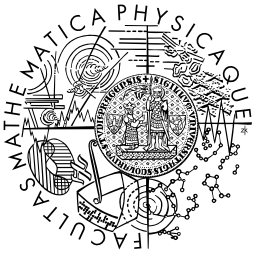


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

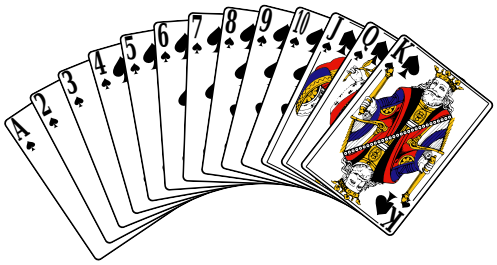
**MASTER THESIS**

Bc. Karel Ha

# Solving Endgames in Large Imperfect-Information Games such as Poker


Department of Applied Mathematics

Supervisor of the master thesis: doc. Mgr. Milan Hladík, Ph.D.
Study programme: Computer Science
Study branch: Discrete Models and Algorithms



Prague 2016

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 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

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

Title: Solving Endgames in Large Imperfect-Information Games such as Poker

Author: Bc. Karel Ha

Department: Department of Applied Mathematics

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

Abstract: Endgames have a distinctive role for players. At the late stage of games, many aspects are finally clearly defined, deeming exhaustive analysis tractable. Specialised endgame handling is rewarding for games with perfect information (e.g., Chess databases pre-computed for entire classes of endings, or dividing Go board into separate independent subgames).

An appealing idea would be to extend this approach to imperfect-information games such as the famous Poker: play the early parts of the game, and once the subgame becomes feasible, calculate an ending solution. However, the problem is much more complex for imperfect information.

*Subgames* need to be generalized to account for *information sets*. Unfortunately, such a generalization cannot be solved straightaway, as it does not generally preserve optimality. As a consequence, we may end up with a far more exploitable strategy.

There are currently three techniques to deal with this challenge:

(a) disregard the problem entirely;

(b) use a decomposition technique, which sadly retains only the same quality;

(c) or formalize improvements of strategies into a so-called *subgame margin*, for which we construct a "gadget" game that optimizes for the subgame margin.

The last approach is our own result presented at the Thirtieth AAAI Conference on Artificial Intelligence in 2016.

We experimentally compare the three solutions using a top participant of the AAAI-14 Computer Poker Competition, the leading playground for agents in imperfect-information setting.

Keywords: algorithmic game theory, imperfect-information games, Nash equilibrium, subgame, endgame, counterfactual regret minimization, Poker

I wish to thank and dedicate my work to:

→ Milan, for assuming the (uneasy) role of my supervisor, for revealing his way of "the optimized consumption", and for helping my EPFL dream come true;

→ Martin and Matěj, for accepting me under their guardian wings as a protégé, and for assisting me in my very first publication;

→ the facilities of CERN, University of West Bohemia, CrossCafe, and Education and Research Library of Pilsen Region, for being my sanctuaries when I needed to escape the silence of my room;

→ the Wikipedia, for filling me up with all the wisdom and knowledge, which I often didn't know I was missing;

→ the team of AlphaGo, for making such an awesome step towards general AI;

→ azquotes.com, for inspiring and amusing quotes, which I "abused" to lighten up the dimness of this work;

→ Donald Knuth, for all his hilarious quotes, and for teaching me never to use Microsoft Word again;

→ to my little brother, for his patience as my roommate during my work on this;

→ and last but not least, to my loving parents, for running our childhood smoothly, providing for us selflessly, and especially, for waking us up every single morning during our school years... despite our sincerest resistance.

> The whole thing that makes a mathematician's life worthwhile is that he gets the grudging admiration of three or four colleagues.
>
> Donald Knuth

# Contents

# Introduction

> A successful person isn't necessarily better than her less successful peers at solving problems; her pattern-recognition facilities have just learned what problems are worth solving.
>
> Ray Kurzweil

Endgames have a special role for game playing: the situation towards the end becomes simpler and many game aspects are clearly defined, e.g., remaining late-game pieces in Chess, or the division of territories in Go. As opposed to openings and mid-games, endgames therefore offer the opportunity for a thorough scrutiny such as an exhaustive search.

This approach is fruitful in games with perfect information (Part I), where there has been an extensive use of it: a perfect play has been pre-computed for Chess endgames with up to 7 pieces (Chapter B), while Go endgames can undergo dynamical in-play analysis, either using combinatorial game theory (Chapter C) or Monte Carlo simulations aided by convolutional neural networks (Chapter D).

An appealing idea is to extend this approach to imperfect-information games: play the early parts of the game, and once the subgame becomes feasible, calculate a solution using a finer-grained abstraction in real time, creating a combined final strategy. Although this approach is straightforward for perfect-information games (Chapter A), it is a much more complex problem for imperfect-information games (Part II).

Firstly, we have to take into consideration *information sets*, and generalize the concept of a *subgame* for imperfect information (Chapter E). Unfortunately, directly re-solving a subgame does not in general preserve any guarantee of optimality, resulting in far more exploitable strategies. We will observe this phenomenon on the elementary example of Rock-Paper-Scissors game with a single subgame (Chapter F).

The posed challenge may be tackled in three different ways:

(a) Ganzfried and Sandholm (Carnegie Mellon University) disregard the problem entirely, by advocating empirical improvement over theoretical guarantees (Chapter G).

(b) Burch, Johanson, and Bowling (University of Alberta) propose a decomposition technique, which retains at least the same quality of subgame strategies. Nonetheless, the method receives no incentive to strive for maximal gains (Chapter H).

(c) We—Moravčík, Schmid, Ha, Hladík (Charles University) and Gaukrodger (Koypetition)—formalize the improvement of subgame strategies into the notion of a *subgame margin*. On top of that, we devise a construction of an equivalent extensive-form game, which maximizes subgame margins (Chapter I).

Finally, Chapter J provides some suggestions to further develop these ideas, and enhance them for future work.

# 0. Preliminaries

> I can't go to a restaurant and order food because I keep looking at the fonts on the menu.
>
> Donald Knuth

Let us start with some games...

## 0.1 Game Theory

> You have to learn the rules of the game. And then you have to play better than anyone else.
>
> Albert Einstein

Game theory is a mathematical discipline dealing with interactions (either conflicts or cooperation) between 2 or more agents. Founded by (Von Neumann and Morgenstern 1953), game theory has countless application in economics, political science, and psychology, as well as logic, computer science, biology and Poker.

Informally, a *game* has typically

- ♠ players, who are assumed to be rational decision-makers (i.e., they would not make a suboptimal choice of actions),
- ♠ actions available to each player,
- ♠ payoffs (or utilities) for every possible combination of such actions.

Games can be represented in two forms:

- ♠ a normal-form game (NFG) is a representation using (pay-off) *matrices*. Examples of NFGs can be found in Section 0.2.
- ♠ an extensive-form game (EFG) models a *game with turns* using a *game tree*. They are discussed in greater details in Section A.1 and Section E.1.

## 0.2 Examples of Games

> Happy Hunger Games! And may the odds be ever in your favor.
>
> Suzanne Collins, *The Hunger Games*

**Rock-Paper-Scissors (RPS)** is a game commonly used to make decision between two people. Two players pick a (symbolical) tool of their choice: Rock, Paper or Scissors. Each tool wins against one of the remaining tools and loses against the other one: Scissors cuts Paper, Paper covers Rock and Rock crushes Scissors (check also Section F.1).

**Chess** is arguably world's most famous game, frequently employed to sharpen one's intellectual facilities. For decades, this mind sport has functioned as a "playground" for artificial intelligence (AI) research.

The most notable milestone of computer Chess took place in New York City, in 1997: the AI community (with the rest of the world) was astonished by the victory of IBM's computer system *Deep Blue* against the world Chess champion, the grandmaster *Garry Kasparov*.

> It was an impressive achievement, of course, and a human achievement by the members of the IBM team, but Deep Blue was only intelligent the way your programmable alarm clock is intelligent. Not that losing to a $10 million alarm clock made me feel any better.
>
> Garry Kasparov

The rules of Chess are widely known; for completeness, however, Chess tutorials such as (Karpov 1997) are recommendable.

**Go** is another ancient game with countless number of both amateur and professional players. The rules (Section 0.3) are even simpler than in Chess; yet the game is far more complex, with more positions than atoms in the observable Universe (Section D.1).

As a game of intellect, Go is especially popular with mathematicians. A dedicated mathematical discipline of combinatorial game theory (Section 0.4) has been developed by excellent John Conway, in order to decompose and study endgame positions (Chapter C).

A few months ago, the Go world experienced a duel similar to the one of Deep Blue vs. Kasparov. The Humanity, represented by legendary Korean player Lee Sedol, was defeated by the AI of Google DeepMind's program *AlphaGo* (Chapter D).

**Nim** is a mathematical game in which two players take turns in removing objects from distinct heaps. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap. The goal is to be the player to remove the last object.

**Poker** is a popular card game and an example of an imperfect-information game: as opposed to all previous games, there is a vital role in players' *private* cards and the *chance*. We will see that notion of a subgame needs to be adapted for imperfect-information games, which will cause several complications and will call for novel solutions (Part II).

Following the examples of Deep Blue and AlphaGo, also the Poker world had a chance to meet a superhuman AI champion: *Cepheus*—the first Pokerbot which has *weakly solved* Limit Texas Hold'em variant of Poker (Bowling et al. 2015).

Currently, the game of the most active computer Poker research is the two-player version of No-Limit Texas Hold'em (Heads-Up NLHE). For the rules of it, read for instance Appendix A of (Ganzfried and Sandholm 2015).

**Note**

The following examples and figures are taken from the classic textbook *Algorithmic Game Theory* (Nisan et al. 2007).

> Football is a more beautiful game in high definition.
>
> José Mourinho

**Matching Pennies** is a 2-choice version of RPS. Two players, *Matcher* and *Mismatcher*, pick either H(eads) or T(ails) of a (penny) coin. If chosen sides match, Matcher wins (receives the utility of 1) and Mismatcher loses (receives the utility of $-1$). If the sides mismatch, Mismatcher wins and Matcher loses.

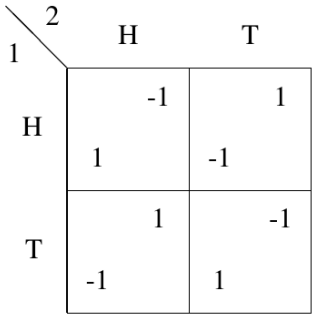



Figure 1: Matching Pennies: player 1 as the Matcher, player 2 as the Mismatcher (Nisan et al. 2007)

**Prisoner's Dilemma** is a notorious example from game theory. Two players, prisoners, are on trial for a crime, each with choice to confess or to remain silent.

If they both keep silence, charges against them cannot be proved, and both will serve a short prison term of 2 years (for minor offenses). If just one of them confesses, his term will be reduced to 1 year and he will be used as a witness against the other, who in turn will get a sentence of 5 years. Finally, if they both confess, they both will get a small relief for cooperation, and each will be sentenced to 4 years instead of 5. (Nisan et al. 2007, Section 1.1.1)

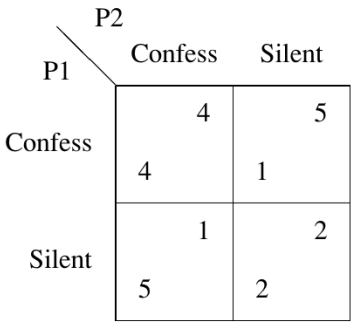



Figure 2: Prisoner's Dilemma (Nisan et al. 2007)

The situation modeled by the Prisoner's Dilemma arises naturally in a lot of different situations. One such example is ISP Routing Game.

**ISP Routing Game** models an Internet Service Provider (ISP), who needs to send Internet packets from source $s_1$ (in his own network ISP1) to target $t_1$ (in the network ISP2 of another provider). The two networks are connected via two transit points, $C$ (for confess) and $S$ (for silent):

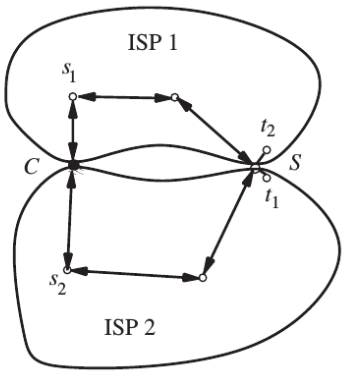



Figure 3: ISP Routing Game (Nisan et al. 2007)

There is a unit cost per edge: if the provider sends a packet through closer $C$, it costs him 1 unit and the opponent pays 4 units for routing from $C$ to $t_1$. *ISP1 behaves selfishly!* On the contrary, if he sends a packet through farther $S$, it costs 2 units. The opponent however pays only 1 unit, because $t_1$ is nearer to $S$ than to $C$. *ISP1 behaves altruistically!*

Now ISP2 will send traffic, too, choosing again between $C$ or $S$. If we accumulate all costs from both traffics, and write it down for each combination of selfish/altruistic players, we get the identical NFG as in Figure 2.

**Traffic Light** Two players, drivers of cars, arrive at a crossroad perpendicularly to one another. If at most 1 driver crosses, the situation will be safe and their payoffs will be non-negative, with a slightly better payoff for the passing driver. If however both drivers decide to pass the crossroad, the result will be sub-optimal, as both drivers obtain drastically negative payoffs and die.

| 1 \ 2 | Cross | Stop |
|---|---|---|
| Cross | -100<br>-100 | 0<br>1 |
| Stop | 1<br>0 | 0<br>0 |

Figure 4: Traffic Light (Nisan et al. 2007)

This is an example of a *coordination game*, where a common trusted *coordination device* is desirable. Such a device (e.g., a traffic light or the right-of-way priority rule) justify the concept of a *correlated equilibrium* (Nisan et al. 2007, Subsection 1.3.6).

**Battle of Sexes** is another coordination game: Two players, Boy and Girl, are arranging an activity for their date. The Girl wishes to go (S)hopping, while the Boy wants to go for a (B)eer:

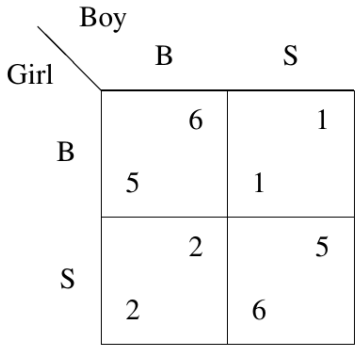



Figure 5: Battle of Sexes (Nisan et al. 2007)

Notice they both prefer to agree (rather than disagree) on the activity, because this way they will be together. If however, they disagree and end up alone without a date, both would rather spend the evening doing their favorite activity.

## 0.3 The Game of Go

> You don't have to be really good anymore to get good results. What's happening with Chess is that it's gradually losing its place as the par excellence of intellectual activity. Smart people in search of a challenging board game might try a game called Go.
>
> Hans Berliner

*Black* and *White* place pieces (*stones*) on the unoccupied intersections (*points*) of a *board* with a $19 \times 19$ grid of lines. Players take turns, Black moves first. Surprisingly, there are only 2 basic rules of Go:

**The rule of liberty** Every stone remaining on the board must have at least one open point (an intersection, called a *liberty*) directly next to it (up, down, left, or right), or must be part of a connected group that has at least one such liberty next to it.

Stones or groups of stones which lose their last liberty are removed from the board.

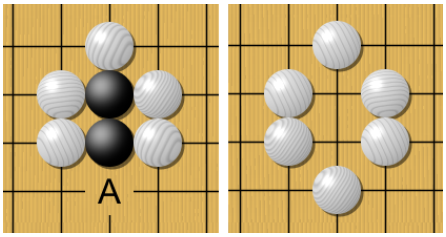



Figure 6: The rule of liberty

**The *Ko* rule** The stones on the board must never repeat the previous position of stones, so as to prevent unending cycles in game play.

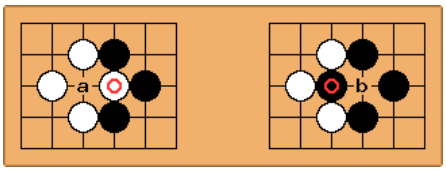



Figure 7: The *Ko* rule

Since Ko rule applies only to a previous move, this gives rise to *Ko fights*: the player, who is "banned" from repeating a move, makes a play elsewhere, which may have no particular good qualities, but which demands an instant reply. Then the ban comes to an end, and recapture is again possible for Ko. This kind of distracting play is called a *Ko threat*.

There are several **scoring rules** to determine the winner of a game. In the match of AlphaGo against Lee Sedol (Chapter D), the *area scoring* was used. Under area scoring system, player's score is:

- the number of stones that the player has on the board
- plus the number of empty intersections surrounded by that player's stones
- plus *komi(dashi)* points for the White player (i.e., a compensation for the first move advantage of the Black player)

*Elo rating* can be used to denote players' **ranks**. Alternatively, *kyu/dan* (in Japanese) or *gup/dan* (in Korean) system is also widely popular:

| **Rank Type** | **Range** | **Stage** |
|---|---|---|
| double-digit kyu[1] (DDK) | 30–21k | beginner |
| double-digit kyu | 20–11k | casual player |
| single-digit kyu (SDK) | 10–1k | intermediate amateur |
| amateur dan | 1–7d | advanced amateur |
| | (8d is special title) | |
| professional dan | 1–9p | professional player |
| | (10d is special title) | |

Table 1: Ranks in Go

**Handicap** system is used to even up differences in ranks: Black can place 1 or more stones in advance as a compensation for White's greater strength.

[1] gup in Korean

## 0.4 Combinatorial Game Theory

> The simplest game of all is the *Endgame*, 0. I courteously offer you the first move, and call upon you to make it. You lose, of course, because 0 is defined as the game in which it is never legal to make a move.
>
> John Conway, *On Numbers and Games*

**Note**

This section is based on the founding work *On Numbers and Games* (Conway 1976). For more on combinatorial game theory, read also the illustrative (Berlekamp, Conway, and Guy 1983).

The combinatorial game theory (CGT) studies games without chance such as Nim or Go. Combinatorial games can be viewed as mathematical models of Go positions.

**Definition 1.** A (combinatorial) game is an ordered pair of sets of games:

$$G = \{\mathcal{G}^L | \mathcal{G}^R\}$$

where $\mathcal{G}^L = \{G^{L_1}, G^{L_2}, \ldots\}$ and $\mathcal{G}^R = \{G^{R_1}, G^{R_2}, \ldots\}$ are sets of new positions, to which the two players of the game, Left and Right, can move.

To avoid complications with the inductive definition, note that $\mathcal{G}^L$ and $\mathcal{G}^R$ can (potentially) be empty or infinite. In order to start the basis of induction, the empty set is used, to define the so-called *Endgame*[2]:

$$0 = \{\varnothing | \varnothing\} = \{|\},$$

No player may move in the Endgame; the first player to move thus always loses.

One may view independent local subgames on the Go board as combinatorial games. To combine the subgames (with finished analyses), we need the notion of a (combinatorial) *game sum*:

**Definition 2.** The sum $G + H$ of games $G = \{A, B, C, \ldots Z | a, b, c, \ldots z\}$ and $H = \{A', B', \ldots Z' | a', b', \ldots z'\}$ is the game

$$\{A + H, \ldots Z + H, G + A', \ldots G + Z' | a + H, \ldots z + H, G + a', \ldots G + z'\}$$

This was an approach of Martin Müller to solve endgames in Go (Chapter C).

[2] This is just a coincidence in names and an example of John Conway's humor.

# Part I

# Perfect Endgames of Perfect-Information Games

# A. Setting the Scene for Perfect Information

> In order to improve your game, you must study the endgame before everything else. For whereas the endings can be studied and mastered by themselves, the middle game and opening must be studied in relation to the end game.
>
> José Raúl Capablanca

For many games, solving their late stages (*endgames*) can be done in a dynamic way, by the in-play analysis. In other words, we are often able to postpone the computation of the endgame strategy until the endgame itself is actually reached in the real play.

