DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Development and Analysis of Strategies for the Card Game "The Game"

Felix Dietrich

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Development and Analysis of Strategies for the Card Game "The Game"

Entwicklung und Analyse von Strategien für das Kartenspiel „The Game“

Author:	Felix Dietrich
Supervisor:	Prof. Dr. Harald Räcke
Advisor:	Prof. Dr. Harald Räcke
Submission Date:	September 16, 2019

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, September 16, 2019 Felix Dietrich

Contents

1 Introduction

The Game by Steffen Benndorf and published by Nürnberger-Spielkarten-Verlag GmbH [Ben15] is a cooperative patience card game for up to 5 players. The players work together and form four piles by playing cards onto them. Among the piles, there are two piles in ascending order starting at number 1, and two piles in descending order starting at 100. The deck consists of 98 shuffled cards with numbers ranging from 2 to 99. The goal of the game is to lay as many cards as possible onto the piles, and it is won if the players manage to lay down all cards. At the start of the game each player draws 8 cards which make up the player's hand. Turns are made in clockwise direction and a player can play a turn as follows. On ascending piles, the card laid on top has to be greater than the last laid card on the pile. The same holds for descending piles in reverse order. There is an exception to this rule, a recover mechanic called the backwards trick which allows to play cards exactly 10 lower or higher in reverse order. At least two cards have to be played before a player can end its turn, but one card at a time may be played if there are no cards to be drawn. After ending a turn, the player refills the hand to 8 cards again from the deck, if possible. The game is lost when a player cannot complete its turn by playing 2 or more cards. Ideally, all cards have to be played to win the game, but the creator of the game says, that having less than 10 cards remaining is an excellent outcome.
The players are allowed to communicate, but they are not allowed to disclose any numbers. For example, a player might say: "Leave this pile to me", "Only play a small step on this pile" or similar phrases. Even though the players are allowed to communicate, this paper focuses on the case where no communication is allowed since communication seems difficult to simulate using the computer. Since we don't allow communication between multiple players, we can regard the game as a single player game where every player plays the same strategy for the sake of this thesis.
The Game is simple, but it seems very difficult to win. It is not obvious what the best tactics are for approaching this card game. While playing, some heuristics come to mind, but it is unknown how good they really are. The aim of this thesis is to find strategies for beating The Game and shed a light on how well they perform, and on how they behave.
The approach for finding a strategy in this thesis is to do positional evaluation similar to the way chess is solved on the computer [Sha50]. The strategy is to pick the best

turn which leads into the most advantageous position. This means that strategies in this thesis are defined by an evaluation function giving positions a certain value.
Even though the mechanics of the game have not that much in common with most other games and problems, the core ideas involved in the presented strategies might be transferred to other domains.

2 Fundamentals

This chapter introduces the parameters, model, and notation used throughout the thesis. It also covers how the performance of a strategy will be evaluated for chapter 3.

2.1 Related Work

An analysis of the complexity of The Game [KR] has previously been conducted. It considers the one-player version where the deck is known and reveals, that one can decide that a game is winnable for a given deck in polynomial time. This result does not help us with how the player should play for games where the deck is unknown.
The game Tetris can be considered as similar to The Game. It contains multiple column i.e. piles where differently shaped pieces i.e. cards are laid onto, decreasing the capacity to the top limit of the column. Filling a row across all columns removes a row, which is comparable to the backwards trick. Approximate dynamic programming and reinforcement learning are used to tackle Tetris [GGS13], but The Game has some fundamental differences. First, the distribution of cards to draw changes as opposed to uniformly picking from 6 pieces. And second, Tetris lays pieces over multiple columns using up their capacity, whereas in The Game, playing a card sets the pile card to the played card. Using a form of dynamic programing [Smi07] might therefore be harder to do. Reinforcement learning [Hay+09], where the computer can learn a strategy by repeatably playing and altering it, could be applied for The Game, but this thesis focuses on using a positional evaluation similar to chess [Sha50] as basis for the presented strategies.
Another game of Steffen Benndorf called Quixx [Ben12] is similar to The Game, but no previous work has been found for that game. It is a competitive dice game where essentially instead of drawing cards as in The Game, the players throw dices and play numbers resulting from two dices onto a pile. Every player has their own set of piles and there is no form of recovering a pile with a backwards trick. If a player reaches the end of a pile by playing 12 or 2 and has 5 or more cards played on it, the player can lock the pile for the other players to prevent them from getting more points on this pile. The goal is to get a higher score by fitting more numbers onto a pile than the opponents. The key difference to The Game is, that the distribution of the numbers is

known and that there are elements of game theory involved, because the players react to what their opponents do.

2.2 Parameterization

In the following section, possible parameters of the game are being defined so that they can be easily referred to. Only the parameters for the default rules of the game [Ben15] have been used in this paper. Other variants which make the game harder, such as having less hand cards available or being forced to play more cards each turn, are not analyzed. Nevertheless, all listed parameters can easily be changed in the C-program used to simulate the strategies. A brief overview of the parameters and their default configuration can be seen in Table 2.1.

Table 2.1: Parameters of all quantities of The Game.

