluster. This problem is proved to be NP-complete. Therefore, a heuristic which finds only local optimum is used.

**K-means algorithm**

The algorithm consists of these four steps:

1. Place $k$ points into the data space which we want to cluster. These points represent initial centroids for clusters, and they are typically chosen randomly from input data points.
2. Assign each point to the nearest centroid.
3. For each group of points assigned to same centroid, calculate new centroid as a mean of these points.
4. Repeat steps 2 and 3, until the centroids remain unchanged.

To obtain better results, more runs of algorithm with different starting centroids are used. Solution with lowest squared error is then chosen.

### 4.2.2 K-medoids clustering

The **k-medoids clustering** is closely related to the k-means clustering. Like k-means, it clusters data into $k$ groups and tries to minimize the squared sum of distances between the center that represents the cluster and the data points belonging to that cluster. Difference from the k-means clustering is that k-medoids chooses a **medoid** as a center of a cluster in contrast to a mean. A medoid is usually defined as a data point from the cluster which has the lowest average dissimilarity to all objects in that cluster.

This clustering method is considered to be more robust to noise and outliers in comparison with k-means.

### 4.2.3 Partitioning around medoids algorithm

The most used algorithm for k-medoids clustering method is the **Partitioning Around Medoids (PAM)**. This algorithm is heuristic and it finds only local optimum. It consist of these four steps:

1. Randomly select $k$ data points as centers of clusters.
2. Associate each data point to the center which is closest in a chosen metric.
3. For each group of points associated to the same center find new center as point that has the lowest average distance to other points in the group.
4. Repeat steps 2 and 3 until there is no change in centers of the groups.

For better results, more runs with different cluster centers are used and the best resulting clustering is then picked.

As we can see, algorithm uses only distances between the data points, so all distances can be precomputed before the algorithm starts. Because computation of distances between each pair of points is independent, this task can be highly parallelized. This allows us to use distance measure with high computational demands.

In practice, k-medoids often have better performance than k-means.

## 4.3 Earth mover's distance

**The Earth Mover's Distance (EMD)** is a measure of distance between two multi-dimensional distributions in some feature space where a distance measure between features form that space, called the **ground distance**, is given [21].

When EMD is used as a distance between the histograms in distribution aware abstraction, as described in previous chapter, features are bins of these histograms and the ground distance is defined as a distance between centers of intervals defined by these bins.

When EMD is used as a distance between the public states, features for one public state will be all information sets with corresponding public information. More specifically, in case of poker, features for some public board will be all hands which a player can hold on that board. We define ground distance used in our abstraction later in the chapter.

Intuitively, Earth mover's distance measures the "minimum work" we need to change one distribution into another. We can see distribution as a mass of earth properly spread in space. Then, unit of work is defined as a moving unit of earth by unit of ground distance.

### 4.3.1 Formal definition

A distribution can be represented by a set of clusters. Each **cluster** is represented by pair of the **cluster representative** and fraction of the distribution belonging to that cluster called **cluster weight**. Such representation is called the **signature of the distribution**.

Now we can formalize the distance between these signatures by linear program [22]. Let $P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$ be the first signature with $m$ clusters; $Q = \{(q_1, w_{q_1}), ..., (q_n, w_{q_n})\}$ the second signature between $n$ clusters and $D = [d_{i,j}]$ the ground distance matrix where $d_{i,j}$ is the ground distance between clusters $p_i$ and $q_j$.

EMD between these two signatures is then defined as a minimum of:

$$\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{i,j} f_{i,j}}{min(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{p_j})} \tag{4.1}$$

Where $F = [f_{i,j}]$ is the flow; $f_{i,j}$ is the flow between $p_i$ and $q_j$ and it satisfies these constraints:

$$f_{i,j} \geq 0 \tag{4.2}$$

$$\sum_{j=1}^{n} f_{i,j} \leq w_{p_i} \tag{4.3}$$

$$\sum_{i=1}^{m} f_{i,j} \leq w_{q_j} \tag{4.4}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j} = min(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{p_j}) \tag{4.5}$$

Constraint 4.2 allows flow to moving "earth" only from $P$ to $Q$ and not the reverse way. Constraint 4.3 limits the amount of earth that can be sent from cluster to its weight. Constraint 4.4 limits amount of earth that can be sent to cluster to its weight. Finally, constraint 4.5 forces flow to move the maximum possible amount of dirt. This amount is also called the **total flow**. We can see, that EMD is by definition normalized by the total flow. This is not necessary when the total flow is always constant.