On the other hand, endgames can be also pre-computed (often up to a perfect play) and cached for later. Such a "memoization" approach is advantageous in popular games such as Chess.

## A.1 Extensive Form for Perfect-Information

> Trees that are slow to grow bear the best fruit.
>
> Molière

Games with perfect information can be naturally represented with a *game tree*:

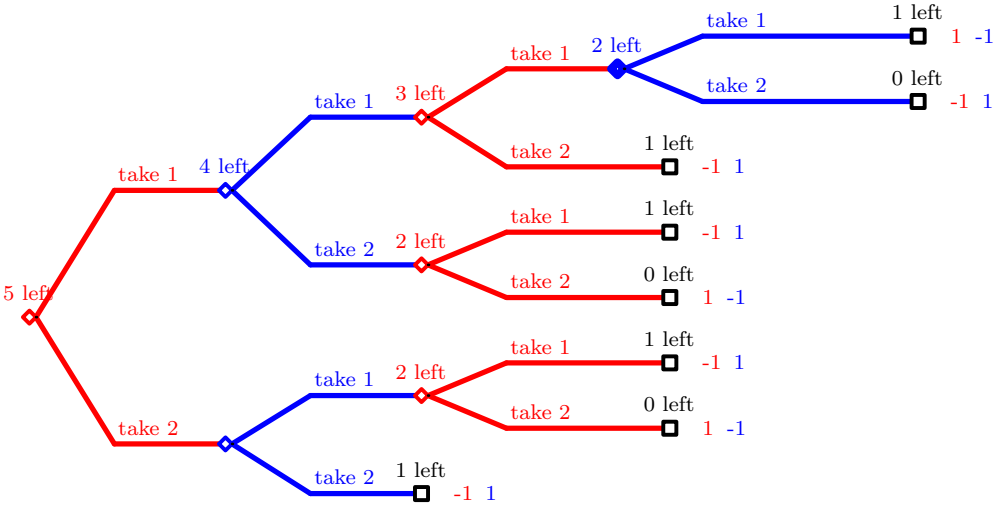



Figure A.1: Game tree of (1,2)-Nim with 1 heap of 5 stones (rendered by `Gambit` using the included example `nim.efg`)

The representation with a (directed) tree (instead of a pay-off matrix) is called an extensive form.

Formally, an *extensive-form game (EFG)* for a perfect-information game (Osborne and Rubinstein 1994, p. 200) consists of:

- ♠ a finite set of *players* $P$,
- ♠ a finite set $H$ of all possible *histories* (i.e., paths from the root to vertices of the game tree) such that
  - ♢ each history consists of individual *actions* (i.e., tree edges),
  - ♢ $\varnothing \in H$ (i.e., the tree root),
  - ♢ relation $h \sqsubseteq h'$ means that *history $h$ is a prefix (ancestor) of $h'$*,
  - ♢ if $h' \in H$ and $h \sqsubseteq h'$, then $h \in H$,
  - ♢ set $Z \subseteq H$ is the set of *terminal histories*, i.e., histories that are not prefixes of any other histories.
- ♠ the set of available actions $A(h) = \{a : (h, a) \in H\}$ for every (non-terminal) node $h \in H \setminus Z$ (i.e., edges to children),
- ♠ a function $p\colon P \to H \setminus Z$, which assigns an *acting player* $p(h)$ to each non-terminal history $h$.
- ♠ a *utility function* $u_i\colon Z \to \mathbb{R}$.

We'll also need to know a strategy, the expected utility and a best response (BR).

- ♠ A (behavior) *strategy* $\sigma_i$ of player $i$ defines a probability distribution over actions $A(h)$ at every node $h \in H$ with $p(h) = i$ (i.e., everywhere where player $i$ acts). The set $\Sigma_i$ contains all possible strategies of player $i$, and by $\sigma_{[S \leftarrow \sigma^*]}$ we denote the combined strategy of $\sigma^*$ (within part $S$) and $\sigma$ (elsewhere).

  Additionally, a strategy can be *pure* (strictly one action is always chosen) or *mixed* (a probability distribution over pure strategies).
- ♠ A *strategy profile* $\sigma$ is a tuple of players' strategies: $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{|P|})$. The set of all possible strategy profiles is denoted as $\Sigma$ and it is the Cartesian product $\Sigma = \prod_{i \in P} \Sigma_i$.

  By $\sigma_{-i}$ we denote the strategy profile of $i$'s opponents:

  $$\sigma_{-i} = (\sigma_1, \sigma_2, \ldots \sigma_{i-1}, \sigma_{i+1}, \ldots \sigma_{|P|})$$
- ♠ We use the symbol $\pi^\sigma$ (resp. $\pi^\sigma_i$) to evaluate the overall (resp. player $i$'s) probability corresponding to a strategy profile $\sigma$. The probability $\pi^\sigma(h)$ of reaching node $h$ can be decomposed to each player's contribution as

  $$\pi^\sigma(h) = \prod_{i \in P} \pi^\sigma_i(h)$$

  Furthermore, the probability $\pi^\sigma_{-i}(h)$ (shortly $\pi_{-i}(h)$) is the product of all players' contribution, except for $i$'s one:

  $$\pi^\sigma_{-i}(h) = \prod_{j \neq i} \pi^\sigma_j(h)$$

♠ Given a strategy profile $\sigma$, the *expected utility* $u_i(\sigma)$ for player $i$ is the sum of utilities in the leaves, weighted by the probabilities of reaching them:

$$u_i(\sigma) = \sum_{z \in Z} \pi^{\sigma}(z) \cdot u_i(z)$$

♠ Player $i$'s *best response* $BR_i(\sigma_{-i})$ (briefly $BR_i(\sigma)$) is a strategy $\sigma_i \in \Sigma_i$ maximizing his expected utility against other players:

$$u_i(\sigma) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$$

Given a fixed $\sigma_{-i}$, one technique to find some BR is to recursively traverse the game tree and pick the most valuable action at each node (i.e. the child with the maximal expected utility).

**Note**

There may be several best responses due to a multiplicity of actions with the maximum value. Let us assume $BR_i(\sigma_{-i})$ denotes any of them: this simplification is not harmful at all, since each BR leads to the same (maximal) value of the total utility.

♠ A *Nash equilibrium (NE)* is a strategy profile where players are playing best responses against each other. Formally, $\sigma$ is an NE if every $i \in P$ has

$$u_i(\sigma) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$$

(i.e., no player has any incentive to deviate from his strategy).

♠ For a given profile $\sigma$, its exploitability

$$\epsilon_\sigma = \frac{u_1(CBR(\sigma_2), \sigma_2) + u_2(\sigma_1, CBR(\sigma_1))}{2}$$

expresses how much $\sigma$ loses to a worst-case opponent, averaged over both players. An NE has an exploitability of 0, because it is unexploitable.

Finally, two common properties of games are:

♠ *two-player* property: $P = \{1, 2\}$

♠ *zero-sum* property: For any profile $\sigma \in \Sigma$, players' utilities cancel out

$$\sum_{i \in P} u_i(\sigma) = 0.$$

In particular, two-player zero-sum games capture the antagonistic behavior: what one player gains, the other one loses. Typical examples are games such as RPS, Chess, Go, Heads-Up Poker, etc.

## A.2 Subgames

> At that time I did not suspect that it ["An Oligopoly Model with Demand Inertia" (Selten 1968)] often would be quoted, almost exclusively for the definition of subgame perfectness.
>
> Reinhard Selten

In the perfect-information setting, a *subgame* is a game corresponding to a subtree of the game tree. Specifically, any node in the tree induces its own subgame:

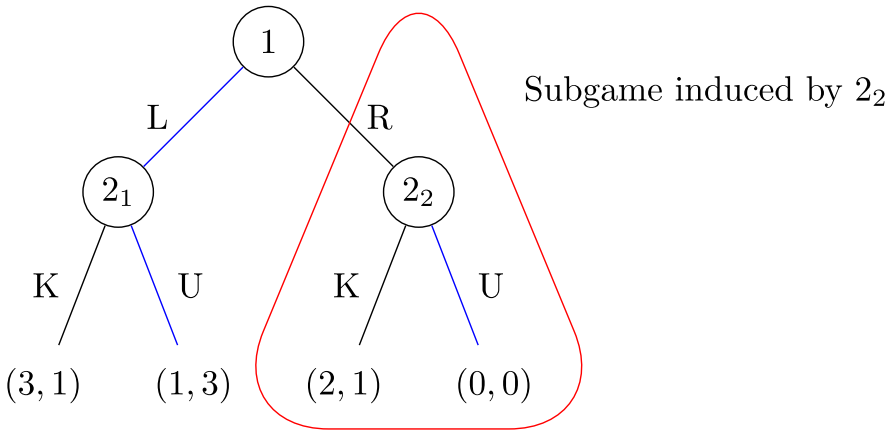



Figure A.2: Subgame induced by node $2_2$

**Note**

For the upcoming proofs, we define the notation for *the restriction of the expected utility* on a subgame rooted in node $h \in H$:

$$u_i^h(\sigma) = \sum_{z \in Z,\, h \sqsubseteq z} \pi^\sigma(h, z)\, u_i(z)$$

where $\pi^\sigma(h, z)$ is the probability of reaching $z$ from $h$ under $\sigma$.

If we fix the strategies of opponents, re-solving subgames can only improve our strategy:

**Theorem 1** (subgame re-solving and utility)**.** *The re-computed sub-strategies can be "placed back" into the full-game strategy, without decreasing the total expected utility.*

*Proof.* Let $\sigma$ be any full-game strategy and $\sigma_S$ be its restriction to a subgame $S$ rooted at node $r$. Assume we found a better (or even optimal) strategy $\sigma^*$ for the subgame $S$, i.e., the new expected utility of any player $i$ is better within $S$:

$$u_i^r(\sigma^*) \geq u_i^r(\sigma_S) \tag{A.1}$$

The new re-combined strategy $\sigma' = \sigma_{[S \leftarrow \sigma^*]}$ cannot decrease $i$'s utility:

$$\begin{aligned}
u_i(\sigma') &= \sum_{z \in Z} \pi^{\sigma'}(z)\, u_i(z) = \sum_{z \in Z \cap S} \pi^{\sigma'}(z)\, u_i(z) + \sum_{z \in Z \setminus S} \pi^{\sigma'}(z)\, u_i(z) \\
&= \left[ \pi^{\sigma'}(r) \sum_{z \in Z \cap S} \pi^{\sigma^*}(r, z)\, u_i(z) \right] + \sum_{z \in Z \setminus S} \pi^{\sigma'}(z)\, u_i(z) \\
&= \pi^{\sigma}(r)\, u_i^r(\sigma^*) + \sum_{z \in Z \setminus S} \pi^{\sigma}(z)\, u_i(z) \\
&\overset{\text{(A.1)}}{\geq} \pi^{\sigma}(r)\, u_i^r(\sigma_S) + \sum_{z \in Z \setminus S} \pi^{\sigma}(z)\, u_i(z) \\
&= \sum_{z \in Z \cap S} \pi^{\sigma}(z)\, u_i(z) + \sum_{z \in Z \setminus S} \pi^{\sigma}(z)\, u_i(z) = u_i(\sigma)
\end{aligned}$$

Therefore, combining the new $\sigma^*$ with the full-game $\sigma$ does not decrease the total expected utility. ■

Note that Theorem 1 also extends to imperfect-information subgames (Definition 4, Section E.5). Nevertheless, the perfect-information setting offers one additional advantage: no opponent can exploit the newly computed sub-strategy by adjusting his play, either in the trunk or in the subgame.

**Theorem 2** (unexploitable subgame re-solving)**.** *Assume we are given a two-person zero-sum game, its subgame $S$ rooted at node $r$ and a fixed player $i$. Let $\sigma_i$ be the original strategy, $\sigma_i^*$ the re-computed strategy that only differs within $S$, let $\sigma_{-i} = BR(\sigma_i)$ and $\sigma^*_{-i} = BR(\sigma_i^*)$ be opponent's corresponding best responses, and let $\sigma = (\sigma_i, \sigma_{-i})$ and $\sigma^* = (\sigma_i^*, \sigma^*_{-i})$ be the resulting profiles.*

*If $u_i^r(\sigma) \leq u_i^r(\sigma^*)$ (player $i$'s worst-case utility does not decrease in the subgame), then at every node $h \in H \setminus S$ we also have $u_i^h(\sigma) \leq u_i^h(\sigma^*)$ (player $i$'s worst-case utilities outside the subgame do not decrease).*

*Proof.* By definition, all best responses produce equal expected utilities, because they all maximize opponent's value. We can therefore do all calculations of utilities with respect to an arbitrary best response, e.g., the standard BR retrieved by the *backward induction*[1].

Let $T_h$ be the subtree rooted at $h$. If $T_h$ does not contain $S$ (i.e., $h \not\sqsubseteq r$), then the best responses $\sigma_{-i}$ and $\sigma^*_{-i}$ are identical within $T_h$, trivially implying $u_i^h(\sigma) = u_i^h(\sigma^*)$. On the other hand, if $T_h$ does contain $S$, we will prove the inequality by mathematical induction on the height of $T_h$.

For the base case, the minimum-height tree containing $S$ is the subgame tree $T_r$ itself, where the inequality holds by assumptions. For the inductive step, let node $h$ has $k$ children $h_1, \dots h_k$, and without loss of generality $T_{h_1}$ contains $S$. By the induction hypothesis, $u_i^{h_1}(\sigma) \leq u_i^{h_1}(\sigma^*)$. Because the subtrees of the remaining children do not contain $S$ (i.e., $h_j \not\sqsubseteq r$ for $j = 2, \dots, k$), $\sigma$ and $\sigma^*$ are again identical there. Hence, $u_i^{h_j}(\sigma) = u_i^{h_j}(\sigma^*)$ for $j = 2, \dots, k$, and we get

$$u_i^{h_j}(\sigma) \leq u_i^{h_j}(\sigma^*), \quad j = 1, \dots, k \tag{A.2}$$

[1] Recursively select the action with the maximum utility, or any of them in case there is more than one maximal.

If $i$ is the acting player of $h$ (i.e., $p(h) = i$), then by expressing utilities as weighted sums of children's utilities, we have

$$\begin{aligned} u_i^h(\sigma) &= \sum_{j=1,\dots,k} \pi^{\sigma}(h, h_j)\, u_i^{h_j}(\sigma) \\ &= \sum_{j=1,\dots,k} \pi^{\sigma^*}(h, h_j)\, u_i^{h_j}(\sigma) \\ &\overset{\text{(A.2)}}{\leq} \sum_{j=1,\dots,k} \pi^{\sigma^*}(h, h_j)\, u_i^{h_j}(\sigma^*) = u_i^h(\sigma^*) \end{aligned}$$

where the second equality holds because $\sigma$ and $\sigma^*$ are identical outside the $S$. If otherwise the opponent is acting (i.e., $p(h) \neq i$), he aims to choose an action with minimal $i$'s utility (due to the zero-sum property):

$$u_i^h(\sigma) = \min_{j=1,\dots,k} u_i^{h_j}(\sigma) \overset{\text{(A.2)}}{\leq} \min_{j=1,\dots,k} u_i^{h_j}(\sigma^*) = u_i^h(\sigma^*)$$

So in both cases $u_i^h(\sigma) \leq u_i^h(\sigma^*)$. ■

**Corollary 1.** *If a player employs a re-computed subgame strategy, the opponent has no way to adjust his strategy, in order to increase his own overall utility.*

*Proof.* Apply Theorem 2 to the root of the game (i.e., $h := \varnothing$). ■

This means that we can deal with subgames of perfect-information games separately, either by pre-computation or by dynamic re-solving. This approach has met success, e.g., in Checkers with $5 \times 10^{20}$ states, which would be otherwise intractable: Checkers has been solved both in terms of the game's value and an optimal NE strategy (Schaeffer et al. 2007).

Moreover, we may freely combine the newly found strategies with the original ones: Theorem 1 guarantees they won't be inferior, and Theorem 2 guarantees their unexploitability by the opponent.

**Note**

The situation in imperfect-information setting is different: in fact, Claim 3 (p. 50) proves that a re-solved subgame strategy can indeed be exploited.

## A.3 Working Examples

> Few things are harder to put up with than the annoyance of a good example.
>
> Mark Twain

Subgame solutions are used particularly in perfect-information games with an extensive form such as Chess or Go. There, the endgame technique has been used for long time as one way to defeat the colossal size of the game tree. In these

specific domains, endgame solving has additional significance, as it improves the playing quality of agents as well.

In Chapter B, we will see the power of subgame pre-computation with the example of Chess solutions to endings, stored in so-called *endgame tablebases*. They are used in real world to aid professional Chess players, either in proving their guaranteed victory or in analysing past games. Moreover, since tablebases are mathematically proven to be optimal, they provide a glimpse into the world of "perfect Chess" played by "all-knowing super-players".

In contrast, Chapter C demonstrates how the in-play approach to endgames is beneficial in the game of Go. Once reaching the late stage, the board is partitioned into distinct parts. The corresponding (and provably independent) subgames are re-solved "on the fly", just to be afterwards combined using the *combinatorial game theory*.

Chapter D reviews the modern approach to computer Go used by Google DeepMind: their Go program *AlphaGo* combines *Monte Carlo tree search* with *neural networks* to treat each position as if it were an endgame. This in particular means that several moves into the future are simulated and the corresponding part of the game tree is unrolled and possibly expanded. The rest of the tree is discarded, effectively leaving only the relevant subgame for the oncoming play.

# B. Chess Endgames

## B.1 What are Endgames in Chess?

> Studying openings is just memorizing moves and looking for traps. Studying the endgame is Chess.
>
> Josh Waitzkin

The notion of *Chess endgame* has no strict definition and differs by various authors. The common sense says it begins when only few pieces are left. Here are another examples of endgame definitions:

- ♠ positions in which each player has less than thirteen points in material (not counting the king) (Speelman 1981, pp. 7–8)
- ♠ positions in which the king can be used actively (but there are some famous exceptions to that) (Speelman 1981, pp. 7–8)
- ♠ positions having four or fewer pieces other than kings and pawns (Minev 2004, p. 5)
- ♠ positions without queens (Fine 1952),
- ♠ positions when each player has less than a queen plus rook in material
- ♠ positions when the player who is about to move can force a win or a draw against any variation of moves (Portisch and Sárközy 1981)
- ♠ positions with these three characteristics (Alburt and Krogius 1999):
  (a) endgames favor an *aggressive* king;
  (b) *passed pawns*[1] increase greatly in importance;
  (c) *Zugzwang*[2] is a factor in endgames (rarely in other stages of the game).
- ♠ *Not Quite an Endgame (NQE)* are positions where each player has at most one piece (other than kings and pawns), and positions with more material where each player has at most two pieces (Flear 2007, pp. 7–8)

Nevertheless, endgames have one thing in common: the complexity of the board is often manageable by computers, making it feasible to compute the perfect strategy.

[1] a pawn with no opposing pawns to prevent it from advancing to the $8^{\text{th}}$ rank (i.e., there are no opposing pawns in front of it either on the same file or on an adjacent file)

[2] a disadvantageous situation in Chess wherein one player has to make a move, even though he would prefer to pass and not move

## B.2 Endgame Tablebases

> Studying the endgame is like cheating.
>
> Michael Frannett

An *endgame tablebase* is a database of pre-calculated, exhaustively analyzed Chess endgames stored as tables of positions together with the best consequent moves. Upon reaching an arbitrary tablebase position, the database thus provides an optimal strategy to play perfect Chess.

Tablebases are designed for a given set of pieces, e.g., ♔King and ♕Queen versus ♚King (KQ-K). There are 3 basic steps in their process of creation:

1 **Generation**: Computer generates all legal positions for the given pieces. For each position, the tablebase evaluates the situation separately for White-to-move and Black-to-move.