Symbol	Default	Parameter
$c_\downarrow$	2	Lowest playable card
$c_\uparrow$	99	Highest playable card
$p_{\#\uparrow}$	2	Number of piles in ascending order
$p_{\#\downarrow}$	2	Number of piles in descending order
t_{min}	2	Number of minimum cards to play in a turn
h	8	Hand capacity
b	10	Backwards trick distance

The set of cards used in the game is $C = \{c_\downarrow, \dots, c_\uparrow\}$, where $c_\downarrow \in \mathbb{Z}$ the lowest and $c_\uparrow \in \mathbb{Z}$ the highest playable card, with $c_\downarrow < c_\uparrow$. There are $p_{\#\uparrow} \in \mathbb{N}_0$ piles in ascending order and $p_{\#\downarrow} \in \mathbb{N}_0$ in descending order, with $p_{\#\uparrow} + p_{\#\downarrow} > 0$. This means that the program supports configurations where $p_{\#\uparrow} \neq p_{\#\downarrow}$ and where only either ascending or descending piles exist. The hand capacity $h \in \{1, 2, \dots, |C|\}$ denotes the number of cards the player can have in its hand. At the start of the game and after each turn the player tries to refill its hand to h hand cards from the deck. The minimum number of cards to play in a turn when there are still cards left in the deck is denoted by $t_{min} \in \{1, 2, \dots, h\}$. For the backwards trick, we call the distance the player is allowed to play a card onto a pile in reverse order the backwards trick distance $b \in \{0, 1, \dots, |C| - 1\}$. Setting the backwards trick distance to 0 would prohibit the use of the backwards trick.

2.3 Model and Notation

This section covers the model of the game and the notations used throughout the paper.

2.3.1 Indexing of the Piles

There are $p_{\#\uparrow}$ piles in ascending and $p_{\#\downarrow}$ piles in descending order. We give the piles a fixed order and refer to them with the pile index set $P = P^{\uparrow} \cup P^{\downarrow}$, where $P^{\uparrow} = \{1^{\uparrow}, \dots, p_{\#\uparrow}{}^{\uparrow}\}$ the index set for ascending piles and $P^{\downarrow} = \{1^{\downarrow}, \dots, p_{\#\downarrow}{}^{\downarrow}\}$ the index set for descending piles. With this, $k^{\uparrow} \in P^{\uparrow}$ refers to the k^{th} pile in ascending order, and $k^{\downarrow} \in P^{\downarrow}$ refers to the k^{th} pile in descending order. See Figure 2.1, for an example pile indexing, where the piles located on the left-hand side are identified with the index set $P = \{1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow}\}$.

2.3.2 State

A state s of the game can be fully described by the tuple

$$s = (L, H, p_{\uparrow}, p_{\downarrow}),$$

where $L \subseteq C = \{c_{\downarrow}, \dots, c_{\uparrow}\}$ is the set of already laid cards, $H \subseteq C$ is the set of cards in the hand, and $p_{\uparrow} \in \{c_{\downarrow} - 1, \dots, c_{\uparrow}\}^{p_{\#\uparrow}}$, $p_{\downarrow} \in \{c_{\downarrow}, \dots, c_{\uparrow} + 1\}^{p_{\#\downarrow}}$ are tuples of the state of the piles in ascending and descending order respectively. The cards which remain in the deck D can be constructed with $D = C \setminus (L \cup H)$.
An example state for the default game configuration can be seen in Figure 2.1. In this figure, card 34 has just been played onto the pile with index $2^{\uparrow}$ as first card of the turn. This illustrates that a state captures every moment of the game where a card has been laid down and not just moments where full turns have been played. Note that in the figure, the piles state of descending piles $p_{\downarrow} = (100, 80)$ contains $(p_{\downarrow})_1 = 100 = c_{\uparrow} + 1$, a card not contained in the deck, to indicate that no card has been played onto pile $1^{\downarrow}$ so far. The same holds for ascending piles where $1 = c_{\downarrow} - 1$ indicates that no card has been played onto an ascending pile. The initial pile states are therefore $p_{\uparrow_{init}} \in \{c_{\downarrow} - 1\}^{p_{\#\uparrow}}$ and $p_{\downarrow_{init}} \in \{c_{\uparrow} + 1\}^{p_{\#\downarrow}}$ for the piles in ascending and descending order. Cards that are played are directly added to the set of laid cards L, as seen in Figure 2.1 where card 34 has just been played. The state does not store where each card has been laid onto, it suffices that only the top cards of the piles are saved in $p_{\uparrow}$ and $p_{\downarrow}$ for the presented strategies. The pile cards could have been modeled as a set to reduce the state space, but the piles needed to be identifiable especially for the weighted capacity strategies introduced later in section 3.3. Both the order of the laid card L and the hand cards H is not important for our decision making, they are therefore represented by a set.
The set of all initial states is given by $S_{init} = \{\varnothing\} \times \binom{C}{h} \times \{c_{\downarrow} - 1\}^{p_{\#\uparrow}} \times \{c_{\uparrow} - 1\}^{p_{\#\downarrow}}$, where