### 4.3.2 Properties of EMD

EMD has several advantages for our usage:

- It naturally extends the notion of distance from ground distance between elements to distance between the whole sets. We will use this property for clustering the public states.
- It is true metric when the sum of weights are equal for all signatures and the ground distance is metric. In our case, these conditions are satisfied and we can use it as an input for clustering algorithms.
- It describes similarity in very natural way. This was shown on domain of image processing [21, 22].
- It has very good performance in the domain of poker where it outperforms standard distance measures like L2 [17, 5].

## 4.4 Distance between hand strength distributions

As we have mentioned in the previous chapter, distribution aware clustering uses earth mover's distance between hands strength distribution histograms. In this case, clusters of distribution signature are bins of the histograms and the distance between them is defined as a distance between bin centers.

### 4.4.1 Computation of EMD between two hand strength distributions

Signatures corresponding to hand strength distributions have dimension one, count of the clusters is constant and sum of the weight is always one. This allows us to compute EMD very fast.

In one-dimensional EMD algorithm, the input are two signatures $P$, $Q$. All hand strength histograms have the same bins, thus the corresponding clusters are also the same and $p_i = q_i$. We will assume clusters are ordered (that means $p_i < p_{i+1}$), and the distance between consecutive clusters is constant. These assumptions are naturally satisfied in our case.

This allows us to compute EMD by scanning array of clusters and keep the track of how much "earth" need to be moved between consecutive bins. It can be expressed by formula [4]:

$$s_0 = 0 \tag{4.6}$$
$$s_{i+1} = (w_{p_i} + s_i) - w_{q_i} \tag{4.7}$$
$$EMD = \sum |s_i| \tag{4.8}$$

Variable $s$ represents a "shovel" we use to move earth between the clusters. It is easy to see that algorithm runs in time linear to the size of the histogram.

## 4.5 Distance between public states

In our poker application, we represent board card combination as a set of the hands players can hold on that board, and we measure the distance between boards as EMD between these sets.

There are 1755 distinct non-isomorphic possible boards on the flop and we need to compute distance matrix which captures the distance between each possible pair for the clustering algorithm.

On each flop board, a player can hold one of 1176 hands. Each cluster of flop signature represents one hand. The count of possible hands is constant for all flop boards, thus the count of the clusters is also constant. We assigned arbitrary but constant weight to the each cluster.

To compute EMD between the boards, some ground distance has to be specified. When this distance will be metric, the resulting measure will also be metric. We will discuss two possible ground distance definitions later in the chapter.

### 4.5.1 Computation of EMD

Definition of EMD 4.2 gives us good defined linear program. The problem described by this program is called the **transportation problem**. We can solve this problem directly by some LP solver, or we can use some combinatorial algorithm.

It is easy to see this optimization problem as an instance of the **minimum-cost flow problem**, as well as an instance of the **minimum-weight perfect matching problem** [4]. For both of these problems, there are powerful polynomial time solving algorithms. We obtained the best results by using minimum-cost flow problem solver.

The distance between each pair of public states has to be computed for clustering. Thus, the count of distance computations grows asymptotically quadratic to the count of public boards. Because each single distance computation is an optimization problem, the total computational time is quite big. Fortunately, since single computation is independent, we used massive parallelization.

## 4.6 Ground distance between hands

Crucial part of our approach is to specify good ground distance. It has huge impact on the final distance measure and consequently on the quality of final bucketing.

### 4.6.1 Distribution aware ground distance

In distribution aware abstraction, the distance between two hands is specified as EMD between their hand strength distributions. Since we can use any metric to specify ground distance between hands, use of the same measure seems to be a natural choice.

Thus, this was the first option of ground distance we had examined. Contrary to our expectations, the resulting abstraction had poor quality, according to human experts and it had also bad performance in our experiments.

We have an idea, why this approach did not work. When we cluster hands on some round of game, our target is a good answer to the player's question: "How does my hand look like?". But when we cluster public boards, the question is different: "What kind of hands will the opponent have in this situation?". To answer the second question, we have to know how had the opponent played his hands on rounds before the actual one. Thus, our ground distance should take into account also information about the properties the hand had during previous rounds of the game.