In the case of KQ-K, the number of positions amounts to $\approx 40000$. Such number is due to the symmetry argument (Levy and Newborn 2009): assume the ♚ is on square a1, b1, c1, d1, b2, c2, d2, c3, d3, or d4 (see diagram on Figure B.1). Other squares are equivalent by symmetry of rotation or reflection.

Now there are $\approx 60$ remaining squares for the ♔ and at most 62 squares for the ♕. Therefore, there are at most $10 \times 60 \times 62 = 37200$ KQ-K positions.

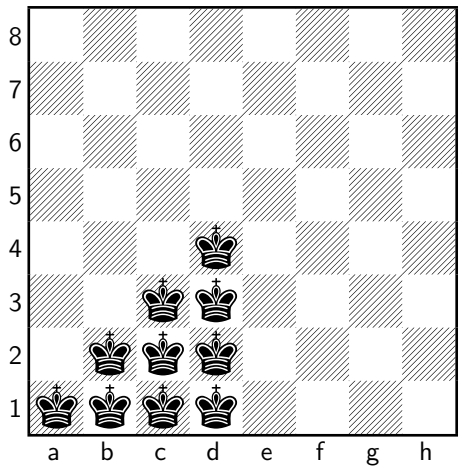



Figure B.1: Non-symmetric positions for ♚ (Levy and Newborn 2009)

Adding one new piece into a *pawnless* endgame multiplies the count of positions by $\approx 60$ (the approximate quantity of unoccupied positions). Pawns would break the front-back and diagonal symmetries, because they care about direction in their moves (Muller 2006).

2 **Evaluation**: The generated positions are evaluated using the *backward induction*[3], which in Chess is also called the *retrograde analysis*. Each position is evaluated as a win or a loss in a certain number of moves. At the end of the retrograde analysis, positions which are not designated as wins or losses are necessarily draws.

[3] *backward reasoning* applied in general game theory, in order to solve easier subgames

Invented by Richard Bellman in 1965, the retrograde analysis faithfully follows the approach of *dynamic programming* (Bellman 1965):

(a) checkmated positions are determined in the beginning

(b) a position is winning in $n + 1$ moves if the player can reach a position winning in $n$ moves (more precisely, where the opponent loses in at most $n$ moves)

Positions are generated in the order of increasing depth to mate (DTM), i.e., the number of moves necessary to force a checkmate.

Alternatively, Tim Krabbé (Krabbé 2014) describes retrograde analysis by generating (from the perspective of White to mate):

(1) a database of all possible positions given the material (see the previous step of Generation),

(2) a sub-database made of all positions where Black is mated,

(3) positions where White can reach mate (DTM = 1),

(4) positions where Black cannot prevent White giving mate next move,

(5) positions where White can always reach a position where Black cannot prevent him from giving mate next move (DTM = 2).

(6) And so on, always a ply[4] further away from mate until all positions connected to mate are found.

By connecting these positions back to mate, the shortest path through the database is formed. Such a path contains perfect play: White moves towards the quickest mate, Black moves towards the slowest mate (which can be a draw or even Black's win).

3 **Verification**: The self-consistency of the tablebase is verified by independently searching for each position (both Black and White to move). The score of the best move has to be in line with the score in the table. As pinpointed by (Bourzutschky 2006), this (seemingly optional) step is important:

> This is a necessary and sufficient condition for tablebase accuracy. Since the verification program was developed independently of the generation program (I don't even have the source code for the generation program) the likelihood of errors is pretty small.

---

Additionally, the computation of tablebases can be simplified if *a priori information* is provided. For instance, a position KRP(a2)-KBP(a3) with pawns blocking each other (diagram in Figure B.2) reduces number of possibilities for the pawns: instead of $48 \times 47 = 2,256$ permutations for the pawns' locations, only single one needs to be considered (Herik, Herschberg, and Nakad 1987).

[4] In standard Chess terminology, one move consists of a turn by each player. Therefore a *ply* in Chess is a half-move. Thus, after 20 moves in a Chess game, 40 plies have been completed: 20 by White and 20 by Black.

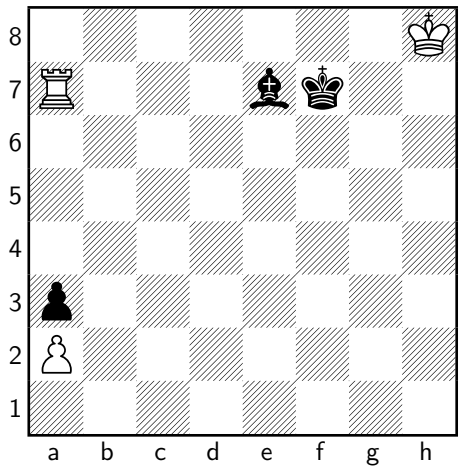


Figure B.2: KRP(a2)-KBP(a3) (Herik, Herschberg, and Nakad 1987)

## B.3 Applications of Tablebases

FRATBOT #1: Mate in 143 moves.
FRATBOT #2: Oh, p**h. You win again!
BENDER: Uh-oh, nerds!

*Futurama*, Season 1, Episode 11

♠ **Complexity of solving Chess**
*Generalized Chess*[5] has been proven to be EXPTIME-complete (Fraenkel and Lichtenstein 1981): it takes exponential time to determine the winner of any position in the worst case. The result, however, gives no lower bound on the amount of work required to solve regular $8 \times 8$ Chess.

Conversely, there has been progress from the other side: as of 2012, all 7 and fewer piece endgames (2 kings and up to 5 other pieces) have been solved (*Lomonosov Tablebases* at http://tb7.Chessok.com/). Evidently, focusing on endgame significantly decreases the complexity; tablebases thus provide "powerful arsenal" to play perfect Chess endings.

♠ **Effects on Chess theory**
Tablebases have enabled significant breakthroughs in Chess endgame theory. They caused major changes in the view on many endgame piece combinations, which were considered to result in a completely different way. Some impressive examples (Wikipedia 2016):

- ♢ KQR-KQR endgames. Despite the equality of material, the player to move wins in 67.74% of positions (Haworth 2001).
- ♢ In both KQR-KQR and KQQ-KQQ, the first player to check usually wins (Nunn 2002, p. 379, 384).
- ♢ Many positions are winnable although at first sight they appear to be non-winnable. For example, the position in Figure B.3 is a win for

[5]Chess played with an arbitrarily large number of pieces on an arbitrarily large chessboard

Black in 154 moves (during which the white pawn is liquidated after around eighty moves).

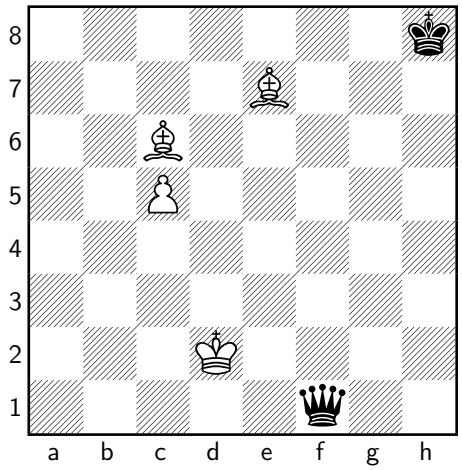



Figure B.3: Black to move wins in 154 moves. (Wikipedia 2016)

♢ In the position of Figure B.4, the White pawn's first move is at move 119 against optimal defense by Black:

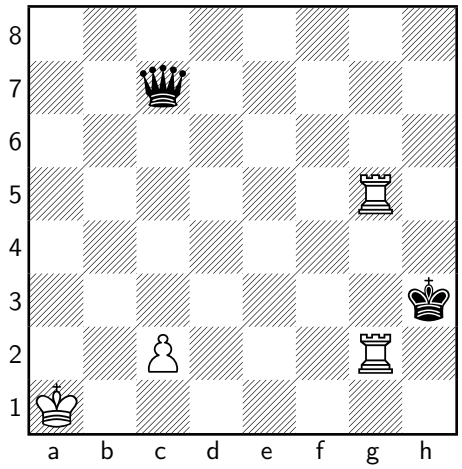



Figure B.4: 119 moves to pawn's first move (Wikipedia 2016)

♠ **The longest checkmates**

The researchers behind the Lomonosov tablebases discovered following endgames, proved to be the longest 7-man checkmates (Lomonosov Tablebases 2014).

A **pawnless** endgame ♖Rook, ♗Bishop and ♘Knight against ♕Queen and ♘Knight (Figure B.5) can be mated after stunning number of 545 moves!

It is the longest pawnless endgame possible with 7 pieces, others are far behind. The closest endgame to this one by length (♖Rook, ♗Bishop and ♘Knight against ♕Queen and ♗Bishop) is much less complex and won by the side with more pieces, not the queen.

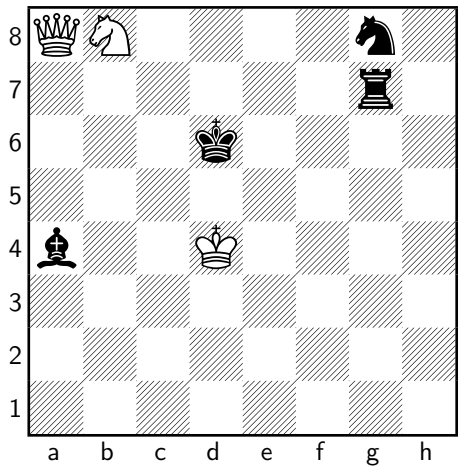


Figure B.5: KRBN-KQN: White mates in 545. (Lomonosov Tablebases 2014)

No human expert and not even the best Chess programs are able to find the winning solution. Top Chess players admit that they fail to understand the logic behind the initial 400 moves. There are even no captures until move 523. The authors attribute this immense complexity to the balance of piece values: 11 against 12, which is the minimal advantage.

One simple idea to create an even more complex endgame **with pawns** is to build on the previous position and make use of pawns' promotion. The ending position of ♖Rook, ♗Bishop and ♘Knight versus ♕Queen and ♙Pawn in Figure B.6 implements this idea:

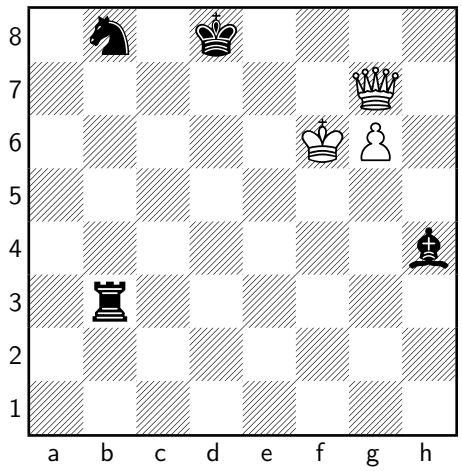


Figure B.6: The longest 7-man checkmate: White forces a mate after 549 moves. (Lomonosov Tablebases 2014)

White promotes his ♙Pawn on the 6$^{\text{th}}$ move. Strangely, the promotion is to ♘Knight instead of ♕Queen, so as to check the ♚King and avoid losing the ♕Queen (Figure B.7). Afterwards, the familiar KRBN-KQN ending emerges and checkmate is forced by White in 544 moves.

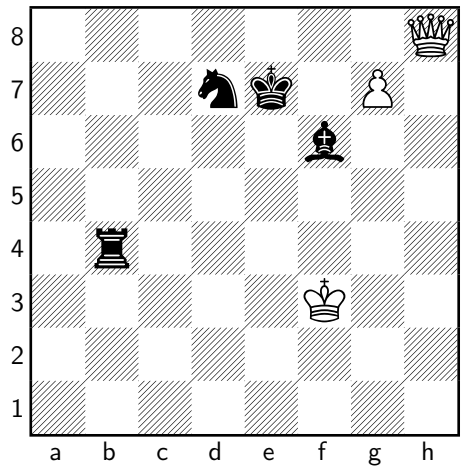


Figure B.7: Mate in 544: g8 = ♘ (Lomonosov Tablebases 2014)

The KRBN-KQP(g6) position is the absolute winner of "the competition for the longest 7-man checkmate". It took 3 additional years to prove it, because pawn endings also require *minor endings* (positions after capturing pieces or promoting pawns), posing great challenges to computational power, storage and data organization.

(Lomonosov Tablebases 2014) show another 6 monstrously huge endgames, each with over 500 forced moves to checkmate. The authors remark on the surprising gap in number of moves between the "hateful eight" KRBN-KQN and derived positions, and the next ($9^{\text{th}}$ and $10^{\text{th}}$) longest 7-piece endgames: KBNP-KBP and KNNP-KRB require only 346 moves to mate.

And what about the view on the longest 8-man endgames? The longest 6-man mate takes 262 moves (KRN-KNN). One more piece (7-man endings) doubles the maximum depth. Promisingly, the longest 8-man ending may reveal a mate in over 1000 moves. However, there are much more positions with relatively balanced strengths on both sides, making 8-man endgames much richer but also more complicated. On top of that, the complexity of computation is simply ludicrous: one would need about 10 PB of disk space and 50 TB of RAM. As of 2014, only top 10 supercomputers can solve this problem. (Lomonosov Tablebases 2014)

**Note**

These examples of enormously lengthy checkmates call into the question the *50-move rule*: a draw claimable by any player, if no pawn has been moved and no capture has occurred in the last 50 moves. In situations with a checkmate forced after a huge yet proven number of moves, a draw would be incorrectly announced, in spite of a guaranteed victory.

# C. Go Endgames Using Ad-Hoc Mathematics

> Combinatorial game theory captures an essential part of what Go is about. I think that in one form or another, it will become a key component of all successful future Go programs.[1]
>
> Martin Müller 1995

The ancient game of Go offers astounding complexity: the board size, the quantity of possible moves and the average game length produce an astronomical number of valid positions.

More than in any other classic game, human intuition plays a vital role for a successful play: Go offers great richness of geometrical, combinatorial and logical structure, ready to be exploited by human players.

Quoting Google DeepMind, the game of Go is agreed to be "the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves" (Silver et al. 2016).

**Note**

This chapter is based on the work of Martin Müller (Müller 1995) submitted as a doctoral thesis at ETH Zürich. Since the focus of our survey are imperfect-information games rather than Go, this chapter mostly summarizes his work.

Martin Müller reported in 1995 that (at that time) Go programs "comprehend only the most basic concepts of Go" and that "to make progress, [he feels] it is necessary both to encode more Go-specific knowledge and to push forward the application of theories such as combinatorial game theory to Go" (Müller 1995). Here follows the overview and the results of his endeavors.

## C.1 Why Focus on Go Endgames?

(Müller 1995) mentions the following advantages of Go endgames for research:

- ♠ The situation towards the end of game becomes clearer. This simplification helps to study Go in an easier and more controlled way.
- ♠ Some endgame positions allows for an exact solution within reasonable time.

[1] This prognosis is in fact misaligned: AlphaGo, the most successful Go program designed so far, uses almost no Go-specific knowledge. To compare Müller's prediction with current trends, consult the Note on page 34 and Chapter D.

- ♠ Understanding parts of board improves the understanding of the whole game. Such partial evaluations aids various heuristics for solving the full board. Humans reason similarly: by observing the score since the early midgame, they decide based on such a heuristic analysis (Shukaku 1985).
- ♠ Methods and techniques for partitioning, searching and scoring during endgame are frequently also applicable to the midgame and opening.
- ♠ Endgames provide a simplified, more manageable sub-domain allowing the use of stronger theoretical models than the larger, more general problem.

## C.2 Partitioning into (Sub)games and Playing The Game-Sum

> Divide et impera.
>
> Philip II of Macedon

A typical Go position contains several *subgames* (local scenes) suitable for independent analysis. Initially, the subgames have no influences on one another due to their mutual distance. Nevertheless, as the game progresses, they gain the crucial role thanks to a more distinct board partitioning via walls of safe stones:

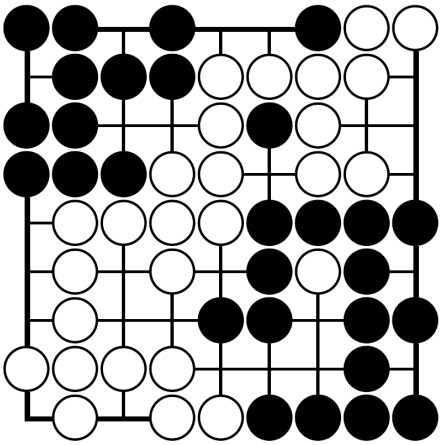


Figure C.1: An immortal wall enabling an exact analysis during late endgame (Müller 1995)

No move can have any influence across these "immortal" walls and significant parts of the board belong with certainty to one of the players. The connected components of remaining stones define local subgames independent from each other.

In the opening and midgame, partitioning can be only approximate. On the other hand, the partitioning in the endgame becomes more precise: the statuses of all big groups have been settled, and the outlines of territories are clear. If each local game is simple enough for complete analysis (such as in Figure C.1), CGT can compute an optimal move for the full board position.

Now follows the general procedure of (Müller 1995) for playing Go as a sum of local games:

1. Board partition: find safe blocks, safe territories, and local areas.
2. Generate local game trees in each area.
3. Evaluate local terminal positions.
4. Transform local game trees into combinatorial games and simplify them.
5. Find an optimal move in the CGT sum-game and play it.

Müller proposes heuristic algorithms for playing the entire game, and exact algorithms for late endgame positions.

Undoubtedly, the task of solving endgames in 1995 (when computer Go was at its infancy) already played an important role.

## C.3 Combinatorial Game Theory for the Game of Go

> You get surreal numbers by playing games.
> I used to feel guilty in Cambridge that I spent all day playing games, while I was supposed to be doing mathematics. Then, when I discovered surreal numbers, I realized that playing games is math.
>
> ---
>
> John Conway

(Müller 1995) suggests to replace the "standard model" of computer Go by a sum-game model. Here follows some general benefits and problems of the sum-game model for computer Go, as listed the work:

Benefits

+ suitability to the knowledge and style of Go programs (at that time): local fighting, surrounding territories, etc.
+ reusing of local game analysis for a high-quality game play
+ simplifying move generation and evaluation of positions
+ translating Go terms into the theory, without need for separate programming
+ evaluating opponent's endgame moves
+ assessing sufficiently good moves even with a *reduced* local game database
+ computing the game-theoretic value of a position long before the end
+ opportunities for parallelism: independent local searches, evaluations and operations on mathematical games, etc.
+ perfect computer play in late endgame

Obstacles

- required accurate board partition and dependency recognition
- common issues of selective search: misleading evaluation due to missing crucial moves in complicated games
- the lack of long-range full-board planning: "This is probably not a big issue until programs reach amateur Dan or even professional level." (Müller 1995)

Additionally, a strict endgame analysis is impossible, when any of these limits is reached:

- **partition:** There is insufficient amount of blocks that can be proven safe. Some of the areas are hence too large for the complete search.
- **summation:** No move with dominating incentive[2] exists, and both summing and partial search would take too long.
- **Ko:** CGT relies on the independence of local subgames. In the case of Ko, the independence is broken: a locally bad move may be globally best if it serves as a Ko threat (recall Section 0.3). Müller mentioned in 1995 there was ongoing research to generalize the theory for handling Kos.
- **unneccessary complexity in "straightforward" situations:** In cases where the focus of play is straightforward (i.e., only one local situation is relevant), CGT introduces additional complexity. It investigates moves for both players in this and every other position.

## C.4 Board Partition and Subgame Dependencies

> Independence? That's middle class blasphemy. We are all dependent on one another, every soul of us on Earth.
>
> George Bernard Shaw

In addition, (Müller 1995) mentions several further obstacles of the sum-game model, caused by the *heuristic* board partition:

- **the impossibility of a perfect precise split:** During the opening and midgame, there are almost no surrounded spaces. Moreover, the surrounding stones would not be invulnerable yet. Heuristics is needed for board partition to split the imperfectly surrounded areas.
- **the infeasibility of exhaustive search in still large subgames:** The program is obliged to give a move in few seconds or minutes. However, there still remain areas intractable for an exhaustive search.