$\binom{C}{h} = \{A \in \mathcal{P}(C) \mid |A| = h\}$. Note that the number of initial states is rather large with $|S_{init}| = \binom{|C|}{h}$ states.
For referring to the different components of state $s = (L, H, p_\uparrow, p_\downarrow)$, introduce projections

$$\begin{aligned} L &: S \to \mathcal{P}(C),\ L(s) = L, \\ H &: S \to \mathcal{P}(C),\ H(s) = H, \\ p_c &: S \times P \to C,\ p_c(s,p) = \begin{cases} (p_\uparrow)_i & \text{if } i^\uparrow = p \\ (p_\downarrow)_i & \text{if } i^\downarrow = p. \end{cases} \end{aligned}$$

$L(s)$ yields the laid cards, $H(s)$ the hand cards, and $p_c(s,p)$ the pile card at pile index p of state s. Additionally, introduce $D(s) = C \setminus (L(s) \cup H(s))$ as shorthand for referring to the remaining cards in the deck.

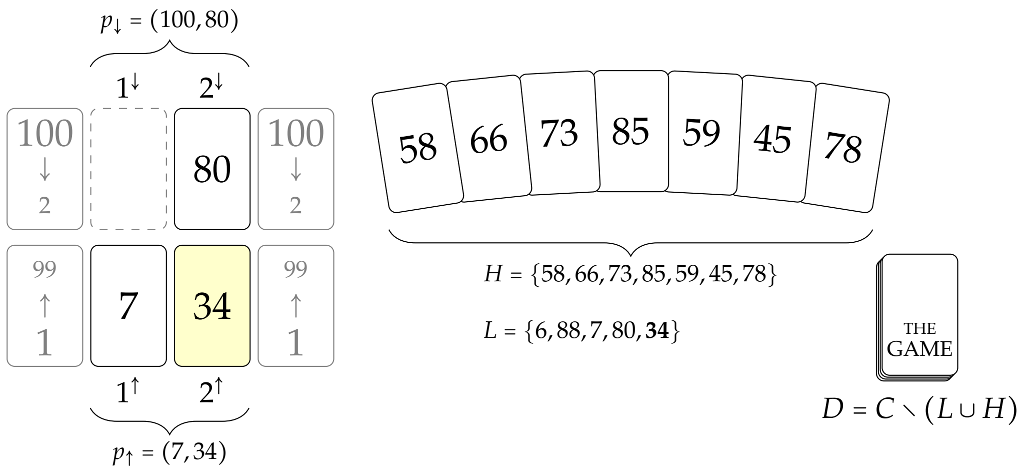

Figure 2.1: Example state for the default parameter configuration where card 34 has just been played onto pile $2^\uparrow$ as first card of the turn.

2.3.3 Move

A move $m = (c, p) \in C \times P$ is a tuple consisting of the card to play c and, the location where the card is played onto, the pile index p. The set of all possible moves $M(s)$ starting from state s is given by

$$M(s) = \left\{ m = (c,p) \in H(s) \times P \;\middle|\; \begin{array}{l} p \in P^\uparrow \Rightarrow p_c(s,p) < c \vee p_c(s,p) - b = c, \\ p \in P^\downarrow \Rightarrow p_c(s,p) > c \vee p_c(s,p) + b = c \end{array} \right\}.$$

State s can be transitioned with move $m = (c, p) \in M(s)$ into the state s' with the following transition function

$$t(s, m) = (L \cup \{c\}, H \setminus \{c\}, p'_\uparrow, p'_\downarrow) = s',$$

with

$$(p'_\uparrow, p'_\downarrow) = \begin{cases} ((p_{\uparrow_1}, \dots, p_{\uparrow_{i-1}}, c, p_{\uparrow_{i+1}}, \dots, p_{\uparrow_{p_{\#\uparrow}}}), p_\downarrow) & \text{if } i^\uparrow = p \in P^\uparrow \\ (p_\uparrow, (p_{\downarrow_1}, \dots, p_{\downarrow_{i-1}}, c, p_{\downarrow_{i+1}}, \dots, p_{\downarrow_{p_{\#\downarrow}}})) & \text{if } i^\downarrow = p \in P^\downarrow, \end{cases}$$

and $s = (L, H, (p_{\uparrow_1}, \dots, p_{\uparrow_{p_{\#\uparrow}}}), (p_{\downarrow_1}, \dots, p_{\downarrow_{p_{\#\downarrow}}}))$. This transition will also be denoted with the arrow notation $s \xrightarrow{m} s'$.

2.3.4 Turn

A turn starts at state s with a full hand if the deck is not empty $D(s) \neq \emptyset$. The player plays a sequence of n moves $s \xrightarrow{m_1} s_1 \xrightarrow{m_2} \dots \xrightarrow{m_n} s_n$ with $n \geq t_{min}$ as a turn and then draws again from the deck to refill its hand. According to the rules of The Game the player is allowed to play one card at a time as turn if the deck is empty $D(s) = \emptyset$. For the purpose of simplicity, we ignore this case and require that each turn has to be a sequence of $n \geq t_{min}$ moves.
Let $T_n : S \to \mathcal{P}(S)$ be the function which returns all from a starting state reachable states by doing exactly $n \in \mathbb{N}_0$ consecutive moves, playing n cards. T_n is given by

$$T_n(s) = \begin{cases} \bigcup_{m \in M(s)} T_{n-1}(t(s, m)) & \text{if } n > 0 \\ \{s\} & \text{if } n = 0 \\ \emptyset & \text{otw.} \end{cases}$$

The set $T_n(s)$ is a set of states. It does not contain any information on what moves have been taken to get to them, it only contains the states the moves end up in.