Human players also try to find the answer to the second question during the play. They call this skill "hand reading".

### 4.6.2 Distribution history aware ground distance

EMD between hand strength distributions seems to be good for comparing hand properties on current round of the game. To capture information about hand properties on previous rounds, we will not represent hand only by single hand strength distribution, but rather by a vector of these distributions, one for each round until the current one. We call this vector the **distribution history vector**.

We define the distance between two distribution history vectors $p = (p_i, .., p_n)$ and $q = (q_i, ..., q_n)$ as:

$$\sqrt{\sum_i (EMD(p_i, q_i))^2} \tag{4.9}$$

Where $EMD(p_i, q_i)$ is the Earth mover's distance between $i$-th elements of the vectors.

We can define the ground distance between hands as a distance between their distribution history vectors, and we call it the **distribution history aware distance** . It gives us good empirical results in the domain of No-limit Texas Hold'em Poker.

## 4.7 Abstraction of the whole game

In order to obtain abstraction for the whole game, also private state of the game has to be taken into account. For this purpose, arbitrary bucketing can be used. We decided to use distribution aware bucketing due to it's good performance, low computational cost and relatively easy implementation.

### 4.7.1 Nested bucketing

In order to combine our public state abstraction technique with private state space abstraction, we used nested bucketing with two levels of buckets. Top level bucket is determined by public board cards. Second level bucket is determined by the hand that the player holds.

We computed private hand buckets independently for each top level public bucket.

## 4.8 Summary of our approach

First step in our approach is to cluster board card combinations to public buckets. To do that, the distance matrix between all boards has to be computed.

Each board card combination is represented as a set of hands which a player can hold on that board. The distance between boards is defined as EMD between these sets. We used minimum-cost flow solver for EMD computation, and distribution history aware distance as the ground distance.

Once the distance matrix is computed, public board combinations are clustered into the buckets with k-mediods algorithm.

When public bucketing was created, arbitrary hand clustering algorithm is used to cluster private hands. We used distribution aware bucketing for this purpose.

## 4.9 State space abstraction used by our agent in 2014 ACPC competition

We submitted our new agent Nyx 2014 to the 2014 ACPC competition. This agent uses our new abstraction technique.

The size of new abstraction is the same as the size our last year agent's -it is 1000 buckets for each round, except the first one. We decided to expand the size of action space abstraction instead.

We used lossless abstraction for the first round (preflop).

For the second round (flop), we used our new approach. We had 20 top level buckets for public boards, and for each of these buckets, there were 50 lower level buckets for private hands. So the total count of flop buckets was 1000.

For the third round (turn), we used standard k-means distribution aware bucketing with 1000 buckets. We did not use public bucketing for this round.

For the last round (river), we used OCHS bucketing with 1000 buckets.

## 4.10 Implementation notes

### 4.10.1 Representation of buckets

When the bucketing is done, there are two ways how to represent it. We can hold only centers (mediods or means) of buckets. When we want to find which bucket a hand belongs to during a game play, we have to compute distribution histograms for this hand and then to find bucket with the nearest center.

This representation of bucketing has very low memory requirements, but computation of hand strength distribution histogram has to be performed after each change of the game state. We initially used this representation.

#### Tables for bucketing

To evaluate abstraction with some meaningful confidence interval, large number of games has to be played. Bottleneck in a game play speed is computation of bucket number for a hand. To speedup this process, we decided to use lookup tables.

Our tables contain bucket numbers for each possible hand and are indexed by hand identifier. To obtain this identifier, we use algorithm exploiting hand isomorphisms, similar as [24]. This allows us to keep the size of tables manageable small and to keep them in the RAM. We used tables for the flop and turn round in our implementation. This allows us to play hundreds of games per second on the desktop CPU.

# 5. Experimental results

In order to evaluate our new abstraction technique, we performed experiments in the domain of No-limit Texas Hold'em Poker. State space abstraction has to be much smaller for no-limit poker than in limit poker. Thus, algorithms for abstraction have to be very effective.