[2] the improvement made by moving in a game – Müller 1995, Chapter 3, p. 38

In such subgames, Müller employs *selective search* method. He limits the number of generated moves and stops the search before reaching a terminal position. For each local game, nodes to expand need to be decided and *expert modules for search and evaluation* need to be selected. A post-processing stage handles detected dependencies between games.

- **the dependencies between the resulting subgames, which arise with the previously mentioned heuristics:** The effect of such dependencies differs widely. Often it is so small that independence is a useful approximation. However, in cases when a move works as a *double threat*, dependency analysis is crucial.

  Trivial strategies to overcome dependencies involve:

  ♢ ignoring the dependency, as if the games were independent;

  ♢ proving that the dependency does not affect the value of the sum, or the play of the sum game;

  ♢ merging mutually dependent local games, then re-search the combined game, possibly using previously generated information on single games;

  ♢ analyzing the interaction between local games, then use a specialized theory to compute the joint game value;

Hence, the sum-game model makes sense mostly during the endgame phase. This general principle is as well applicable in poker: we wait until the late stage of the game, when there is more impact in re-solving the reached endgames.

## C.5 Local Search and Evaluation in the Endgame

> True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information.
>
> Winston Churchill

Once the board is partitioned, the algorithm of (Müller 1995) performs following steps to convert a Go endgame into a combinatorial game:

1. Generate local game trees: all legal moves are generated from both players' point of view, with exceptions of:

   (a) *pruning rules*[3], e.g., pruning dominated subgame trees: the evaluation of nodes allows for pruning moves dominated by other moves.

   Moves with a dominating alternative are labelled as locally bad moves.

   (b) *termination rules*[4] for static evaluation of a position, without further tree expansion

[3] analogous to *policy networks* of AlphaGo in Chapter D.

[4] analogous to *value network* of AlphaGo in Chapter D.

2. Score local terminal positions:

   (a) *scoring* (Section 0.3) comes in two widely accepted variants:

   ♢ Chinese variant, which counts stones and empty points belonging to either color,

   ♢ Japanese variant, which counts territory and prisoners.

   Both variants have straightforward implementation because safe stones, dead stones, territories and neutral points are known exactly in the endgame.

   (b) *terminal positions* can be recognized by

   ♢ no more legal moves,

   ♢ no more good moves,

   ♢ the value of position already known from the transposition table, pattern, or local position database.

   A position is additionally considered terminal, once we can assess its value from another source. Such a value can be any value in the sense of mathematical games.

3. Evaluate local games as mathematical games. Explicitly, evaluate terminal positions, and back up the values in the tree, resulting in a mathematical-game evaluation of each node.

   Even though the program reaches a solved position, (Müller 1995) notes that a "suicidal" opponent may later cause great troubles, by endangering his immortal stones. Such an act would violate the assumptions on board partitioning and give rise to new unexpected positions with a rather complicated optimal play.

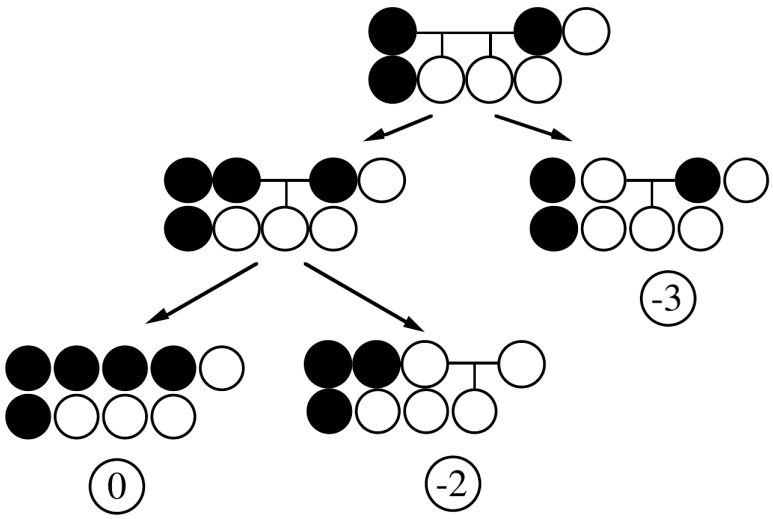



Figure C.2: Local game tree with evaluation of terminal nodes (Müller 1995)

4. Select an option in the resulting (abstract) sum game.

5. Translate the chosen option from the abstract game into a corresponding move in Go. This move is taken as the first move with sufficient incentive.

## C.6 Storing Search Results into Database

> I'm blessed with very fast memorization skills, so I don't have to read it too far in advance.
>
> Jaimie Alexander

Search and analysis produce the complete description of possible endgame plays, facilitating the perfect strategy. The results are stored in a *database of local positions*. Local subgames during a live play are then matched one-to-one to a set of database positions. The value of a full board position is the sum of local position values and the algorithm for sum-game evaluation selects the best available response.

A version with lower memory requirements is implemented, too. This is achieved by saving only a selection of positions, and re-doing the search, in case the subgame is missing in the database.

An important question is what to store. Table C.1 gives some possible answers:

| **Type of database** | **Content** |
|---|---|
| Full | → every position discovered during exhaustive search |
| Color-complete (Black or White) | → every position reachable by non-dominated moves of the color and arbitrary opponent's moves (pruned bad moves of the color) |
| Optimal | → at least one position corresponding to every non-dominated option of every reachable position (guaranteed optimal play from every position in database) |
| Sufficiently good | → guaranteed optimal score from a starting position (might fail to exploit some opponent mistakes) |

Table C.1: Possible database variants of local games (Müller 1995)

One needs to make a trade-off between re-computation time and storage space. A sensible solution is to store only complicated moves in the database, and re-calculate the rest when necessary. It is a good idea to save only the information whether a move is locally good or bad, but discard all *refutations*[5]. Such a type of database is robust against a good opponent; however, it suffers from a play of a locally bad opponent. This rare situation would call for the re-computation of a subtree in order to find a refutation.

The number of possible moves in an $n$-point area is approximately $2n$ ($n$ for each player), and such a play generates an $n-1$ point area. Thus, the size of the game tree is approximately $2n \cdot 2(n-1) \cdot \ldots \cdot 2 = 2^n n!$ nodes. Such a combinatorial explosion makes even fairly small endgames prohibitively expensive.

[5] trees proving that bad moves are inferior

Müller overcomes this hardness with a transposition table. A *transposition table* detects identical board positions, reducing the size of the search space from $\approx 2^n n!$ to $3^n$ states (see the comparison in Table C.2).

| $n$ | $2^n n!$ | $3^n$ |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 8 | 9 |
| 3 | 48 | 27 |
| 4 | 384 | 81 |
| 5 | 3840 | 243 |
| 6 | 46080 | 729 |
| 7 | 645120 | 2187 |
| 8 | 10321920 | 6561 |
| 9 | 185794560 | 19683 |
| 10 | 3715891200 | 59049 |

Table C.2: Reduction of search space by transposition table

## C.7 Pattern Learning

To understand is to perceive patterns.

Isaiah Berlin

Pattern recognition is a key component of the AlphaGo program (Chapter D). Arguably, the program's success may also be attributed to learning Go patterns from human expert plays.

For comparison, (Müller 1995) hails computer Go as *a vehicle for research in visual perception, machine learning and neural networks* (advocating to Enderton 1991; Schraudolph, Dayan, and Sejnowski 1994; Stoutamire 1991; Wilcox 1979).

Being introduced just and only to Go rules, learning programs can typically pick up basic Go principles, such as saving a stone from capture, and familiarize themselves with low-level game concepts. However, these concepts and principles have already been integrated into other far superior *expert-based programs*.

Neural networks are therefore more adapt at playing locally good shapes, but possess no idea when to play it (this part has changed thanks to AlphaGo). These systems are vulnerable to other Go agents with better knowledge of tactics.

Conclusively, (Müller 1995) suggests *pattern matching* as a promising means for advancements in computer Go.

When patterns are broken, new worlds emerge.

Tuli Kupferberg

## C.8 Contributions of Combinatorial Game Theory to Go Endgames

> You can retire from a job, but don't ever retire from making extremely meaningful contributions in life.
>
> Stephen Covey

Müller's doctoral thesis acclaims these contributions to computer science:

- ♠ "Scientists are fascinated by problems which can be stated simply, yet are hard to solve. Computer Go is a prime example. We have brought the divide-and-conquer approach, a fundamental paradigm of computer science, to bear on computer Go."
- ♠ "The application of a sophisticated mathematical theory to computer Go provides an example of algorithms for a nontrivial decomposition of a complex problem."

These are contributions specific to computer Go:

- ♠ "We have implemented a *late endgame player*, a niche where program play surpasses human play in both speed and exactness. We did this by applying concepts from CGT to Go. The program plays a wide variety of'late endgame positions perfectly."
- ♠ "We have developed algorithms for board partition and dependency analysis."

In conclusion, the work employs a technique of applying an elaborate mathematical theory (CGT) to deal with endgames. In particular, CGT is used to "connect" individual relevant subgames, which may be solved independently.

As we will see later, the situation in the case of Poker is slightly more delicate: the imperfect-information property prevents us from using an immediate divide-and-conquer approach. Instead, we will *augment* the information, by saturation with appropriate game states (Definition 3 in Section E.5).

**Note**

It is also interesting to compare the method of (Müller 1995) with the modern approach of *AlphaGo*.

The ad-hoc Go-specific knowledge is replaced with general-learning neural networks and the probabilistic *Monte Carlo Tree Search* algorithm. Such a combination has the capability to surpass professional human players at the highest ranks.

What's more, this solution can be adapted to other games without substantial modifications. With enough data, the system can be trained without any need to imbue it with game-specific knowledge. Follow Chapter D for more details.

# D. Go Endgames Using Neural Networks

> Creativity involves breaking out of established patterns in order to look at things in a different way.
>
> Edward de Bono

As of today, over 20 years have passed since Müller's work on computer Go (Chapter C). *DeepMind*, a London-based AI start-up recently acquired by Google, has developed the strongest computer Go program so far: *AlphaGo*.

**Note**

This chapter is based on the article "Mastering the game of Go with deep neural networks and tree search" (Silver et al. 2016), which documents the internals of AlphaGo.

For more details, also consult my presentations on AlphaGo given at:

- ♠ Spring School of Combinatorics 2016 (Charles University)
  `http://www.slideshare.net/KarelHa1/mastering-the-game-of-go-with-deep-neural-networks-and-tree-search-presentation`
- ♠ Optimization Seminar (Charles University)
  `http://www.slideshare.net/KarelHa1/alphago-mastering-the-game-of-go-with-deep-neural-networks-and-tree-search`
- ♠ Distributed Computing group (ETH Zürich)
  `http://www.slideshare.net/KarelHa1/alphago-for-disco-group-of-eth-zurich`

## D.1 Game-Tree Search

> There are more possible positions in Go than there are atoms in the universe. That makes Go a googol [$10^{100}$] times more complex than Chess.
>
> Google DeepMind

Optimal value $v^*(s)$ determines the outcome of the game from every board position $s$ under perfect play by all players. This value can be computed by recursively traversing the search tree containing approximately $b^d$ possible sequences of moves, where $b$ is the breadth of game (number of legal moves per position) and $d$ is its depth (game length).

In the case of Chess, these are $b \approx 35$ and $d \approx 80$, whereas Go has $b \approx 250$ and $d \approx 150$. This amounts to a terrifying overnumerousness: there are more Go

positions than atoms in the observable Universe. Therefore, exhaustive search is deemed intractable.

How to handle the size of the game tree?

- ♠ For the breadth, we train *a neural network to select* moves.
- ♠ For the depth, we train *a neural network to evaluate* the current position.
- ♠ For the tree traverse, we use *Monte Carlo tree search* method.

**Monte Carlo tree search (MCTS)** is a Monte Carlo heuristic version of the classical tree search. However, instead of traversing the entire game tree, the MCTS selects the most promising moves, expanding the search tree based on random sampling.

In each iteration, the game is played-out to the very end, by choosing moves at random. The final outcome of each playout is then used to accordingly weigh the nodes in the game tree. Thus, better nodes are more likely to be chosen in future playouts.

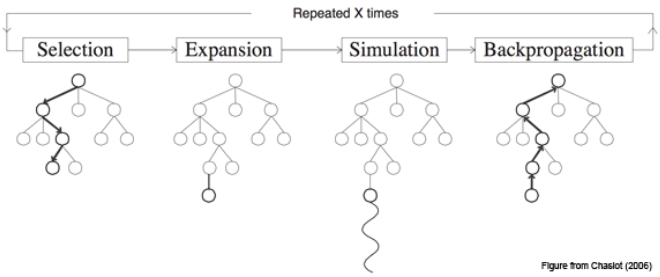



Figure D.1: The scheme of MCTS

## D.2 Neural networks

> Your brain does not manufacture thoughts. Your thoughts shape neural networks.
>
> Deepak Chopra

Inspired by biological neural networks, an artificial neural network (ANN) is a network of interconnected nodes that make up a model (see Figure D.2).

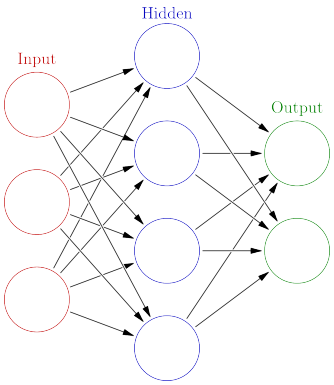



Figure D.2: A shallow neural network with 3 layers

We can think of ANNs as statistical learning models that are used to approximate functions with a large number of inputs. Neural networks are typically used when the amount of inputs is far too large for standard machine learning approaches.

A **deep neural network (DNN)** is a term for any neural network with many hidden layers (but most commonly we use it for convolutional neural networks). It can model complex non-linear relationships in domains such as speech, images, videos, board positions in Go, etc.

A **convolutional neural network (CNN)** is a neural network suitable for high-dimensional inputs (e.g., many pixels in an image). CNNs are frequently used in computer vision, e.g., for identifying objects in an image, face detection in photos, etc. They are resistant to expectable transformations of input, such as changes in illumination or translations of objects in a picture.

## D.3 Training Pipeline of Neural Networks

> Pattern recognition and association make up the core of our thought. These activities involve millions of operations carried out in parallel, outside the field of our consciousness. If AI appeared to hit a brick wall after a few quick victories, it did so owing to its inability to emulate these processes.
>
> Daniel Crevier

AlphaGo employs two kinds of deep CNNs—a *policy network* (for move selection) and a *value network* (for board evaluation):

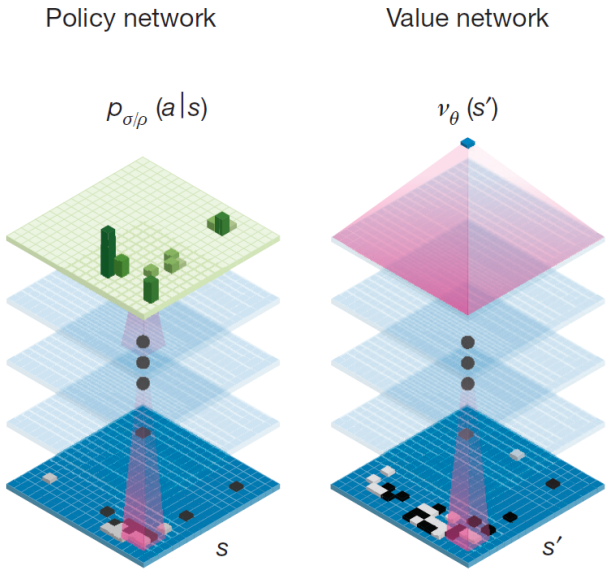



Figure D.3: Comparison between policy and value network (Silver et al. 2016)

The following figure outlines the entire training process of AlphaGo's networks.

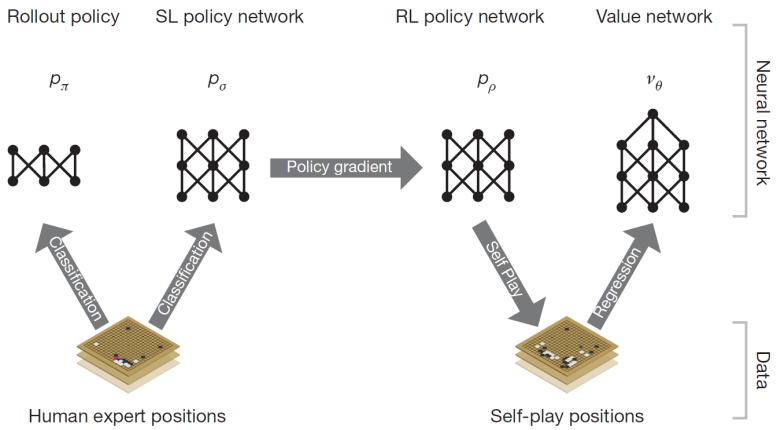



Figure D.4: Training the neural networks of AlphaGo: the pipeline and the architecture (Silver et al. 2016)

**Rollout policy** $p_\pi$ is a CNN rapidly sampling actions during a *rollout* (a fast-forward simulation from a position to the end of a game). It predicts expert human moves less accurately but much faster than $p_\sigma$ (below). The output is a probability distribution over all moves.

**Policy network** is a move-selecting CNN. It addresses the problem of the game-tree breadth. There are two flavors of these networks:

- ♠ **SL policy network** $p_\sigma$ is trained by supervised learning to predict expert human moves.
- ♠ **RL policy network** $p_\rho$ is trained by reinforcement learning to win in the games of self-play.

**Value network** $v_\theta$ is a CNN evaluating board positions, so as to address the problem of the game-tree depth. It is trained by regression to predict the outcome in positions of the self-played games.

## D.4 Main Algorithm of AlphaGo

> Computers are good at following instructions, but not at reading your mind.
>
> Donald Knuth

Finally, the neural networks are combined with MCTS into the main algorithm.

During the play, AlphaGo simulates up to 100 possible continuations per each move by selecting the most promising actions and following them. This way, it descends the game-tree down to a depth given by a parameter. At that point, leaf nodes are evaluated in two ways:

(1) using the dedicated value network $v_\theta$,

(2) simulating the self-play until the terminal positions, using the fast rollout policy $p_\pi$.

The two are mixed into the final value of the leaf node.

Once all simulations for a single round are finished, all new values are back-propagated to the root, thus updating necessary variables on the way up.

For more details, consult (Silver et al. 2016) or my mentioned presentations.

**Note**

Main algorithm demonstrates the connection to *our theme of end-games*: the simulations may be viewed as "solving endgames". In particular, the $p_\pi$ rollouts somehow remind of exhaustive search for the game value during the late stage of the play. In this sense, the approaches of AlphaGo and endgame computation bear a striking similarity.

On the other hand, AlphaGo performs the same simulation algorithm during the whole game: from an empty board until the final move. Therefore, it would be imprecise to talk about "endgame".

## D.5 Playing Strength

> I know AlphaGo is a computer, but if no one told me, maybe I would think the player was a little strange, but a very strong player, a real person.
>
> Fan Hui

In order to assess the playing strength of AlphaGo, DeepMind has organized an internal tournament against other Go programs.[1] Figure D.5 displays the outcome of the tournament:

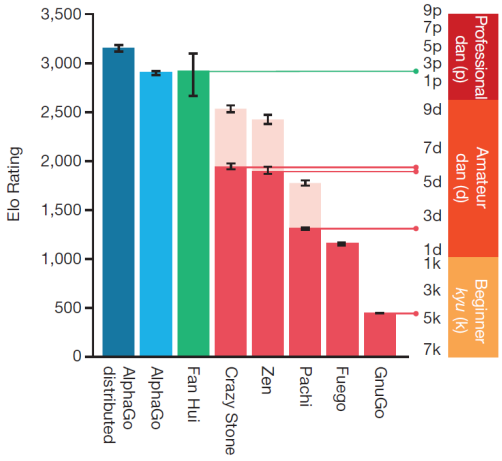



Figure D.5: Tournament with other programs and Fan Hui (Silver et al. 2016)

The 95%-confidence intervals were used. Programs were allowed approximately 5 seconds of computational time per move, and some of them (with faded

[1] *CrazyStone* and *Zen* are the strongest commercial programs, whereas *Pachi* and *Fuego* are strongest among the open-source ones.

tops of bars) also later played with four *handicap stones* (i.e., free extra moves at the beginning of each game) against all others.