2.3.5 End of Turn

State $s' = (L, H, p_\uparrow, p_\downarrow) \in T_n(s)$ can be reached by playing a turn onto state s with $n \in \{t_{min}, \dots, |H|\}$ cards. This state has to draw cards from the deck $D(s)$ again to refill the hand. To do this $m = h - |H|$ cards are uniquely chosen uniformly at random from the deck $D(s)$ and added to the set R, if $|D|(s) > m$. If $|D(s)| \leq m$, all remaining cards will be drawn from the deck, therefore $R = D(s)$. Introduce a new state $s_{ref} = (L, H \cup R, p_\uparrow, p_\downarrow)$ where the cards R are drawn, and continue to play the game from s_{ref} onwards.

2.3.6 No More Turns

If $T_{t_{min}}(s) = \varnothing$ for state s, no full turn can be played anymore. There might still be playable cards in the hand $H(s)$ but the player can no longer draw any new cards. In chapter 3, strategies are played as long as there are turns, and the final moves are picked without a particular strategy. If there is no turn to play, one can just check all possible remaining move sequences doable with the cards in hand $H(s)$ and pick the longest sequence. Just picking the longest sequence of moves without a strategy is possible as soon as the deck is empty, but the number of possible move sequences can be potentially very large in some rare cases. This might come at a hindrance if a large sample size of games is simulated, but this is just speculation and has not been tested. We just assume that a good strategy takes the best steps on its own. Because of this we continue to do turns according to the definition of a turn in subsection 2.3.4 with a strategy until no full turn can be played any more, even if no strategy is needed to do so.

2.4 Simulation

We want to analyze strategies and measure how well each strategy performs. There are two key performance measurements for a strategy: First the average number of remaining cards at the end of the game, and second the more important measurement the win rate indicating how many games are won on average. Generally, these measurements should correlate to each other but there might be a strategy which plays riskier and loses earlier but wins more often on average than another strategy. This means that a strategy with a lower mean of remaining cards does not necessarily have a higher win rate than a strategy with a higher mean of remaining cards.
A way of getting exact values is to play a strategy for each permutation of the deck. This is computational infeasible because of $|C|!$, the number of permutations. This number is too large. Checking $98! \approx 10^{154}$ permutations for the default game configuration is too costly to compute. Finding a formula which easily calculates the win rate for a strategy by theoretical means seams to be also infeasible. Therefore, an approximation of the two measurements has to be done by simulating a smaller sample size $n \ll |C|!$ of games, and then estimating the measurements based on this sample.

2.4.1 Estimators for the Measurements

To approximate the win rate and the average number of cards remaining at the end of the game for a strategy, we take a sample of size n of random permutations of the deck and play the strategy. For each game played $i \in [n]$ a random variable X_i, with

an unknown distribution indicating the number of cards remaining at the end of the game, is introduced. The full sample is then $X = (X_1, X_2, \dots, X_n)$.
We just take the sample mean $m(X) = \frac{1}{n} \sum_{i \in [n]} X_i$ to approximate the average number of cards remaining. For approximating the win rate $w \in [0, 1]$, we simply take $w(X) = \frac{1}{n} |\{i \in [n] \mid X_i = 0\}|$, the number of won games i.e. the number of games with 0 cards remaining, divided by the sample size n.

2.4.2 Choosing a Sample Size

We have to choose a sample size n for our sample $X = (X_1, \dots, X_n)$ such that the strategy can be simulated in a reasonable amount of time and also that the resulting win rate $w(X)$ and the mean of the remaining cards $m(X)$ do not deviate much from their exact value. Since we do not know the exact their exact values, we try to estimate their deviation. Figure 2.2 shows an attempt in quantifying this deviation. The relative standard deviation of the win rates of 10 samples for the perceived capacity strategy introduced later in section 3.1. The standard deviation of the 10 sample win rates with sample size according to the abscissa has been divided by the win rate of a sample using sample size $n = 10\,000\,000$. The aim of relativizing the standard deviation with the win rate of the large sample is to make the quantification of the deviation somewhat transferable to other strategies. The figure shows a relative standard deviation of 1.0% at around a sample size of $n = 160\,000$. The perceived capacity strategy has a win rate of 4.0% for the large sample. This means that with a relative standard deviation of 1%, $[3.96\%, 4.04\%]$ is the region of the win rates within standard deviation measured for the 10 sample win rates. Choosing a sample size larger seems to result in reasonable accuracy.
For a search for the right parameters of a strategy, a sample size of $n = 250\,000$ has been chosen for the most part of this thesis. For strategies where only turns of length $t_{min} = 2$ are played the sample size for the measurements was chosen to be $n = 1\,000\,000$, to be reasonably sure that the approximation is accurate.