## 5.1 Game parameters

### 5.1.1 Stack size

One of the important parameters of a poker game is the size of players' stacks. Many games with different stacks' sizes are played regularly. In the ACPC competition, both players have a stack of 200 big blinds. Most matches among humans start with stack size of 100 big blinds, but a variety of games is played with higher and also lower stacks. Even games with stacks lower than 20 big blinds are often played.

In our experiments, we used action abstraction with stacks equal to 25 big blinds.

One of the advantages of using small stack is that one-to-one play produces small observed standard deviation and we can obtain narrow confidence interval for our results.

Another advantage is that we can keep action space of the game relatively small, but still capable to represent advanced betting strategies.

### 5.1.2 Action abstraction

To keep the size of action abstraction reasonably small is very important, because it allows us to produce strong strategies in manageable computational time.

In betting abstraction we used, there are up to five bet sizes allowed according to the game state and previous bets of players. All-in bet is always a valid option.

Only 2513 betting sequences are possible in the abstraction, what is few magnitudes less than in action abstractions we used for our ACPC competitors.

## 5.2 Used state space abstractions

We used two state space abstractions for our experiments. Both abstractions had the the same amount of buckets - 169 for the first round and 1000 for later rounds.

### 5.2.1 Our new abstraction

The first abstraction we used, was our new abstraction for 2014 ACPC competition. It used lossless abstraction for the first round. For the second round it used nested abstraction, with 20 buckets for public information and 50 buckets for private information. Public buckets were created by our new abstraction technique and private buckets by distribution aware method.

For the third round distribution aware abstraction without any public bucketing was used. Finally, for the last round, OCHS bucketing was used. We used imperfect recall and all information from previous rounds was forgotten at the beginning of each new round.

### 5.2.2 Old state of the art abstraction

The other abstraction used for comparison was our abstraction from 2013 ACPC competition. It is the same as 2014 abstraction, except for the second (flop) round. Distribution aware abstraction without public information bucketing is used for this round.

## 5.3 Used Strategies

We computed $\epsilon$-equilibrium strategies for both abstractions. We used one million iterations of public sampled CFR algorithm [6] to obtain these strategies. This version of CFR exploits poker game structure and it is thus very effective in our domain.

Both strategies used the same action space abstraction and therefore we did not need to translate actions from these strategies into the original game.

## 5.4 Evaluation method

We chose one-to-one performance as our evaluation method because, as shown in [17], we can obtain very good comparison results with relatively low computation time.

Initially, we used ACPC protocol for players' communication. Since we used precomputed bucketing tables, we were able to play quite fast, approximately 100 games per second. However this still was not sufficient for our purposes. Because of hte high standard deviation, we were not able to obtain confidence interval tight enough with this approach.

### 5.4.1 Our new comparison algorithm

Instead of playing a game via ACPC protocol, we deiced to use a game tree traversal similar to that used by CFR algorithm. In this approach, we sample one random public board and some arbitrary number of betting histories in each iteration.

This technique is the same as the one used to obtain counterfactual value in the Monte Carlo CFR algorithms [6]. It was proved to give unbiased results.

We exploited poker structure compute values for all hands for both players in linear time. To do that, we used the same trick as is used by PCS-CFR algorithm [14].

In order to obtain good results with low variance, we sampled each betting history according to the probability that players would play this history.

Because traversal directly on the game tree was used, there was not any communication overhead and we were able to play large number of iterations

in a short time period. Much bigger part of the game tree was sampled than in the standard one-to-one play, thus the observed standard deviation was much smaller.

This allowed us to compare strategies with narrow confidence interval.

## 5.5 Measure of winnings

In heads-up poker, players switch their positions in every hand. This makes the game symmetrical and the game value is consequently a zero. Therefore, the agent's performance can be measured by sum of chips he is able to win.

Unit for this measurement is one **milliblind per hand (mb/h)**, where a milliblind is 0.001 big blinds, the biggest bet players are forced to post on the start of the game.

## 5.6 Obtained Results

Our new public state abstraction outperformed old state of the art abstraction by **3.473 mb/h** in the domain of No-limit Texas Hold'em Poker with 25 big blinds stack.

We run 195,000,000 iterations of our new comparison algorithm to obtain this result. This gave us confidence interval $\pm$ 0.4 mb/h.