The Elo scale was used to measure the playing strength: a 230-point gap equals to a 79%-probability of winning, as well corresponding to about one *amateur dan* rank difference on KGS Go Server[2]. The ranks achieved by programs on KGS are shown on the right side.

Finally, AlphaGo played two spectacular duels against professional human players:

**Fan Hui**

- ♠ professional $2^{\text{nd}}$ dan
- ♠ European Go Champion in 2013, 2014 and 2015
- ♠ European Professional Go Champion in 2016

→ **AlphaGo won 5:0** in a formal match in October 2015, becoming *the first program to ever beat a professional Go player in an even game.*[3]

**Lee Sedol "The Strong Stone"**

- ♠ professional $9^{\text{th}}$ dan
- ♠ the $2^{\text{nd}}$ in international titles
- ♠ the $5^{\text{th}}$ youngest (12 years 4 months) to become a professional Go player in South Korean history
- ♠ the top Go player in the world over the past decade
- ♠ Legendary Lee Sedol would win 97 out of 100 games against Fan Hui.

→ **AlphaGo won 4:1** in March 2016. Each game was won by resignation.

The winner of the match was awarded the prize of $1,000,000. Google DeepMind donated this money to charities, including UNICEF, and various Go organisations.

Lee Sedol received $170,000 (i.e., $150,000 for participation in all games and extra $20,000 for every victory).

> I have never been so much celebrated and praised for winning one single game.
>
> Lee Sedol

**Note**

Following from this, computers seem to have reached the "super-human" level of expertise and they now appear to be superior to humans in the game of Go.

[2] https://www.gokgs.com/

[3] https://deepmind.com/alpha-go.html

# Part II

# Imperfectness of Imperfect-Information Endgames

# E. Setting the Scene for Imperfect Information

> Intelligence is a game of imperfect information. We can guess our opponent's moves, but we can't be sure until the game is over.
>
> Khalid Muhammad

So far we were dealing with *perfect-information* games: neither chance nor hidden information were involved in Chess or Go. Any player can theoretically (with unlimited computational resources and/or infinite amount of time) look ahead, investigate all possible continuations, pick the best one and play perfectly. Of course in case of Chess and Go, an exhaustive search may easily take longer than the age of the Universe. Nevertheless, there are no mathematical limitations preventing such analysis.

*Imperfect-information* games, on the other hand, need to also include private information, e.g., opponent's cards in Poker. We cannot (and should not) play differently in equivalent states, whose indistinguishability is caused by private information of our opponent. Such missing knowledge is captured in the concept of an *information set*.

## E.1 Extensive Form for Imperfect-Information

> Education is only a ladder to gather fruit from the tree of knowledge, not the fruit itself.
>
> Albert Einstein

Recall extensive forms for perfect-information games from Section A.1. An *extensive form* (for an imperfect-information game) is the usual extensive form with additional:

- ♠ **c**hance player $c$ (e. g. a dice, the card dealer, the nature etc.). The set of players is thus $P \cup \{c\}$ and the probability corresponding to a strategy profile $\sigma$ includes the chance:

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

- ♠ function $f_c$ determining the probability distribution over actions $A(h)$ for every chance node $h$ (i.e., $p(h) = c$).

- ♠ partition $\mathcal{I}_i$ of nodes $\{h \in H : p(h) = i\}$, which is called the *information partition* of player $i$. Its element $I \in \mathcal{I}_i$ is an *information set* of player $i$. By $I(h) \in \mathcal{I}_i$ (with $p(h) = i$) we denote the information set containing $h$, and by $Z(I) = \{z \in Z | z \sqsupset h, h \in I\}$ the set of terminal nodes reachable from $I$.

An information set represents grouping of histories that are indistinguishable from $i$'s point of view. In the game of poker, for example, this might be because of opponents' hidden cards.

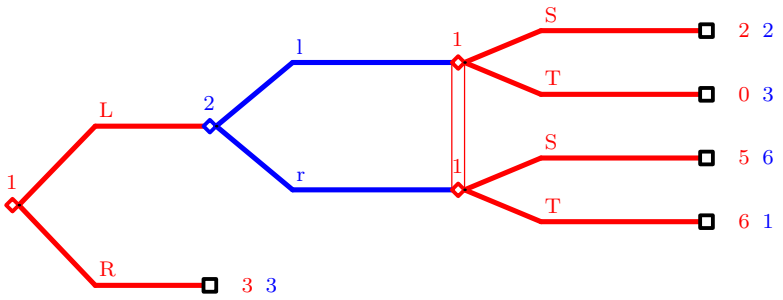



Figure E.1: An imperfect-information game tree with information sets (Nisan et al. 2007, p. 67)

There are further notions related to EFGs:

- A (behavior) *strategy* $\sigma_i$ of player $i$ gives a probability distribution over $A(I)$ at every $I \in \mathcal{I}_i$, and $\pi^\sigma(I, a)$ is the probability of action $a$ at the information set $I$. Again, $\Sigma_i$ denotes the set of all possible strategies for player $i$.

- $\sigma|_{I \to a}$ denotes the strategy identical to $\sigma$ with the only one exception: the action $a$ is always played at the information set $I$.

- The *counterfactual value* $v_i^\sigma(I)$ is a "what-if" value in a hypothetical scenario, where one had taken the actions to arrive at $I$.

  Formally, it is the expected utility provided that the information set $I$ is reached and all players play according to strategy $\sigma$ with exception of player $i$, who plays to reach $I$:

  $$v_i^\sigma(I) = \sum_{h \in I,\, h' \in Z} \frac{\pi_{-i}^\sigma(h)\pi^\sigma(h, h')u_i(h')}{\pi_{-i}^\sigma(I)}$$

  Likewise, for $a \in A(I)$

  $$v_i^\sigma(I, a) = \sum_{h \in I,\, h' \in Z} \frac{\pi_{-i}^\sigma(h)\pi^\sigma(h \cdot a, h')u_i(h')}{\pi_{-i}^\sigma(I)}$$

- A *counterfactual best response* $CBR_i(\sigma_{-i})$ (briefly $CBR_i(\sigma)$ or $CBR(\sigma_{-i})$) of player $i$ is a strategy maximizing the counterfactual value at each information set $I \in \mathcal{I}_i$:

  $$\pi^\sigma(I, a) \geq 0 \iff v_i^\sigma(I, a) = \max_{a' \in A(I)} v_i^\sigma(I, a')$$

  Note that $CBR_i(\sigma)$ is always a best response $BR_i(\sigma)$, but the reverse implication does not need to hold: a best response $\sigma$ can select an arbitrary action in an unreachable information set $I$ (the one where $\pi^\sigma(I) = 0$). Such best responses are in general not counterfactual best responses.

- For the sake of notation's simplicity, we will define a *counterfactual best-response value (CBV)* as the counterfactual value for the strategy, where player $i$ plays according $CBR_i(\sigma_{-i})$ rather than the original $\sigma$. Formally, it is
$$CBV_i^\sigma(I) = v_i^{(\sigma_{-i}, CBR_i(\sigma_{-i}))}(I)$$
- An EFG with (non-singleton) information sets can have a *perfect recall*: any two states from the same information set $I \in \mathcal{I}_i$ share the same history of past actions and same passed information sets of $i$.

  Games without this property have a *imperfect recall*: at some stage of the game a player forgets what happened so far, either some actions taken or some information sets reached.
- In perfect-recall games, $z[I]$ denotes $z$'s unique ancestor $h$ that lies in $I$ (i.e., $h \in I$ and $h \sqsubseteq z$).

## E.2 Sequence Form

> Error is ever the sequence of haste.
>
> Duke of Wellington

**Note**

This section is based on Nisan et al. 2007, Section 3.10.

One possible way (Nisan et al. 2007, pp. 73–74) to solve EFGs is by solving a corresponding linear program (LP) called the *strategic form*. However, the number of pure strategies (over which players are mixing) is generally exponential, as we need to combine all action choices across all information sets (see Figure E.2). The size of the *strategic-form* LP increases rapidly, deeming the problem intractable.

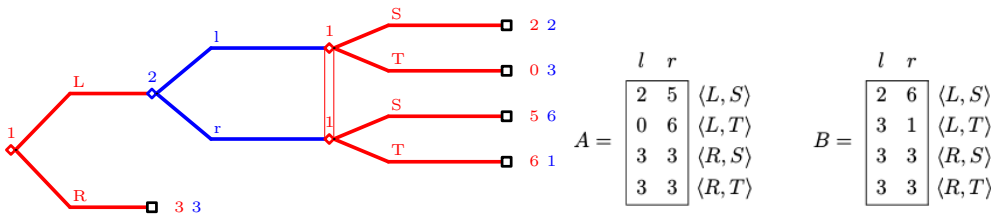



Figure E.2: An EFG and its corresponding strategic-form pay-off matrices. (Nisan et al. 2007, p. 67)

A possible solution is to use the *sequence form* (Nisan et al. 2007, pp. 70–73). This form of LP has variables just for *sequences of actions*. Only sequences from the same root-to-leaf paths are then combined:

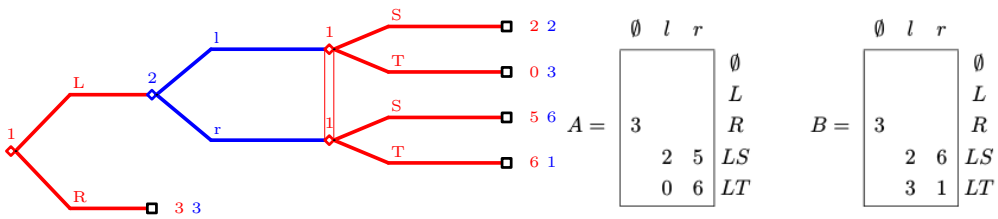



Figure E.3: A game with sequences $S_1 = \{\varnothing, L, R, LS, LT\}$ and $S_2 = \{\varnothing, l, r\}$ and its corresponding sequence-form pay-off matrices. (Nisan et al. 2007, p. 67)

♠ *Sequence* $s_i(I)$ describes player $i$'s history of actions, i.e., edges passed along the path from the root to information set $I \in I_i$.

Set $S_i$ denotes the set of all $i$'s valid sequences: every non-empty sequence $s \in S_i$ is uniquely determined by its last action $a \in A_I$ at an information set $I \in \mathcal{I}_i$. We can characterize

$$S_i = \{\varnothing\} \cup \{s_i(I) \cdot a \mid I \in \mathcal{I}_i,\, a \in A(I)\}$$

which leads to the linear size $|S_i| = 1 + \sum_{I \in \mathcal{I}_i} |A(I)|$.

♠ The *realization probability* (of a sequence $s \in S_i$ under a strategy $\sigma_i \in \Sigma_i$) is $\pi^{\sigma_i}[s] = \prod_{a \in s} \pi^{\sigma_i}(a)$.

♠ Recall that a mixed strategy $\mu_1$ is a probability distribution over (all) pure strategies of player 1. A *realization plan*[1] $x\colon S_1 \to \mathbb{R}$ expresses the probability of following sequence $s$ under a mixed strategy $\mu_1$:

$$x(s) = \sum_{\text{pure strategy } \sigma} \mu_1(\sigma)\, \pi^{\sigma}[s]$$

If $\mu_1$ is a behavior strategy and sequence $s$ ends in a state $h$, we directly get $x(s) = \pi_1^{\mu_1}(h)$.

## E.3 Solving Games with Linear Programming

> If you optimize everything, you will always be unhappy.
>
> Donald Knuth

**Note**

This section is based on Nisan et al. 2007, Section 3.11.

Realization plans play the role of variables in sequence-form LP formulations. We think of realization plans as vectors $x := (x_s)_{s \in S_1} \in \mathbb{R}^{|S_1|}$ and $y := (y_s)_{s \in S_2} \in \mathbb{R}^{|S_2|}$. Not all vectors $x \geq \mathbf{0}$, $y \geq \mathbf{0}$ can be realization plans, though. We need to add two more types of LP constraints.

The first kind of necessary conditions states that the probability of a sequence can be decomposed into probabilities of its immediate continuations:

$$\begin{aligned} x_{s_1(I)} &= \sum_{a \in A(I)} x_{s_1(I \cdot a)}, \quad I \in \mathcal{I}_1, \\ y_{s_2(I')} &= \sum_{a' \in A(I')} y_{s_2(I' \cdot a')}, \quad I' \in \mathcal{I}_2. \end{aligned} \tag{E.1}$$

The second kind of conditions ensures that the probabilities in realization plans sum to 1:

$$x_{\varnothing} = 1, \quad y_{\varnothing} = 1 \tag{E.2}$$

[1] For player 2, a realization plan $y\colon S_2 \to \mathbb{R}$ would be defined similarly.

Interestingly, the conditions (E.1), (E.2) are also sufficient, giving a full characterization for realization plans (Nisan et al. 2007, Proposition 3.10, p. 71).

Now we re-formulate (E.1) and (E.2) using *sequence constraint matrices*:

$$Ex = e,\ x \geq \mathbf{0} \quad \text{and} \quad Fy = f,\ y \geq \mathbf{0}. \tag{E.3}$$

The first player gets a sequence constraint matrix $E \in \{-1, 0, 1\}^{(1+|\mathcal{I}_1|)\times|S_1|}$, where columns are indexed by sequences from $s \in S_1$ and rows by equations of (E.1) and (E.2). Specifically, the first row of $Ex = e$ has form

$$x_\varnothing = 1,$$

and the row corresponding to an information set $I \in \mathcal{I}_1$ looks like

$$x_{s_1(I)} - \sum_{a \in A(I)} x_{s_1(I \cdot a)} = 0.$$

The *sequence constraint vector* $e$ on the right-hand side is thus $(1, 0, \ldots, 0)^\top$. Matrix $E$ is sparse as it has only $\mathcal{O}(|\mathcal{I}_1| + \sum_{I \in \mathcal{I}_i} |A(I)|)$ non-zero elements. For $F \in \{-1, 0, 1\}^{(1+|\mathcal{I}_2|)\times|S_2|}$ and $f = (1, 0, \ldots, 0)^\top$, constraints $Fy = f$ are analogous.

The *sequence-form payoff matrix* $A \in \mathbb{R}^{|S_1|\times|S_2|}$ of player 1 is described for every sequence $s \in S_1$ and $s' \in S_2$ by

$$A_{s,s'} = \sum_{\substack{z \in Z \\ s_1(z)=s,\ s_2(z)=s'}} \pi_c(z)\ u_1(z),$$

where $\pi_c(z)$ is the contribution of the chance to the probability, along the path to leaf $z$. The sequence-form payoff matrix of player 2 is $-A$ from the zero-sum property. Because the probability of reaching any leaf $z$ can be decomposed as $\pi^\sigma(z) = \pi^{\sigma_1}[s_1(z)] \cdot \pi^{\sigma_2}[s_2(z)] \cdot \pi_c(z)$, we can write player 1's utility as

$$u_1(\sigma) = \sum_{z \in Z} \pi^{\sigma_1}[s_1(z)] \cdot \pi^{\sigma_2}[s_2(z)] \cdot \pi_c(z) \cdot u_1(z) = \sum_{z \in Z} x_{s_1(z)} y_{s_2(z)} \pi_c(z)\, u_1(z) = x^\top A y$$

So a best-response realization plan $x$ against arbitrary realization plan $y$ maximizes the utility $u_1(\sigma) = x^\top (Ay)$. By (E.3), we can find such $x$ as a solution to the following LP (parametrized by $y$):

$$\begin{aligned} \max\ & (Ay)^\top x \\ & Ex = e \\ & x \geq \mathbf{0}. \end{aligned} \tag{E.4}$$

This LP has a finite optimal value, because it is

(a) *feasible*: Any valid realization plan $x$ will do.

(b) *bounded*: Since $x$ expresses probabilities, $\mathbf{0} \leq x \leq \mathbf{1} = (1, \ldots, 1)^\top$, giving an upper bound $(Ay)^\top x \leq |Ay|^\top \mathbf{1}$.

The corresponding dual LP with a vector of dual variables $u \in \mathbb{R}^{1+|\mathcal{I}_1|}$:

$$\begin{aligned} \min \ & e^\top u \\ & E^\top u \geq Ay. \end{aligned} \tag{E.5}$$

is therefore also feasible, bounded and has the equal optimal value by the strong duality theorem. In a zero-sum game, when player 1 wants to maximize his utility $x^\top Ay = u^\top e$, his opponent wants to minimize it by his choice of a realization plan $y$. Realization plans are characterized by constraints of (E.3). All of this leads to this LP formulation

$$\begin{aligned} \min_{u,y} \ & e^\top u \\ E^\top u - Ay &\geq \mathbf{0} \\ Fy &= f \\ y &\geq \mathbf{0} \end{aligned} \tag{E.6}$$

with such a dual LP

$$\begin{aligned} \max_{v,x} \ & f^\top v \\ Ex &= e \\ F^\top v - A^\top x &\leq \mathbf{0} \\ x &\geq \mathbf{0}, \end{aligned} \tag{E.7}$$

where $v$ is the vector of (negative) CBVs for player 2 (Nisan et al. 2007; Čermák, Bošanský, and Lisý 2014). The (E.7) is the standard *sequence-form LP formulation* used for solving EFGs.

Notably, $E$, $F$ and $A$ are sparse matrices, with the number of variables and non-zero values linear in the size of the game tree. So zero-sum EFGs can be solved by (dual) linear programming for the *linear-sized* LPs (E.7).

## E.4 Solving Games with Learning Algorithms

> Perfecting oneself is as much unlearning as it is learning.
>
> Edsger Dijkstra

In contrast to solving games by linear programming, we can approximate Nash equilibrium of EFGs using various *learning algorithms*:

**(Vanilla) counterfactual regret minimization (CFR)** is a *self-playing* algorithm based on *regret matching*. In the learning phase, it improves itself by summing the total amount of *counterfactual regret* (a regret related to counterfactual values), for every action at each decision point. The average strategy over all iterations converges towards a NE for the game, and it is thus used as the final strategy during the play. (Zinkevich et al. 2007)

**Monte Carlo counterfactual regret minimization (MCCFR)** is a *Monte Carlo* version of CFR. Rather than traversing the full game tree, it probabilistically samples actions to speed up computation per learning iteration.

The partial tree exploration is nevertheless compensated by weighting regrets with appropriate probabilities. (Johanson et al. 2012)

**CFR$^+$** is a new CFR-type algorithm based on *regret-matching$^+$*. A *cumulative counterfactual regret$^+$* ($\max(\cdot, 0)$) is used instead. It typically outperforms the previously known algorithms by an order of magnitude or more in terms of computation time, while also potentially requiring less memory. Another advantage is that many of the cumulative regret values are zero, whereas in CFR, negative regret continues to accumulate indefinitely. This reduces the entropy of the data needed during the computation. (Tammelin et al. 2015)

These iterative methods run faster than methods with sequence-form LPs, and use significantly less memory. Moreover, they guarantee a convergence to Nash equilibrium in limit (Zinkevich et al. 2007, Theorem 4). Therefore, they currently prevail as dominant solutions, and are standardly used for solving large imperfect-information games. For a detailed overview, study also (Nisan et al. 2007, Chapter 4) or (Bošanský 2013).

## E.5 Subgames Revisited

> When you think you can't, revisit a previous triumph.
>
> Jack Canfield

So far, the information sets have grouped only those states where the player was the acting player. For subgames, it is also necessary to include the states that are indistinguishable from the point of view of other players. Therefore, we define the *augmented information sets* (Burch, Johanson, and Bowling 2014):

**Definition 3** (augmented information set)**.** For any player $i \in P$, let $H_i(h)$ be the sequence of player $i$'s information sets reached by player $i$ on the path to $h$, and the actions taken by player $i$.