2.4.3 Short Turns

As seen in subsection 2.4.2, we should probably take a large sample size n of more than $n = 200\,000$. This leads to a problem, if we want to simulate a strategy over all turns possible. The number of possible turns, especially right at the beginning of the game, can be very large. Early simulation attempts considered the millions of possible turns per state and took multiple minutes to finish simulating a single turn on a very basic strategy. It is therefore infeasible to make a positional evaluation on all possible positions a turn can end up in and still have a meaningful sample size. The turns to be

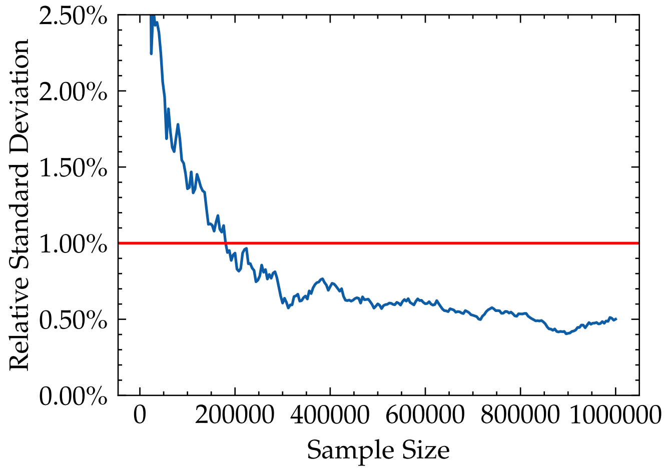

Figure 2.2: Relative standard deviation of 10 sampled win rates using the perceived capacity strategy. The estimated standard deviation has been divided by the win rate 4.0% of the perceived capacity strategy, calculated with $n = 10\,000\,000$ samples, to make the standard deviation relative.

considered have to be reduced significantly, ideally to a couple of hundreds, to be able to simulate a large enough sample.

Most of the time it seems more favorable to split up one long turn into a sequence of short turns because the player obtains more information about the unknown cards of the deck earlier. For instance, if we reach a good position by playing a turn involving 6 cards, we could just split the long turn into three short turns where only $t_{min} = 2$ cards are played each. After each short turn the player draws another 2 cards which could help in finding an even better position. Playing less cards reveals more information about the deck and seems therefore favorable in most cases. Rare cases, close to the end of the game, could be an exception to this. In those cases, playing 3 or more cards could reduce the risk of having a dead draw and not being able to take another full turn. This might therefore increase the chance of winning since the game continues.

With this assumption in mind, we limit the set of positions to consider only those which can be reached by doing a short turn using $t_{min} = 2$ cards. At first glance, doing so does not seem to result in a large drawback, but a short turn leading into the good position might be regarded as having a "worse" value. In other words, there are positions which seem worse if only turns of length 2 are considered but better if longer turns were to be considered. Short turns with $t_{min} = 2$ are examined in chapter 3 in the hopes that the resulting win rate does not suffer too much. Longer turns where only the short turn which led to the position of the long turn are played, are covered in chapter 4.

2.4.4 Simulation Program

For simulating the various strategies presented in this paper a C-program has been written and used. The C-program is only responsible for producing game results in form of how many cards remain at the end of the game or the history of states the games traversed. It has been recompiled for every game configuration and every parameter of a strategy with the help of a Python supporting program. By recompiling the C-program with static parameters the number of dynamic allocations has been dramatically reduced to increase the performance of the simulation. The data analysis of the game results has been done with Python and Jupyter Notebook to visualize the data.

3 Strategies

In the following section, various strategies for valuing game positions of The Game are explored. The strategies covered in this thesis are based on giving a position i.e. state s a value $v(s)$. The basic algorithm for the strategy is then to choose the position with the best value for a given set of positions. See Procedure 1 for the procedure which generates a game result for a random game. A game result is just the number of remaining unplayed cards at the end of the game. With this procedure we get one sample X_i for our estimators introduced in subsection 2.4.1 where the win rate $w(X)$ and the average number of cards remaining $m(X)$ are estimated for the full sample $X = (X_1, \dots, X_n)$ with sample size n.

Procedure 1 Simulate a random game

```
Input: $v : S \to \mathbb{R}$
Output: Sample of cards remaining at the end of the game $X_i$
  $s \leftarrow$ choose uniformly at random from $S_{init}$
  while $T_{t_{min}}(s) \neq \emptyset$ do
    $s \leftarrow \operatorname{argmax}_{s' \in T_{t_{min}}(s)} \{v(s')\}$
    $s \leftarrow$ refill hand from deck for $s$ {see subsection 2.3.5}
  end while
  $s \leftarrow$ play longest possible move sequence for $s$ {see subsection 2.3.6}
  return $|C \setminus L(s)|$
```

An overview of the progression of the key measurements of all presented strategies can be seen in Table 3.2 at the end of this chapter.

3.1 Perceived Capacity

We want to formulate a very simple value function for a state. To do this we observe what a human does intuitively when playing the game. The player wants to keep as many cards playable as possible for future turns. This could be accomplished by finding a turn where the pile values of ascending piles are kept as low as possible and where piles values of descending piles are kept as large as possible. The player wants to keep the remaining distance between the pile card and the last possible card, which can be played on the pile, as large as possible. This distance will be called the perceived capacity of a pile. The value function is then the sum over the perceived capacities of all piles. This means, that in the positional evaluation procedure the state with the overall largest perceived capacity over all piles will be chosen.