## 5.7 Evaluation of Results

As we can see, our new abstraction resulted into significantly stronger game strategy. We believe we will obtain even better results when we will implement our public state space abstraction also for later rounds of the game.

In the new abstraction, we used only 50 buckets for player's hands. This is very small number compared with 1000 buckets used for hands in the older abstraction. Limit of 1000 buckets for each round is due to our ACPC agent's requirements.

We believe that larger number of total buckets, and consequently lager number of buckets for player's private hands, will be another significant advantage for our new abstraction.

Since there is possiblity to win much more in game with 100 or 200 big blind stacks than in our experimental game, the advantage of our abstraction is probably lager in these games.

## 5.8 ACPC 2014

The performance of our state space abstraction along with our other improvements will also be evaluated at ACPC 2014 competition by our agent Nyx 2014. Unfortunately, the deadline for this thesis is before the date when the competition results will be published. Therefore we cannot evaluate our performance. Results of the competition and our performance can be found on competition website[1].

# Conclusion

This thesis described state of the art techniques for the state space abstractions of extensive games with imperfect information. We focused on poker domain, especially on No-limit Texas Hold'em Poker. Abstractions used by our successful ACPC 2013 and new ACPC 2014 competitors were presented.

## 5.9 New public state space abstraction technique

We developed new public state space abstraction technique for our 2014 ACPC competitor. This technique is capable of creating high quality state space abstraction fully automatically. Strong point of our algorithm is the new measure of similarity between public states. This measure has nice properties, and an idea behind it is similar to the techniques behind human reasoning in poker.

## 5.10 Experimental results

In order to examine our new abstraction technique, we used one-to-one play in domain of No-limit Texas Hold'em Poker. Our new algorithm has outperformed state of the art abstraction technique used by our last year agent by significant amount.

## 5.11 Future work

We hope we will be able to successfully implement our new abstraction technique to more rounds of a game, and eventually even to more types of games. We want to investigate the trade-off between state space and action space abstraction and also the trade-off between the size of private and public abstraction.

We also want to continue in our work in other areas of game theory, and eventually to produce top level agent for next years' ACPC competitions.

# Bibliography


[1] ACPC, June 2014. http://www.computerpokercompetition.org.

[2] Nolan Bard, Michael Johanson, and Michael Bowling. Asymmetric abstractions for adversarial settings. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-14)*, 2014.

[3] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[4] Scott Cohen. Finding color and shape patterns in images. Technical Report STAN-CS-TR-99-1620, Stanford University (Stanford,CA US), Stanford, 1999.

[5] Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. 2014.

[6] Richard Gibson, Neil Burch, Marc Lanctot, and Duane Szafron. Efficient monte carlo counterfactual regret minimization in games with many player actions. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1889–1897. 2012.

[7] Andrew Gilpin and Tuomas Sandholm. A texas hold'em poker player based on automated abstraction and real-time equilibrium computation. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1453–1454, New York, NY, USA, 2006. ACM.

[8] Andrew Gilpin and Tuomas Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM (JACM)*, 54(5):25, 2007.

[9] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *IN AAAI'07*, 2007.

[10] Eric Jackson. Slumbot nl: Solving large games with counterfactual regret minimization using sampling and distributed processing, 2013.

[11] Michael Johanson. Robust strategies and Counter-Strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, October 2007.

[12] Michael Johanson. Measuring the size of large no-limit poker games. *CoRR*, abs/1302.7008, 2013.

[13] Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive form games. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)*, 2012.

[14] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2012.

[15] Michael Johanson and Michael Bowling. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 264–271, 2009.

[16] Michael Johanson, Michael Bowling, Kevin Waugh, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 258–265, 2011.

[17] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[18] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 721–728, Cambridge, MA, 2008. MIT Press.

[19] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[20] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[21] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society.

[22] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000.

[23] David Schnizlein, Michael Bowling, and Duane Szafron. Probabilistic state translation in extensive games with large action sets. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 276–284, 2009.

[24] Kevin Waugh. A fast and optimal hand isomorphism algorithm, 2013.

[25] Kevin Waugh, David Schnizlein, Michael Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 781–788, 2009.

[26] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael Bowling. A practical use of imperfect recall. In *Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2009. To appear.

[27] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 905–912, 2008.

# List of Tables

# List of Abbreviations

# Attachments