Then *augmented information set* $I_i(h)$ is defined by the following characterization:

$$I_i(h) = I_i(h') \iff H_i(h) = H_i(h')$$

for any two states $h, h' \in H$.

At this point, we may finally define the notion of a *subgame* (Burch, Johanson, and Bowling 2014):

**Definition 4** (subgame)**.** An (imperfect-information) **subgame** is a forest of subtrees, closed under both the descendant relation and membership within augmented information sets for any player.

Consult Section F.2 for an illustrative example.

# F. Shortcomings of the Imperfect Information for Subgames

> Our shortcomings are the eyes with which we see the ideal.
>
> Friedrich Nietzsche

In this chapter we will notice how the situation for subgames becomes more delicate within the imperfect-information setting. A re-solved subgame strategy can end up being more exploitable: even if we play a best response (BR) in the subgame, we **assume the fixed original strategy** in the trunk of the game tree.

This is, however, not true since the opponent can distinguish the states within our own information set, which—for us—are indistinguishable. He can therefore freely change his behavior in the trunk and manipulate against our best response for his own benefit. For more illustration, study the example in Section F.1 (taken from Burch, Johanson, and Bowling 2014).

## F.1 An Intricate Example

> It's very simple. Scissors cuts paper. Paper covers rock. Rock crushes lizard. Lizard poisons Spock. Spock smashes scissors. Scissors decapitates lizard. Lizard eats paper. Paper disproves Spock. Spock vaporizes rock. And as it always has, rock crushes scissors.
>
> Sheldon Cooper (*The Big Bang Theory*), Season 2, Episode 8

To see the difficulties of combining a re-solved subgame strategy with an original trunk strategy, we will consider *RPS* as our running example:

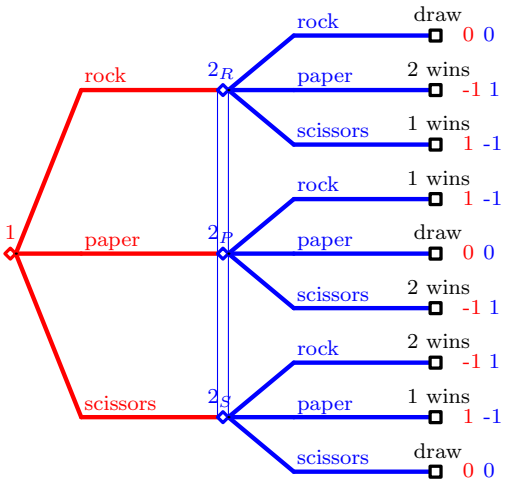



Figure F.1: Extensive form of RPS: Scissors cuts paper. Paper covers rock. Rock crushes scissors. (Burch, Johanson, and Bowling 2014; Ganzfried and Sandholm 2015)

Figure F.1 displays the extensive form of RPS: rather than showing the choices simultaneously, the two players take turns in picking their action. However, their decisions are mutually hidden, and revealed only at the end.

The blue "bubble" encircling the 3 acting nodes of player 2 marks his (only) information set $\mathcal{I}_2 = \{2_R, 2_P, 2_S\}$, where states correspond to the action chosen by the first player. This imperfect information represents the fact that he is unaware of 1's preceding choice.

## F.2 Naïve Re-solving of Rock-Paper-Scissors

> Every mathematical discipline goes through three periods of development: the naïve, the formal, and the critical.
>
> David Hilbert

One apparent subgame (recall Definition 4) of RPS is the endgame when player 2 picks his choice:

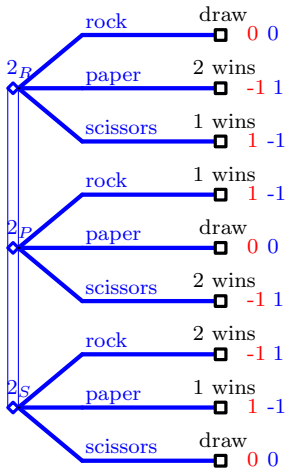



Figure F.2: An (imperfect-information) subgame $S$ of RPS (Burch, Johanson, and Bowling 2014)

At the beginning of subgame $S$, player 2 resides in one of the three states $\mathcal{I}_2 = \{2_R, 2_P, 2_S\}$ depending on opponent's strategy in the (removed) trunk. Specifically, the probability of being in state $2_R$ equals to 1's probability to choose rock, and likewise for states $2_P$ and $2_S$. The three augmented information sets of player 1 (Definition 3) are $\mathcal{I}_1^R = \{2_R\}$, $\mathcal{I}_1^P = \{2_P\}$ and $\mathcal{I}_1^S = \{2_S\}$.

It is not enough to naïvely re-solve subgame $S$ with the assumption of a fixed strategy in the trunk:

**Claim 3** (Ganzfried and Sandholm 2015, Proposition 1; Burch, Johanson, and Bowling 2014)**.** *If subgame $S$ is re-solved with a fixed trunk strategy, there is an equilibristic solution (i.e., a best response against the fixed trunk strategy) that player* 1 *can exploit, once he is allowed to adjust his strategy in the trunk.*

*Proof.* Suppose the initial strategy $\sigma = (\sigma_1, \sigma_2)$ is uniformly random[1]. Evidently, $\sigma$ is an equilibrium for the full game of RPS.

[1]Every move is selected with the same probability $\frac{1}{3}$.

Now, if we discard the trunk and assume the fixed trunk strategy $\sigma_1$, player 2 is to be in any of the states $\mathcal{I}_2 = \{2_R, 2_P, 2_S\}$ with the equal probability. Because of this, each action results in the same expected utility of 0 (e.g., playing rock gives utility $\frac{1}{3} \cdot 0 + \frac{1}{3} \cdot (-1) + \frac{1}{3} \cdot 1 = 0$). Hence, it is irrelevant for which strategy player 2 decides, as all strategies produce equally good utilities, and thus, all are BRs to 1's strategy[2].

Player 2 can opt for a strategy $\sigma_2^R$ of *always playing rock*, as it is a valid BR. This would become the equilibristic solution from the statement. If player 1 can change his strategy in the trunk, though, he may naturally exploit the equilibristic solution of player 2. Namely, a strategy $\sigma_1^P$ of *always playing paper* certainly defeats $\sigma_2^R$. ■

**Corollary 2.** *Even in a two-player zero-sum game with a unique equilibrium and a single subgame, the trunk strategy and the re-solved (even optimal) subgame strategy can fail to merge into a full-game equilibrium.*

## F.3 A New Hope for Subgames with Imperfect Information

> Help me, Obi-Wan Kenobi; you're my only hope.
>
> Princess Leia (*Star Wars: Episode IV – A New Hope*)

The dismal example of RPS shows the obstacles of naïvely using the same approach as with the perfect-information case. Solving subgames separately thus appear as an impossible mission.

However, in the following chapters we will look into two recent techniques, which deal with imperfect-information subgames in a promising way:

(i) endgame solving (Ganzfried and Sandholm 2015) in Chapter G

(ii) CFR-D and decomposition (Burch, Johanson, and Bowling 2014) in Chapter H

We also re-state both of them using equivalent LP formulations. Treating these techniques as optimization problems will help to reveal their underlying features.

Finally in Chapter I, we use these insights to motivate our own, new technique: *subgame-margin maximization* (Moravčík et al. 2016).

[2]Player 1 has only one possibility, the empty strategy, since he takes no action in subgame $S$.

# G. Endgame Solving

> Poker has the only river in the world you can drown in more than once.
>
> an old Poker joke

**Note**

This chapter summarizes the approach, methods and results of the authors Ganzfried and Sandholm 2015.

## G.1 Motivation and Overview

> Memorizing a playbook is like memorizing a script. When they change the script at the last minute it's like changing a play in a game.
>
> Michael Strahan

Two-player zero-sum imperfect-information games can be solved via linear programming (Koller, Megiddo, and Von Stengel 1994), by modelling sequences of moves as variables of a sequence-form LP (recall Section E.3). This approach scales well to games with up to $10^8$ game states. Unfortunately, many attractive Poker games are far larger (Johanson 2013):

- ♠ two-player Limit Texas Hold'em $\approx 10^{17}$ states
- ♠ two-player No-Limit Texas Hold'em[1] $\approx 10^{165}$ states.

It is possible to find approximate equilibrium with iterative algorithms, as well. These methods are guaranteed to converge in the limit, and scale to at least $10^{12}$ states (Hoda et al. 2010; Zinkevich et al. 2007). Nevertheless, that is still not enough for the mentioned big poker variants.

Today, a prevailing approach to enormous imperfect-information games (such as LHE or NLHE) is to reduce their sizes by means of *abstractions*:

**Information abstraction** groups together different signals (e.g., similar poker hands).

**Action abstraction** discretizes an immense action space, making it thus smaller and more manageable.

The method afterwards finds an *approximate equilibrium* in the abstracted game.

An appealing idea to diminish harmful effects of the abstraction and the approximate-equilibrium search, is to solve endgames dynamically: an agent only needs to deal with those portions of the game that are actually reached during a play. As a consequence, the endgame can be re-solved with a finer abstraction and more precise pot and stack sizes.

[1] the most popular online variant of Poker

## G.2 Gadget Game

> What is so brilliant about the gadgets is their simplicity.
>
> Desmond Llewelyn

The idea is to pre-compute coarser strategy for the initial phase of game. After reaching certain end-phase situation during a real play, one can obtain better strategy for the current play using finer abstraction in the endgame that has arisen and players' current probability distributions.

We start by constructing a fine-grained subgame abstraction. The original strategies for the subgame are discarded and only the strategies prior to the subgame (i.e., the trunk) are needed. The strategies in the trunk are used to compute the joint distribution (i.e., belief) over the states at the beginning of the subgame.

Finally, we add a chance node $c$ just before the fine-grained subgame $S$. The node leads to the states at the root of the subgame. The chance node plays according to the computed belief. Adding the chance node "roots" the subgame, thus making it a proper tree of a well-defined game (Figure G.1).

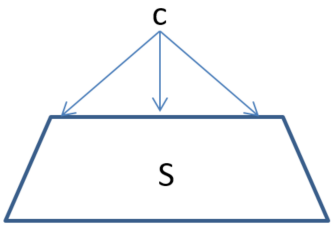



Figure G.1: A gadget game for endgame solving

## G.3 Equivalent Linear Program

The equivalent LP formulation for the abstracted endgame

$$\begin{aligned} \max_{v,x} \quad & f^\top v \\ Ex &= e \\ F^\top v - A^\top x &\leq \mathbf{0} \\ x &\geq \mathbf{0}, \end{aligned}$$

is the sequence-form LP for the gadget game. Same as in the LP from (E.7):

- ♠ $A$ is the sequence-form payoff matrix
- ♠ $x$ is a vector of 1's sequence probabilities
- ♠ $v$ is a vector of 2's (negative) counterfactual best-response values
- ♠ $E$ and $F$ are the sequence constraint matrices
- ♠ $e$ is the sequence constraint vector.

## G.4 Discussion

The endgame solving improves agents' play only *empirically* (Ganzfried and Sandholm 2015, Table 1). Theoretically, however, there is no guarantee of optimality: even if the trunk strategy (and thus the starting distribution) is optimal, the combined strategy can become drastically more exploitable. This undesired effect is noticeable in the extensive form of RPS (Claim 3, Chapter F).

A worse exploitability arises when the opponent can "see a better option" and aim for it. This notion of a better alternative can be summarized by a *counterfactual value*, and Chapter H shows how to use these values to guarantee theoretical bounds.

# H. CFR-D and Decomposition

> Everything that comes together falls apart. Everything. The chair I'm sitting on. It was built, and so it will fall apart. I'm gonna fall apart, probably before this chair. And you're gonna fall apart. The cells and organs and systems that make you you– they came together, grew together, and so must fall apart. The Buddha knew one thing science didn't prove for millennia after his death: Entropy increases. Things fall apart.
>
> John Green, *Looking for Alaska*

**Note**

This chapter summarizes the approach, methods and results of the authors Burch, Johanson, and Bowling 2014.

## H.1 Motivation and Overview

The work of (Burch, Johanson, and Bowling 2014) describes a pioneering technique how to *decompose* subgames of imperfect-information games and solve them independently, while preserving the optimality of the full-game solution. Many benefits of subgame re-solving are inherited from perfect-information games:

- ♠ Run-time information (e.g., endgame actually reached during a real play) can be exploited in a smarter way.
- ♠ Memory and disk limitations (either at run-time or while solving a game) can be overcome by a time/space trade-off.
- ♠ A Nash equilibrium for a game larger than available storage may be computed.
- ♠ If we only need to work with one subgame at a time, then significantly less storage is required.
- ♠ It is not obligatory to store the complete strategy, which might be too large to store. Instead, subgame strategies may be re-computed on demand, when needed.

As for re-solving subgames in the imperfect-information setting, all previous algorithms were limited to rather small domains, with the complete strategy that can fit in available space. As a consequence, several appealing games still resist to be solved, despite the intensive effort put in their research.

**Note**

This used to be the case for the two-player LHE, until the recent, highly-celebrated breakthrough by the same research group (Bowling et al. 2015).

## H.2 Gadget Game

> Dreams about the future are always filled with gadgets.
>
> Neil deGrasse Tyson

**Note**

We will distinguish the states, strategies, utilities, etc., and their translations to the gadget game by adding a tilde to corresponding notations.

From now on for the rest of the thesis, we will be refining the strategy for player 1 in a two-player zero-sum game with the perfect recall.

Again, we start by creating a fine-grained abstraction for the subgame. The original strategy for the subgame (from the coarse abstraction) is then translated into the fine-grained abstraction as $\sigma_1^S$. The translated strategy is now used to compute $CBV_2^{\sigma_1^S}(I)$ for every information set $I$ at the root of the subgame. These values will be useful for the gadget construction to guarantee the safety of the resulting strategy. See Figure H.1 for a sketch of the construction.

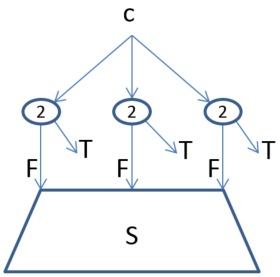



Figure H.1: A gadget game for re-solving subgames. The opponent chooses in every state prior to the endgame either to (F)ollow the action into the endgame, or to (T)erminate. His utility after the (T)erminal action is set to his CBR in that state.

To construct the gadget, we add one chance node at the root of the game, followed by additional nodes for player 2: one for every state at the root of the subgame. At each of these nodes, 2 may either accept the corresponding CBV calculated earlier, or play the subgame (to get to the corresponding state at the root of the subgame).

The chance player distributes the player 2 into these states using the (normalized) $\pi_{-2}^{\sigma}$ (how likely is the state given that 2 plays to reach it). Since the game is zero-sum, this forces player 1 to play the subgame well enough, so that the opponent's value is no greater than the original CBV. Acting like that, 1's overall game utility will not deteriorate:

**Theorem 4** (Burch, Johanson, and Bowling 2014, Theorem 1)**.** *Given a strategy $\sigma_1$, a subgame $S$, and a re-solved subgame strategy $\sigma_1^S$, let $\sigma_1' = \sigma_{1,[S \leftarrow \sigma_1^S]}$ be the combination of $\sigma_1$ and $\sigma_1^S$. If $CBV_2^{\sigma'}(I) \leq CBV_2^{\sigma}(I)$ for all subgame-root information sets $I \in \mathcal{I}_2^{R(S)}$, then $u_2(\sigma_1', CBR(\sigma_1')) \leq u_2(\sigma_1, CBR(\sigma_1))$.*

*Proof.* Consult the appendix of (Burch, Johanson, and Bowling 2014). ■

## H.3 Equivalent Linear Program

This time, the presented LP is not a straightforward sequential-form representation of the gadget construction. Although such a representation would be possible, it would not help to provide a desirable insight. Instead, we formulate an LP that solves the same game (for player 1) while demonstrating the underlying properties of the re-solving approach:

$$\begin{aligned} \max_{v,x} \; & 0 \\ v_I - m &\geq CBV_2^{\sigma_1}(I), \quad I \in \mathcal{I}_2^{R(S)} \\ Ex &= e \\ F^\top v - {A_2}^\top x &\leq \mathbf{0} \\ x &\geq \mathbf{0}, \end{aligned} \tag{H.1}$$

where $\mathcal{I}_2^{R(S)}$ denotes the root information sets and $CBV_2^{\sigma}(I)$ is 2's original counterfactual best-response value in the information set $I$.

The sequence payoff matrices $A_2$ and $A$ (from (G.3)) are slightly different, to reflect different strategies of the chance player in the gadget for endgame solving (Figure G.1) and the gadget for re-solving subgames (Figure H.1).

The formulation uses the following fact: any strategy, where the opponent's CBV is not greater than the original one, is a solution to the game. This follows from the construction of the gadget for re-solving games. Also compare with the subgame of RPS example: a strategy "always play rock" increases opponent's CBV, thus giving him a chance to exploit us there, by playing paper.

It is worth remarking three important points:

(1) (H.1) is not optimizing any value (0 is a constant). Instead, the LP looks for a feasible solution. Though theoretically equivalent, in this case it is semantically different for the strategy.

(2) The original, unrefined strategy is a feasible solution to (H.1).

(3) Points (1) and (2) might suggest the strategy may not improve. Empirical evaluations show otherwise however: using a CFR algorithm to solve the gadget game, the performance of the refined strategy improves over the original (Burch, Johanson, and Bowling 2014). Our experiments further confirm this (Section I.6).

## H.4 Discussion

The innovative aspect of (Burch, Johanson, and Bowling 2014) consists of two main contributions:

(i) A technique to re-solve imperfect-information subgames, which is guaranteed not to increase sub-optimality of the resulting full-game strategy. Owing to summary information about a subgame strategy (namely the counterfactual best-response values), the newly generated strategy is no more exploitable than the original one.

(i) A new off-line solving algorithm called *counterfactual regret decomposition (CFR-D)*, capable of computing an error-bounded NE approximation. CFR-D achieves this by decomposing and independently analyzing subgames.

By sacrificing computation time, the decomposition allows for *sub-linear space costs*. For instance, two-player LHE with CFR-D can be solved in less than 16 GB, rather than more than 200 TB of disk space!

# I. Subgame-Margin Maximization

> I have discovered a truly marvellous proof of this, which this margin is too narrow to contain.
>
> Pierre de Fermat on *Fermat's Last Theorem*

**Note**

This chapter summarizes our own paper (Moravčík et al. 2016).

## I.1 Motivation and Overview

The outline of this chapter is:

(1) listing the steps used by our technique,
(2) using the problem of refining imperfect-information subgames to motivate a value to be maximized,
(3) formalizing this value as the *subgame margin*,
(4) discuss and formalize its properties,
(5) formulate an LP optimizing the SM,
(6) describe a corresponding EFG construction: *a max-margin gadget*.

Our technique follows the steps of the subgame-refinement framework:

(i) create an abstraction for the game;
(ii) compute an equilibrium approximation within the abstraction,
(iii) play according to this strategy,
(iv) when the play reaches final stage of the game, create a fine-grained abstraction for the endgame,
(v) refine the strategy in the fine-grained abstraction,
(vi) use the resulting strategy in that subgame (creating a combined strategy).

Since all the steps except for (v) are identical to already described techniques, we describe only step (v) in details.

## I.2 Subgame Margin

> Your margin is my opportunity.
>
> Jeff Bezos

To address the potential increase in exploitability caused by an opponent altering his behavior in the trunk, we ensure that there is no distribution of starting states that would allow him to increase his CBV when confronted by subgame refinement. The simplest way to ensure this is to decrease his CBV in all possible starting states. We can put a lower bound on this improvement by measuring the state with the smallest decrease in CBV. Our goal is to maximize this lower bound. We refer to this values as the subgame margin (SM).