3.1.1 Perceived Capacity Function

We call the distance for a pile p of a state s between the pile card $p_c(s,p)$ and the last possible playable card for this pile the perceived capacity $c_p(s,p)$. This capacity is given by

$$c_p : S \times P \to \mathbb{N}_0,\ c_p(s,p) = \begin{cases} c^{\uparrow} - p_c(s,p), & p \in P^{\uparrow} \\ p_c(s,p) - c^{\downarrow}, & p \in P^{\downarrow}. \end{cases}$$

For example, for state s in Figure 2.1 the perceived capacity of pile $p = 2^{\uparrow}$ with $p_c(s,p) = 34$ evaluates to $c_p(s,p) = c^{\uparrow} - p_c(s,p) = 99 - 34 = 65$.
Summing the perceived capacities of a pile over all piles gives us then the value function

$$v(s) = \sum_{p \in P} c_p(s,p),$$

for the Procedure 1. We will call the procedure which uses this value function the perceived capacity strategy.

3.1.2 Results

A sample size of $n = 1\,000\,000$ games has been collected to measure the win rate and the average number of cards remaining at the end of the game. The distribution of this sample can be seen in Figure 3.1. Note that having an odd number of cards remaining is more likely than having an even number of remaining cards for games with at least $h = 8$ cards remaining. If more than $h = 8$ card are remaining the deck still contains cards to draw. After playing a turn and drawing dead, so that no turn can be played any more, there is likely to be a playable card among the drawn cards. This card is played, and the game is lost, which is why the odd results are more prevalent than the even results. Games with less cards remaining do not draw any cards from an empty deck and are therefore distributed more evenly.
As seen in Figure 3.1, 4.0% of the games are won meaning that 4.0% of the games have 0 cards remaining at the end of the game. On average the game is lost with 17.1 cards remaining. By just looking at the pile cards and making a decision without considering the cards in hand or in the deck, the player can achieve a seemingly high chance of

winning the game. We hope to increase this chance by using a more computational complex valuing function instead of just computing the distance between two cards for each pile.

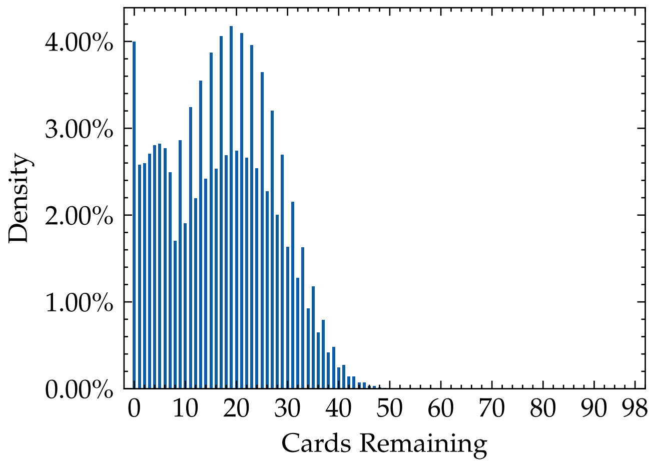

Figure 3.1: Density of the simulation results using the perceived capacity strategy for $n = 1\,000\,000$ games.

3.2 Real Capacity

The perceived capacity of section 3.1 does only consider the pile cards for valuing a position. Over the course of a game, cards are played across ascending and descending piles. This leads to cards being considered in the perceived capacity which are already played. We want to define a new capacity of a pile where only the cards that are relevant, i.e. cards that can still be played onto the pile, are included.

3.2.1 Real Capacity Function

We call the number of still on pile p playable cards for a state s the real capacity $c_r(s,p)$:

$$c_r : S \times P \to \mathbb{N}_0,\ c_r(s,p) = |\{c \in C \setminus L(s) \mid playable(s,c,p)\}|,$$

with

$$playable : S \times C \times P \to \mathbb{B},\ playable(s,c,p) = \begin{cases} p_c(s,p) < c \lor p_c(s,p) - b = c & \text{if } p \in P^{\uparrow} \\ p_c(s,p) > c \lor p_c(s,p) + b = c & \text{if } p \in P^{\downarrow}, \end{cases}$$

which indicates $true$ if card c can be played onto pile p of state s, otherwise $false$. We call it real capacity because it only considers cards which can actually be played on the pile rather than the perceived range of cards which might already have most of its cards played, as in subsection 3.1.1.
The real capacity $c_r(s,p)$ for state s in Figure 2.1 of pile $p = 2^{\uparrow}$ is calculated with $c_r(s,p) = |(\{24\} \cup \{35, 36, \ldots 99\}) \setminus \{80, 88\}| = 64$. Note that the backwards trick card 24 gets included into this capacity as opposed to the perceived capacity strategy. The real capacity captures the number of playable cards better than the perceived capacity with $c_r(s,p) \leq c_p(s,p) + 1$.
The resulting value function is analogous to the one of the perceived capacity strategy with

$$v(s) = \sum_{p \in P} c_r(s,p),$$

where the only difference is the used capacity function.