**Definition 5** (subgame margin)**.** Let $\sigma_1$, $\sigma_1'$ be a pair of player 1's strategies for subgame $S$. Then a subgame margin

$$SM_1(\sigma_1, \sigma_1', S) = \min_{I_2 \in \mathcal{I}_2^{R(S)}} \left( CBV_2^{\sigma_1}(I_2) - CBV_2^{\sigma_1'}(I_2) \right)$$

measures the "gap in decrease" between the old and the new counterfactual best-response values, across all root information sets $I_2 \in \mathcal{I}_2^{R(S)}$.

Subgame margin has several useful properties. The exploitability is strongly related to the value of the margin: if it is non-negative, the new combined strategy is guaranteed to be no more exploitable than the original one. To prove it, we will first re-state Theorem 4 using the following lemma:

**Lemma 5.** *$CBV_2^{\sigma_1'}(I_2) \leq CBV_2^{\sigma_1}(I_2)$ for all $I_2 \in \mathcal{I}_2^{R(S)}$ iff $SM_1(\sigma_1, \sigma_1', S) \geq 0$.*

*Proof.* $CBV_2^{\sigma_1'}(I_2) \leq CBV_2^{\sigma_1}(I_2) \iff 0 \leq CBV_2^{\sigma_1}(I_2) - CBV_2^{\sigma_1'}(I_2) \iff 0 \leq \min_{I_2 \in \mathcal{I}_2^{R(S)}} \left( CBV_2^{\sigma_1}(I_2) - CBV_2^{\sigma_1'}(I_2) \right) = SM_1(\sigma_1, \sigma_1', S)$ ■

**Corollary 3** (Theorem 4 via subgame margin)**.** *Given a strategy $\sigma_1$, a subgame $S$, and a re-solved subgame strategy $\sigma_1^S$, let $\sigma_1' = \sigma_{1,[S \leftarrow \sigma_1^S]}$ be the combination of $\sigma_1$ and $\sigma_1^S$. If $SM_1(\sigma_1, \sigma_1', S) \geq 0$, then $u_2(\sigma_1', CBR(\sigma_1')) \leq u_2(\sigma_1, CBR(\sigma_1))$.*

Moreover, given that the opponent's best response reaches the subgame with non-zero probability, the exploitability of our combined strategy is even reduced. This improvement is at least proportional to the subgame margin:

**Theorem 6** (improvement proportional to the subgame margin)**.** *With the $S$, $\sigma_1$ and $\sigma_1'$ from the Corollary, also assume there exists a best response $\sigma_2^* = BR(\sigma_1')$ such that $\pi^{<\sigma_1', \sigma_2^*>}(I_2) > 0$ for some $I_2 \in \mathcal{I}_2^{R(S)}$. Then*

$$u_1(\sigma_1', CBR(\sigma_1')) - u_1(\sigma_1, CBR(\sigma_1)) \geq \pi_{-2}^{\sigma_1'}(I_2) \cdot SM_1(\sigma_1, \sigma_1', S).$$

*Proof.* Without loss of generality, we may assume

(a) $\sigma_2^* = CBR_2(\sigma_1')$, since we may arbitrarily change strategy in any information set $I$ that is unreachable under $\sigma_2^*$ (i.e., where $\pi_2^{\sigma_2^*}(I) = 0$),

(b) $\pi_2^{\sigma_2^*}(I_2) = 1$, since we can choose any action from a best-response support with probability 1 (obvious by building a best-response recursively bottom-up).

First, we show that if $p(I) = 2$ and $I$ lies on a path from $\varnothing$ (the root of the whole game) to $I_2$, then

$$CBV_2^{\sigma_1'}(I) \leq CBV_2^{\sigma_1}(I) - \pi_{-2}^{\sigma_1'}(I \to I_2) \cdot SM_1(\sigma_1, \sigma_1', S). \tag{I.1}$$

We prove (I.1) by induction on the length of the $I \to I_2$ path (measured in the count of 2's information sets). For the base case $I = I_2$, the claim holds by $\pi_{-2}^{\sigma_1'}(I_2 \to I_2) = 1$ and the definition of the subgame margin:

$$SM_1(\sigma_1, \sigma_1', S) = \min_{I' \in \mathcal{I}_2^{R(S)}} \left( CBV_2^{\sigma_1}(I') - CBV_2^{\sigma_1'}(I') \right) \leq CBV_2^{\sigma_1}(I_2) - CBV_2^{\sigma_1'}(I_2)$$

For the inductive step, let $I \neq I_2$ be an information set from $\varnothing \to I_2$ path, let $I'$ be 2's next information set after $I$ on this path, implying $\pi_2^{<\sigma_1', \sigma_2^*>}(I \to I') = 1$, and let the action $a \in A(I)$ lead to $I_2$, implying $\pi_2^{\sigma_2^*}(I, a) = 1$. We can thus re-expresses the CBV in $I$ as

$$CBV_2^{\sigma_1'}(I) = v_2^{<\sigma_1', \sigma_2^*>}(I, a) \dots$$

by the definition of the counterfactual value and $\pi_2^{<\sigma_1', \sigma_2^*>}(I \to I') = 1$ we get

$$\dots = \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot v_2^{<\sigma_1', \sigma_2^*>}(I') = \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot CBV_2^{\sigma_1'}(I') \dots$$

by inductive hypothesis we bound

$$\begin{aligned} \dots &\leq \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot \left( CBV_2^{\sigma_1'}(I') - \pi_{-2}^{\sigma_1'}(I' \to I_2) \cdot SM_1(\sigma_1, \sigma_1', S) \right) \\ &= \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot CBV_2^{\sigma_1'}(I') \\ &- \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot \pi_{-2}^{\sigma_1'}(I' \to I_2) \cdot SM_1(\sigma_1, \sigma_1', S) \dots \end{aligned}$$

again by the definition of the counterfactual value we re-write back

$$\dots = v_2^{<\sigma_1, CBR(\sigma_1)>}(I, a) - \pi^{<\sigma_1', \sigma_2^*>}(I \to I') \cdot \pi_{-2}^{\sigma_1'}(I' \to I_2) \cdot SM_1(\sigma_1, \sigma_1', S) \dots$$

and by $\pi_2^{<\sigma_1', \sigma_2^*>}(I \to I') = 1$ we conclude

$$\begin{aligned} \dots &= v_2^{<\sigma_1, CBR(\sigma_1)>}(I, a) - \pi_{-2}^{\sigma_1'}(I \to I')\ \pi_{-2}^{\sigma_1'}(I' \to I_2)\ SM_1(\sigma_1, \sigma_1', S) \\ &= v_2^{<\sigma_1, CBR(\sigma_1)>}(I, a) - \pi_{-2}^{\sigma_1'}(I \to I_2)\ SM_1(\sigma_1, \sigma_1', S) \\ &\leq \left( \max_{a' \in A(I)} v_2^{<\sigma_1, CBR(\sigma_1)>}(I, a') \right) - \pi_{-2}^{\sigma_1'}(I \to I_2)\ SM_1(\sigma_1, \sigma_1', S) \\ &= CBV_2^{\sigma_1}(I) - \pi_{-2}^{\sigma_1'}(I \to I_2)\ SM_1(\sigma_1, \sigma_1', S). \end{aligned}$$

If $p(\varnothing) = 2$, then in a zero-sum game we have $u_1(\sigma_1', CBR(\sigma_1')) = -CBV_2^{\sigma_1'}(\varnothing)$ and $u_1(\sigma_1, CBR(\sigma_1)) = -CBV_2^{\sigma_1}(\varnothing)$. Hence, applying (I.1) to the root

$$\begin{aligned} \pi_{-2}^{\sigma_1'}(I_2) \cdot SM_1(\sigma_1, \sigma_1', S) &\leq CBV_2^{\sigma_1}(\varnothing) - CBV_2^{\sigma_1'}(\varnothing) \\ &= u_1(\sigma_1', CBR(\sigma_1')) - u_1(\sigma_1, CBR(\sigma_1)) \end{aligned}$$

proves the theorem. If $p(\varnothing) \neq 2$, then we can simply add a new state for player 2 at the beginning of the game, where 2 has one single action leading to $\varnothing$. ■

Though this lower bound might seem artificial at first, it has promising properties for the subgame refinement. Since we refine the strategy once we reach the subgame, either we face 2's best response that reaches $S$, or he has made a mistake earlier in the game.

Furthermore, the probability of player 1 reaching a subgame is proportional to $\pi_{-2}^{\sigma'_1}(I_2)$. As this term (and by extension, the bound) grows, the probability of reaching that subgame increases. In conclusion, we are more likely to reach a subgame with a larger bound.

## I.3 Linear Program

To formulate the subgame-margin maximization as an LP, we easily modify (H.1):

$$\begin{aligned} \max_{v,x} \; & m \\ v_I - m &\geq CBV_2^{\sigma_1}(I), \quad I \in \mathcal{I}_2^{R(S)} \\ Ex &= e \\ F^\top v - A_2^\top x &\leq \mathbf{0} \\ x &\geq \mathbf{0}, \end{aligned} \tag{I.2}$$

where $m$ is a scalar corresponding to the subgame margin that we aim to maximize. It serves as "a gap" between all values $v_I$ and the given constants $CBV_2^{\sigma}(I)$, and we wish to make this gap as large as possible.

The similarities between (I.2) and (H.1) make it easier to see our improvement: the LP (H.1) only guarantees a non-negative margin, whereas we maximize it. Although the optimization formulation is almost identical to the re-solving, our gadget construction is different.

## I.4 Equivalent Gadget Game

> A new gadget that lasts only five minutes is worth more than an immortal work that bores everyone.
>
> Francis Picabia

One way to find the refined strategy is to solve the corresponding linear program (LP). However, algorithms that are tailor-made for EFGs often outperform the optimization approach (Bošanský 2013). These algorithms often permit the use of domain-specific tricks to provide further performance gains (Johanson et al. 2012). Thus, formulating our optimization problem (I.2) as an EFG will mean that we can compute larger subgame abstractions using the available computing resources. Essentially, the construction of a gadget game equivalent to the LP (I.2) will allow us to compute larger subgames—more than it would be possible with just the plain LP.

All states in the original subgame are directly copied into the resulting gadget game. We create the gadget game by making two alterations to the original subgame:

(i) we shift player 2's utilities using the $CBV_2$ (to initialize all 2's values to zero)

(ii) we add a rooting node $\tilde{d}$ for 2, followed by chance nodes $c_{I_1}, c_{I_2}, c_{I_3}, \ldots$ at the top of the subgame (to allow the opponent to pick any starting state, relating the game values to the margin)

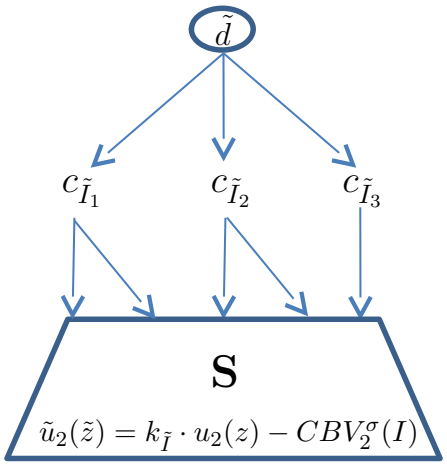



Figure I.1: Our max-margin gadget. When 1's original strategy is used, the terminal nodes' offsets enforce the opponent to have a zero utility in a BR.

The following is a description of the steps (see also Figure I.1 that visualizes the gadget).

1. We establish a common baseline. For comparing changes in the performance of 2's root information sets, they need a common starting point: the original strategy $\sigma = (\sigma_1, \sigma_2)$ with its subgame portion $\sigma_1^S$.

   For every $I \in \mathcal{I}_2^{R(S)}$ we subtract the opponent's original counterfactual best-response value, setting the utility at each terminal node $z \in Z(I)$ to $\tilde{u}_2(\tilde{z}) := k_{\tilde{I}} \cdot u_2(z) - CBV_2^{\sigma}(I)$ (see Section I.5 for detailed explanation). We must not forget to update $\tilde{u}_1(\tilde{z}) = -\tilde{u}_2(\tilde{z})$ either, as we need the game to remain zero-sum. Conditioned on the original strategy $\sigma_1^S$, the shifting gives an expected value of 0 to opponent's starting states.

2. Player 2 is permitted to choose his belief at the start of the subgame, while 1 retains his belief from the original strategy at the starting point of the subgame. Since 2 is aiming to maximize $\tilde{u}_2$, he will always select the information set with the lowest margin. The minimax nature of the zero-sum game motivates player 1 to find a strategy maximizing this value of the lowest margin.

   We create an additional decision node $\tilde{d}$ for player 2. Every action corresponds to choosing an initial information set $I \in \mathcal{I}_2^{R(S)}$. However, since an action can lead only to a single tree node rather than a whole information set, we may not connect this action to state $\tilde{d}$ directly. Instead, each action leads to a new chance node $c_{\tilde{I}}$, where the chance distributes histories $\tilde{h} \in \tilde{I}$ based on the probability $\pi_{-2}^{\sigma}(h)$ and the normalization factor $k_{\tilde{I}} = \sum_{h \in I} \pi_{-2}^{\sigma}(h)$ (again, see Section I.5).

## I.5 Gadget-Game Counterfactual Values

As soon as the opponent picks which $c_{\tilde{I}}$ to enter (i.e., with which cards to enter the subgame), the chance player deals cards to us. The corresponding probability (of being in state $\tilde{h} \in \tilde{I}$) is $\pi_c(c_{\tilde{I}}, c_{\tilde{I}} \cdot \tilde{h}) = \frac{\pi^{\sigma}_{-2}(h)}{k_{\tilde{I}}}$ where normalization factor $k_{\tilde{I}} = \sum_{h \in I} \pi^{\sigma}_{-2}(h)$ ensures that probabilities sum to 1 (i.e., $\sum_{\tilde{h} \in \tilde{I}} \pi_c(c_{\tilde{I}}, c_{\tilde{I}} \cdot \tilde{h}) = 1$). The terminal utilities need to be adjusted for it. First, they are multiplied by the related normalization factor to cancel the effects of the normalization. Then they are shifted by the counterfactual values:

$$\tilde{u}_2(\tilde{z}) := k_{\tilde{I}} \cdot u_2(z) - CBV_2^{\sigma}(I) \tag{I.3}$$

The following lemma states that the construction shifts counterfactual values by the given (original) counterfactual values:

**Lemma 7.** *Let $\tilde{\sigma}$ be a subgame strategy profile and $\sigma' = \sigma_{[S \leftarrow \tilde{\sigma}]}$ its extension to the original game. Then the gadget-game counterfactual value of $I \in \mathcal{I}_2^{R(S)}$ is*

$$\tilde{v}_2^{\tilde{\sigma}}(\tilde{I}) = v_2^{\sigma'}(I) - CBV_2^{\sigma}(I).$$

*Proof.* A strategy profile $\sigma'$ defines probability distributions at each game state, and thus we get

$$\sum_{z \in Z(I)} \pi^{\sigma'}_{-2}(z) \cdot \pi_2^{\sigma'}(z[I], z) = \sum_{h \in I} \pi^{\sigma'}_{-2}(h) = \sum_{h \in I} \pi^{\sigma}_{-2}(h) = k_{\tilde{I}} \tag{I.4}$$

where the first equality is a recursive summation of probabilities to 1, by induction on height.

Therefore, we can re-write the counterfactual value in the gadget game as

$$\begin{aligned}
\tilde{v}_2^{\tilde{\sigma}}(\tilde{I}) &= \sum_{z \in Z(I)} \pi^{\tilde{\sigma}}_{-2}(\tilde{d}, \tilde{z}) \cdot \pi_2^{\tilde{\sigma}}(\tilde{z}[\tilde{I}], \tilde{z}) \cdot \tilde{u}_2(\tilde{z}) \\
&= \sum_{z \in Z(I)} \pi^{\tilde{\sigma}}_{-2}(\tilde{d}, \tilde{z}[\tilde{I}]) \cdot \pi^{\tilde{\sigma}}_{-2}(\tilde{z}[\tilde{I}], \tilde{z}) \cdot \pi_2^{\tilde{\sigma}}(\tilde{z}[\tilde{I}], \tilde{z}) \cdot \tilde{u}_2(\tilde{z}) \\
&= \sum_{z \in Z(I)} \frac{\pi^{\sigma}_{-2}(z[I])}{k_{\tilde{I}}} \cdot \pi^{\tilde{\sigma}}_{-2}(z[I], z) \cdot \pi_2^{\tilde{\sigma}}(z[I], z) \cdot \tilde{u}_2(\tilde{z}) \\
&= \sum_{z \in Z(I)} \frac{\pi^{\sigma}_{-2}(z[I])}{k_{\tilde{I}}} \cdot \pi^{\tilde{\sigma}}(z[I], z) \cdot \tilde{u}_2(\tilde{z}) \\
&\overset{\text{(I.3)}}{=} \sum_{z \in Z(I)} \frac{\pi^{\sigma}_{-2}(z[I])}{k_{\tilde{I}}} \cdot \pi^{\sigma'}(z[I], z) \cdot (k_{\tilde{I}} \cdot u_2(z) - CBV_2^{\sigma}(I)) \\
&= v_2^{\sigma'}(I) - \sum_{z \in Z(I)} \frac{\pi^{\sigma}_{-2}(z[I])}{k_{\tilde{I}}} \cdot \pi^{\sigma'}(z[I], z) \cdot CBV_2^{\sigma}(I) \\
&= v_2^{\sigma'}(I) - \sum_{z \in Z(I)} \frac{\pi^{\sigma'}_{-2}(z) \cdot \pi_2^{\sigma'}(z[I], z)}{k_{\tilde{I}}} \cdot CBV_2^{\sigma}(I) \\
&= v_2^{\sigma'}(I) - CBV_2^{\sigma}(I) \cdot \frac{\sum_{z \in Z(I)} \pi^{\sigma'}_{-2}(z) \cdot \pi_2^{\sigma'}(z[I], z)}{k_{\tilde{I}}} \\
&\overset{\text{(I.4)}}{=} v_2^{\sigma'}(I) - CBV_2^{\sigma}(I).
\end{aligned}$$

■

Hence, the gadget-game counterfactual values are shifted by right offsets and Lemma 7 therefore gives into context the gadget game and SM maximization: gadget-game counterfactual values at $\mathcal{I}_2^{R(S)}$ correspond to margins. Thus, solving the gadget translates to maximizing the minimum of margins, i.e., maximizing the subgame margin.

**Corollary 4.** *A strategy for the max-margin gadget is a Nash equilibrium if and only if it is a solution to the LP (I.2).*

## I.6 Experimental Results

> If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.
>
> Donald Knuth

In this section, we evaluate endgame solving (Chapter G), subgame re-solving (Chapter H) and max-margin subgame refinement (this chapter) on the safe-refinement task for a large-scale game. We use an improved version of the *Nyx* agent, the second strongest participant at the 2014 *Annual Computer Poker Competition* (No-Limit Texas Hold'em Total Bankroll)[1] as the baseline strategy to be re-fined in subgames.

All three of the subgame refinement techniques tested here used the same abstractions and trunk strategy. Following (Ganzfried and Sandholm 2015), we begin the subgame at start of the last round (the river). Although we used card abstraction to compute the original trunk strategy (Johanson et al. 2013; Schmid et al. 2015), the fine-grained abstraction for the endgame is calculated *without the need for card abstraction*. This is an improvement over the original implementation of (Ganzfried and Sandholm 2015), where both the trunk strategy and the refined subgame used card abstraction. This is a result of the improved efficiency of the CFR$^+$ algorithm (and the domain-specific speedups it enables), whereas the endgame solving of (Ganzfried and Sandholm 2015) used linear programming to compute the strategy.

The original strategy uses action abstraction with up to 16 actions in an information set. While this number is relatively large compared to other participating agents, it is still well below the best-known upper bound on the optimal strategy's support size (Schmid, Moravčík, and Hladík 2014). The action abstraction used for the original Nyx strategy has the imperfect recall, whereas the refined subgame uses the perfect recall. We use the same actions in the refined subgame as in the original strategy.

[1] http://www.computerpokercompetition.org/index.php/competitions/results/105-2014-results

We refine only the subgames that has less than 1000 betting sequences (after creating the fine-grained abstraction). This is simply to speed up the experiments. The original agent's strategy is preserved in the trunk of the game for both player 1 and player 2. Once gameplay reaches the subgame (the river), we refine 1's strategy using each of the three techniques. We have run 10000 iterations of the CFR$^+$ algorithm in the corresponding gadget games, with exponential weighting to update the average strategies (Tammelin et al. 2015). Each technique was used to refine $\approx 2000$ subgames. Figure I.2 displays the average margins for the evaluated techniques.

The max-margin technique produces the optimal value, much greater than ones produced by subgame re-solving or endgame solving (which has even negative subgame margins).

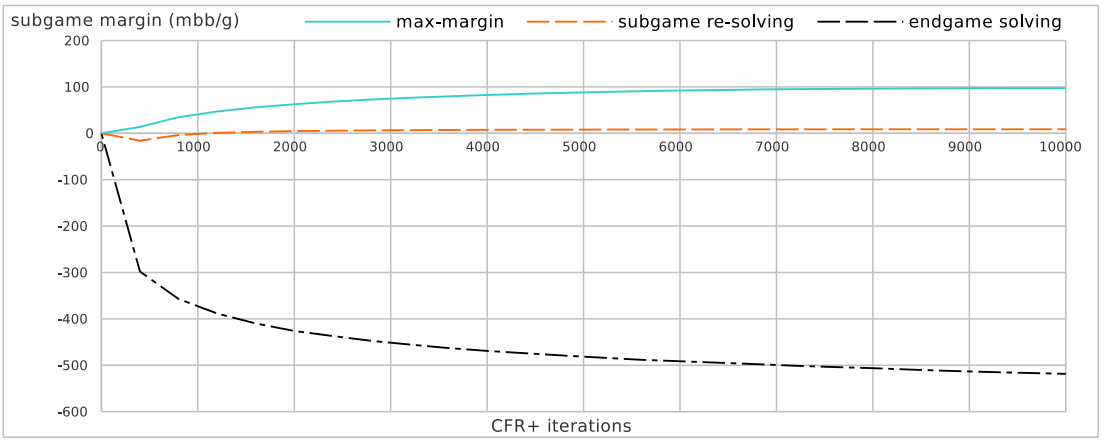



Figure I.2: CFR$^+$ iterations versus SM of refined strategies(in milli big blinds per game). One big blind corresponds to 100 chips.

Here follows the list of 95%-confidence intervals after 10000 iterations:

**endgame solving** $(-518.5 \pm 49.19)$. The considerably negative margin values for the endgame solving suggest that the produced strategy may indeed be much more exploitable.

**subgame re-solving** $(8.79 \pm 2.45)$. The positive margin for re-solving demonstrates that in spite of no explicit construction forcing the margin to be positive, it does increase in practice. Notice however that the margin is far below the optimal level.

**max-margin** $(101.49 \pm 7.09)$. This technique produces a significantly larger subgame margin than the previous methods. The size of the margin suggests the original strategy is potentially rather exploitable. Nevertheless, our technique can substantially decrease the exploitability (see Corollary 3 and Theorem 6).

# J. Ideas for Future Work

> The worst thing that can happen in a democracy—as well as in an individual's life—is to become cynical about the future and lose hope.
>
> Hillary Clinton

One may try to study other aspects of endgames. Some recommendations as inspiration for future work may include

- ♠ other endgame-specific concepts like subgame margin
- ♠ dynamical adjustment of computation upon entering the ending phase
- ♠ solving several endgames in parallel, without interdependence
- ♠ a Poker version of Chess endgame tablebases, which poker agents may use in simpler ending situations to play perfectly

## J.1 Margins via Multi-Objective Optimization

> Raise your quality standards as high as you can live with, avoid wasting your time on routine problems, and always try to work as closely as possible at the boundary of your abilities. Do this, because it is the only way of discovering how that boundary should be moved forward.
>
> Edsger Dijkstra

One especially noteworthy research topic is the study of margins (i.e., declines of CBVs). The subgame margin is their min-aggregation. Tempting idea is to look into alternative aggregations. Consider a multi-criteria analogy to subgame margin maximization (I.2):

$$\begin{aligned} \max_{v,x}\ & \boldsymbol{m} \\ v_I - \boldsymbol{m}_I &\geq CBV_2^{\sigma_1}(I), \quad I \in \mathcal{I}_2^{R(S)} \\ Ex &= e \\ F^\top v - A_2^\top x &\leq \mathbf{0} \\ x &\geq \mathbf{0}. \end{aligned} \tag{J.1}$$

Now $\boldsymbol{m}_I$ corresponds to one specific margin $\boldsymbol{m}_I := CBV_2^{\sigma_1}(I) - CBV_2^{\sigma_1'}(I)$, and $\boldsymbol{m} := (m_I)_{I \in \mathcal{I}_2^{R(S)}}$ is a vector of all such margins. Evidently, this vector linear program (VLP) is a task of *multi-objective optimization* (Ehrgott 2006; Grygarová 1996).

Notice that (I.2) is just a special case of the vector objective function, by the scalarization

$$\max_{v,x} \min_{I \in \mathcal{I}_2^{R(S)}} \boldsymbol{m}_I.$$

The choice of minimum function stems from the nature of the opponent, who makes his best to exploit us and hence aims for minimal decrease in CBVs.

Undoubtedly, one can also try other methods to deal with multi-objective functions:

- ♠ other scalarizations, e.g., weighted sum of margins
- ♠ (Pareto) efficient solutions
- ♠ data envelopment analysis (DEA): http://www.deazone.com/
- ♠ evolutionary multi-objective optimization (EMO), which uses evolutionary algorithms: https://www.cs.cinvestav.mx/∼EVOCINV/

It might be compelling as well as intriguing to understand their meaning in the language of extensive-form games, and perhaps even find their corresponding gadget games.

# Conclusion

> Everything that civilisation has to offer is a product of human intelligence; we cannot predict what we might achieve when this intelligence is magnified by the tools that AI may provide, but the eradication of war, disease, and poverty would be high on anyone's list. Success in creating AI would be the biggest event in human history. Unfortunately, it might also be the last.
>
> Stephen Hawking

Perfect-information endgames can be solved by backward induction, where solutions to subgames are propagated up the game tree. For the imperfect information, on the other hand, endgames need to be adjusted in order to account for information sets. This gives rise to a new definition of an (imperfect-information) *subgame*. As such, the definition does not directly allow to apply the same procedure as in perfect-information endgames: under even the simplest conditions of the Rock-Paper-Scissors game, a naïve endgame re-solution fails to form a Nash equilibrium.

This occurs because the opponent can change his behavior prior to the endgame. Such an exploitation power can be captured by a *counterfactual value*: a hypothetical "what-if" value summarizing opponent's improvement, if he had changed his prior trunk strategy.

We used counterfactual values to define our own notion of *subgame margin*: a gap between the original and the new counterfactual best-response values. We related the SM to the exploitability against a best response, and we proved the overall improvement rising from endgame improvement is proportional to the SM.

Maximizing SM is thus highly advisable. That can be achieved either by solving a variant of a sequence-form linear program, or by applying an iterative learning algorithm to a *gadget game*, an equivalent ad-hoc extensive-form game. The latter approach offers greater benefits in terms of exploiting domain-specific knowledge and employing powerful learning algorithms such as CFR, MCCFR or the modern CFR$^+$ that we chose to solve our *max-margin gadget*.

Finally, we experimentally compared the three contemporary approaches to solving endgames:

(i) endgame solving (Ganzfried and Sandholm 2015)

(ii) CFR-D and decomposition (Burch, Johanson, and Bowling 2014)

(iii) our subgame-margin maximization (Moravčík et al. 2016)

The results of the experiments showed that (i) even produced a worse SM, leading to more exploitable subgame strategies; and although the SM of (ii) increased over time, the improvement was not significant, as the method guarantees only the same (or comparable) quality, not the best one. Since our (iii) was specially designed to maximize SMs, it re-created the most robust (i.e., the least exploitable) subgame strategy. We thus offer a superior solution to treating subgames.

# Bibliography


### Alburt et al.: Just the Facts!: Winning Endgame Knowledge in One Volume

L. Alburt and N. Krogius. *Just the Facts!: Winning Endgame Knowledge in One Volume*. Comprehensive chess course series. Chess Information and Research Center, 1999. ISBN: 9781889323060.

### Bellman: On the Application of Dynamic Programing to the Determination of Optimal Play in Chess and Checkers

Richard Bellman. "On the Application of Dynamic Programing to the Determination of Optimal Play in Chess and Checkers". In: *Proceedings of the National Academy of Sciences of the United States of America* 53.2 (1965), p. 244.

### Berlekamp et al.: Winning Ways for Your Mathematical Plays: Games in Particular

E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays: Games in Particular*. Vol. 2. Acad. Pr., 1983. ISBN: 9780120911523.

### Bourzutschky: 7-man Endgames with Pawns

Marc Bourzutschky. *7-man Endgames with Pawns*. 2006. URL: `http://kirill-kryukov.com/chess/discussion-board/viewtopic.php?t=805` (visited on 06/24/2016).

### Bowling et al.: Heads-Up Limit Hold'em Poker is Solved

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. "Heads-Up Limit Hold'em Poker is Solved". In: *Science* 347.6218 (2015), pp. 145–149. URL: `http://poker.cs.ualberta.ca/15science.html`.

### Bošanský: Solving Extensive-Form Games with Double-Oracle Methods

Branislav Bošanský. "Solving Extensive-Form Games with Double-Oracle Methods". In: *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 1423–1424.

### Burch et al.: Solving Imperfect Information Games Using Decomposition

N. Burch, M. Johanson, and M. Bowling. "Solving Imperfect Information Games Using Decomposition". In: *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence* (2014).

**Conway: On Numbers and Games**

John Horton Conway. "On Numbers and Games". In: *London Mathematical Society Monographs* 6 (1976).

**Ehrgott: Multicriteria optimization**

Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

**Enderton: The Golem Go program**

Herbert D. Enderton. "The Golem Go program". In: *Carnegie Mellon University* (1991).

**Fine: The Middle Game in Chess**

Reuben Fine. *The Middle Game in Chess*. D. McKay Co., 1952.

**Flear: Practical Endgame Play - Beyond the Basics: The Definitive Guide to the Endgames That Really Matter**

G. Flear. *Practical Endgame Play - Beyond the Basics: The Definitive Guide to the Endgames That Really Matter*. Everyman Chess. Everyman Chess, 2007. ISBN: 9781857445558.

**Fraenkel et al.: Computing a Perfect Strategy for $n \times n$ Chess Requires Time Exponential in $n$**

Aviezri S. Fraenkel and David Lichtenstein. "Computing a Perfect Strategy for $n \times n$ Chess Requires Time Exponential in $n$". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1981, pp. 278–293.

**Ganzfried et al.: Endgame Solving in Large Imperfect-Information Games**

Sam Ganzfried and Tuomas Sandholm. "Endgame Solving in Large Imperfect-Information Games". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2015, pp. 37–45.

**Grygarová: Základy Vícekriteriálního Programování**

Libuše Grygarová. *Základy Vícekriteriálního Programování*. 1st ed. Charles University, 1996.

**Haworth: Discarding Like Pieces**

G. McC. Haworth. "Discarding Like Pieces". In: *ICGA JOURNAL* 24.3 (2001), pp. 161–161.

**Herik et al.: A 6-Men-Endgame Database: KRP(a2)KBP(a3)**

H. Jaap Herik, Israel Samuel Herschberg, and Najib Nakad. "A 6-Men-Endgame Database: KRP(a2)KBP(a3)". In: *ICCA JOURNAL* 10.4 (1987), pp. 163–180.

**Hoda et al.: Smoothing Techniques for Computing Nash Equilibria of Sequential Games**

Samid Hoda, Andrew Gilpin, Javier Pena, and Tuomas Sandholm. "Smoothing Techniques for Computing Nash Equilibria of Sequential Games". In: *Mathematics of Operations Research* 35.2 (2010), pp. 494–512.

**Johanson: Measuring the Size of Large No-Limit Poker Games**

Michael Johanson. "Measuring the Size of Large No-Limit Poker Games". In: *arXiv preprint arXiv:1302.7008* (2013).

**Johanson et al.: Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization**

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 11th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 2. International Foundation for Autonomous Agents and Multiagent Systems. 2012, pp. 837–846.

**Johanson et al.: Evaluating State-Space Abstractions in Extensive-Form Games**

Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. "Evaluating State-Space Abstractions in Extensive-Form Games". In: *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 271–278.

**Karpov: Disney's Chess Guide**

Anatoly Karpov. *Disney's Chess Guide*. Batsford, 1997. ISBN: 9780713483352.

**Koller et al.: Fast Algorithms for Finding Randomized Strategies in Game Trees**

Daphne Koller, Nimrod Megiddo, and Bernhard Von Stengel. "Fast Algorithms for Finding Randomized Strategies in Game Trees". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. ACM. 1994, pp. 750–759.

**Krabbé: Stiller's Monsters or Perfection in Chess**

Tim Krabbé. *Stiller's Monsters or Perfection in Chess*. 2014. URL: `https://timkr.home.xs4all.nl/chess/perfect.htm` (visited on 06/24/2016).

**Levy et al.: How Computers Play Chess**

D. N. L. Levy and M. Newborn. *How Computers Play Chess*. Ishi Press, 2009. ISBN: 9784871878012.

**Lomonosov Tablebases: 8 Longest 7-Man Checkmates**

Lomonosov Tablebases. *8 Longest 7-Man Checkmates*. 2014. URL: `http://ldis-sw.cs.msu.ru/articles/Top8DTM_eng` (visited on 06/24/2016).

**Minev: A Practical Guide to Rook Endgames**

Nikolay Minev. *A Practical Guide to Rook Endgames*. Russell Enterprises, 2004.

**Moravčík et al.: Refining Subgames in Large Imperfect Information Games**

Matěj Moravčík, Martin Schmid, Karel Ha, Milan Hladík, and Stephen J. Gaukrodger. "Refining Subgames in Large Imperfect Information Games". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

**Muller: EGTB Generator**

H. G. Muller. *EGTB Generator*. 2006. URL: `http://home.hccnet.nl/h.g.muller/EGTB.html` (visited on 06/24/2016).

**Müller: Computer Go as a Sum of Local Games: an Application of Combinatorial Game Theory**

Martin Müller. "Computer Go as a Sum of Local Games: an Application of Combinatorial Game Theory". PhD thesis. TU Graz, 1995.

**Nisan et al.: Algorithmic Game Theory**

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521872820.

**Nunn: Secrets of Pawnless Endings**

J. Nunn. *Secrets of Pawnless Endings*. Gambit, 2002. ISBN: 9781901983654.

**Osborne et al.: A Course in Game Theory**

Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

**Portisch et al.: Six Hundred Endings**

L. Portisch and B. Sárközy. *Six Hundred Endings*. Pergamon Press, 1981.

**Schaeffer et al.: Checkers is Solved**

Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. "Checkers is Solved". In: *Science* 317.5844 (2007), pp. 1518–1522.

**Schmid et al.: Bounding the Support Size in Extensive Form Games with Imperfect Information**

Martin Schmid, Matěj Moravčík, and Milan Hladík. "Bounding the Support Size in Extensive Form Games with Imperfect Information". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

**Schmid et al.: Automatic Public State Space Abstraction in Imperfect Information Games**

Martin Schmid, Matěj Moravčík, Milan Hladík, and Stephen J. Gaukrodger. "Automatic Public State Space Abstraction in Imperfect Information Games". In: *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

**Schraudolph et al.: Temporal Difference Learning of Position Evaluation in the Game of Go**

Nicol N. Schraudolph, Peter Dayan, and Terrence J. Sejnowski. "Temporal Difference Learning of Position Evaluation in the Game of Go". In: *Advances in Neural Information Processing Systems* (1994), pp. 817–817.

**Selten: An Oligopoly Model with Demand Inertia**

Reinhard Selten. *An Oligopoly Model with Demand Inertia*. Center for Research in Management Science, University of California, 1968.

**Shukaku: How Many Moves is it Possible to Read?**

Takagawa Shukaku. "How Many Moves is it Possible to Read?" In: *Go World* 41 (1985). The Ishi Press, Inc., pp. 30–33.

**Silver et al.: Mastering the Game of Go with Deep Neural Networks and Tree Search**

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (2016), pp. 484–489.

**Speelman: Endgame Preparation: Advanced Analysis of Important Areas**

Jonathan Speelman. *Endgame Preparation: Advanced Analysis of Important Areas*. BT Batsford, 1981.

**Stoutamire: Machine Learning, Game Play, and Go**

David Stoutamire. "Machine Learning, Game Play, and Go". In: *Case Western Reserve University, Tech. Rep* (1991), pp. 91–128.

**Tammelin et al.: Solving Heads-Up Limit Texas Hold'em**

Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. "Solving Heads-Up Limit Texas Hold'em". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

**Von Neumann et al.: Theory of Games and Economic Behavior**

John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. `https://archive.org/details/theoryofgamesand030098mbp`. Princeton University Press, 1953.

**Wikipedia: Endgame tablebase — Wikipedia, The Free Encyclopedia**

Wikipedia. *Endgame tablebase — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Endgame%20tablebase&oldid=730651496`. [Online; accessed 26-July-2016]. 2016.

**Wilcox: Computer Go - The Reitman-Wilcox Program**

Bruce Wilcox. "Computer Go - The Reitman-Wilcox Program". In: *American Go Journal* 14 (1979), pp. 23–41.

**Zinkevich et al.: Regret Minimization in Games with Incomplete Information**

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. "Regret Minimization in Games with Incomplete Information". In: *Advances in Neural Information Processing Systems*. 2007, pp. 1729–1736.

**Čermák et al.: Practical Performance of Refinements of Nash Equilibria in Extensive-Form Zero-Sum Games**

Jiří Čermák, Branislav Bošanský, and Viliam Lisý. "Practical Performance of Refinements of Nash Equilibria in Extensive-Form Zero-Sum Games". In: ECAI. 2014.

# List of Figures

# List of Tables

# List of Abbreviations

**AI** artificial intelligence. 5, 26

**ANN** artificial neural network. 36, 37

**BR** best response. 13, 14, 16, 43, 49–51, 60, 62, 63, 69

**CBR** counterfactual best response. 43, 56

**CBV** counterfactual best-response value. 44, 47, 53, 56, 57, 60, 61, 63, 67–69

**CFR** counterfactual regret minimization. 47, 48, 57, 65, 66, 69, 76

**CFR-D** counterfactual regret decomposition. 51, 58, 69

**CGT** combinatorial game theory. 3, 5, 10, 18, 26–29, 34

**CNN** convolutional neural network. 3, 37, 38

**DEA** data envelopment analysis. 68

**DNN** deep neural network. 37

**DTM** depth to mate. 21

**EFG** extensive-form game. 3, 4, 12, 43, 44, 47, 59, 62, 68, 69

**EMO** evolutionary multi-objective optimization. 68

**ISP** Internet Service Provider. 7, 76

**LHE** Limit Texas Hold'em. 5, 52, 55, 58

**LP** linear program. 44–48, 51–53, 57, 59, 62, 65, 69

**MCCFR** Monte Carlo counterfactual regret minimization. 47, 69

**MCTS** Monte Carlo tree search. 36, 38, 76

**NE** Nash equilibrium. 14, 17, 47, 48, 55, 58, 65, 69

**NFG** normal-form game. 4, 7

**NLHE** No-Limit Texas Hold'em. 5, 52, 65

**NQE** Not Quite an Endgame. 19

**RPS** Rock-Paper-Scissors. 3, 4, 6, 14, 49–51, 54, 57, 69, 76

**SM** subgame margin. 3, 59, 60, 65–67, 69, 76

**VLP** vector linear program. 67