3.2.2 Results

The win rate decreases compared to the perceived capacity strategy to 3.4% for the same sample size $n = 1\,000\,000$, even though it is a computational more expensive strategy. The mean of cards remaining also gets worse and increases to 17.9 cards. The distribution of the cards remaining at the end of the game for this strategy is almost equivalent to the one of the perceived capacity strategy and is therefore not listed. The distribution of the real capacity strategy is shifted to the right by less than a card with an overall smaller chance reaching lower remaining cards.
Adding the cards which are already laid into the capacity yields a better performance as seen with the perceived capacity strategy. The reason for this might be that the real capacity strategy favors to keep densely packed regions of unplayed cards more than sparsely packed ones. This leads to the problem that the real capacity strategy gives up on potentially essential cards necessary for a win more easily by playing more cards from the sparsely packed regions. Another reason why the perceived capacity outperforms the real capacity might be that the backwards trick is given a constantly large capacity improvement in the perceived capacity strategy. The perceived capacity strategy does on average 11.5 backwards tricks over all games and 18.1 backwards tricks for winning games. Those numbers are higher than those for the real capacity strategy, where 9.7 backwards tricks are used on average over all games and 16.3 backwards tricks for winning games. The perceived capacity strategy uses roughly 2 more backwards tricks per game. Favoring the backwards trick generally seems to be a good idea since it can recover cards necessary for the win.

3.3 Weighting Capacities

The average course of won games using the perceived capacity strategy can be seen in Figure 3.2. The lines of the different piles represent the mean card seen on the pile and the region around them the standard deviation. In both the ascending and descending directions there is a pile which advances far faster than the other pile of the given direction. This is because cards are more likely to be closer to the advanced piles in the beginning and are therefore played right onto them according to the perceived capacity strategy. Until around 80 cards remaining almost no progress has been made on the slow piles whereas the two advanced piles are on average over a third of the way. Seeing the different speeds the piles progress at leads to the idea that it might be a good idea to try to regulate their growth speeds by weighting the capacities of the piles differently.

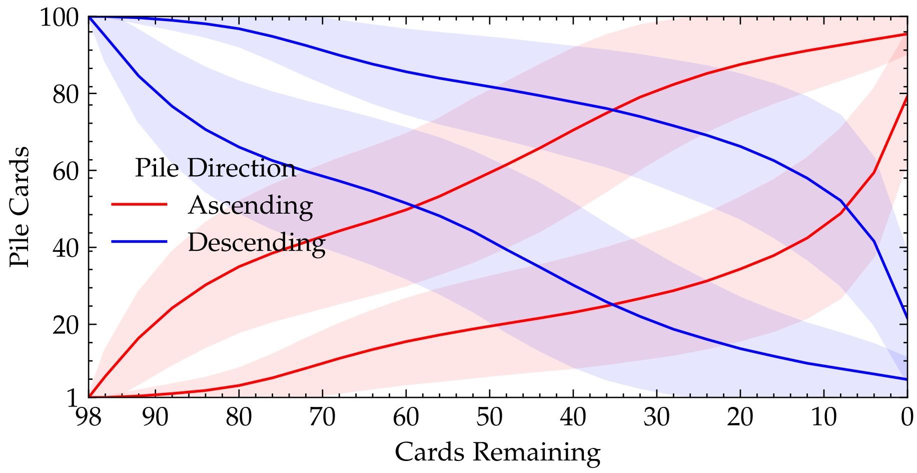

Figure 3.2: Average course of the pile cards for won games using the perceived capacity strategy from section 3.1. The lines represent the mean of the pile cards over all games and the regions around them depict their standard deviation. Before averaging the pile cards of a pile direction, the cards are sorted so that the more advanced pile of one game can be combined with the more advanced piles of other games in a meaningful way.

3.3.1 Weighted Capacity Function

We introduce a weight for each pile with $\omega : P \to \mathbb{R}^+$ and multiply it with the capacity function c, which is either the perceived capacity c_p or the real capacity c_r.

The resulting value function is given by

$$v(s) = \sum_{p \in P} \omega(p) \cdot c(s, p),$$

with either $c = c_p$ or $c = c_r$.

3.3.2 Finding Good Weights

With the value function defined, we need to find a good weighting function $\omega(p)$. Doing a parameter search over multiple parameters $\omega(1^{\uparrow}), \omega(2^{\uparrow}), \omega(1^{\downarrow})$, and $\omega(2^{\downarrow})$ is difficult to simulate since the parameter space is too large. The parameter space has to be reduced to a single dimension. We assume that the weights of the piles in ascending order are symmetrical to the weights of the piles in descending order, meaning that $\omega(1^{\uparrow}) = \omega(1^{\downarrow})$ and $\omega(2^{\uparrow}) = \omega(2^{\downarrow})$. We have to find weights for $\omega(1^{\uparrow}), \omega(2^{\uparrow})$, but only the ratio between the weights is important since the magnitudes of the resulting capacities are relative to each other. Therefore, $\omega(2^{\uparrow})$ can be set to $\omega(2^{\uparrow}) = 1$ and weight $\omega(1^{\uparrow})$ can be linearly searched for.
A weight search using perceived capacity strategy $c = c_p$ for the range $\omega(1^{\uparrow}) \in [0.5, 2.0]$ can be seen in Figure 3.3. There are two extrema at roughly $\omega_1 = 0.675$ and $\omega_2 = 1.475$. With those extrema one can see that only the ratio between $\omega(1^{\uparrow})$ and $\omega(2^{\uparrow})$ is important by taking $\frac{1}{\omega_2} = \frac{1}{1.475} \approx 0.678 \approx \omega_1$. We therefore decide to limit us to a $\omega(1^{\uparrow}) \in (0, 1]$ and set $\hat{\omega}_p = \omega(1^{\uparrow}) = 0.675$ as the best weight for the weighted perceived capacity strategy.
To confirm the assumption that the weights should be symmetrical between the ascending and descending pile directions, we scale the weights for the ascending direction the with factor α. Looking at the results seen in Figure 3.4, clearly confirms our assumption that keeping the weights symmetrical with $\alpha = 1.0$ is best. Nevertheless, this does not exclude that there might still be a better weighting function in the 4-dimensional parameter space of the weights.
The real capacity strategy has a slightly different nature than the perceived capacity strategy. That is why we need to do a new weight search for the real capacity. Figure 3.5 shows the effects of the weight $\omega(1^{\uparrow})$ on the real capacity $c = c_r$. As expected the extrema at roughly $\hat{\omega}_r = \omega(1^{\uparrow}) = 0.6$ for the weighted real capacity strategy is slightly different from the weight $\hat{\omega}_p = 0.675$ for the weighted perceived capacity strategy. This means that the real capacity strategy is weighted more aggressively.

3.3.3 Results

As already seen in Figure 3.3 and Figure 3.5, introducing weights can increases the win rate for both the perceived capacity strategy and the real capacity strategy.

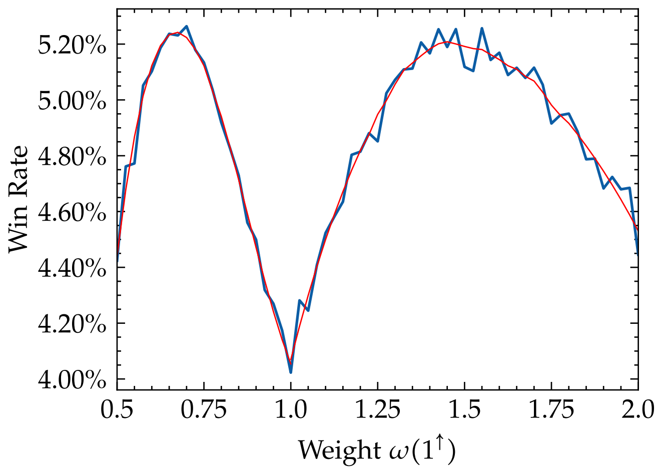

Figure 3.3: Weight search for the weighted perceived capacity strategy $c = c_p$, with weights $\omega(1^{\uparrow}) = \omega(1^{\downarrow})$ and $\omega(2^{\uparrow}) = 1 = \omega(2^{\downarrow})$. The blue line represents the samples collected with sample size $n = 250\,000$ and step 0.025 between the weight parameter $\omega(1^{\uparrow})$. The red line denotes a smoothing of the samples applying the Savitzky and Golay's filter [Sch+11] for ranges $\omega(1^{\uparrow}) \in [0, 1]$ and $\omega(1^{\uparrow}) \in [1, 2]$. This filter will also be used in all other parameter searches so that the location of an extrema can be better marked.

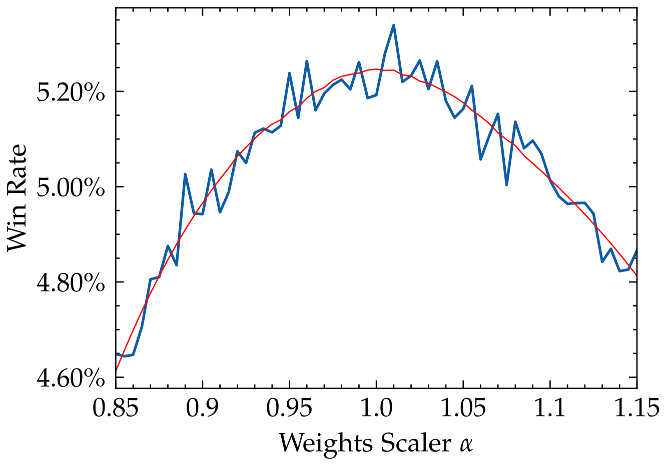

Figure 3.4: Asymmetric scaling of weights for the weighted perceived capacity strategy $c = c_p$, with weights $\omega(1^{\uparrow}) = \alpha \cdot 0.675, \omega(2^{\uparrow}) = \alpha$ and $\omega(1^{\downarrow}) = 0.675, \omega(2^{\downarrow}) = 1$. A sample size of $n = 250\,000$ has been used with an α step of 0.005.

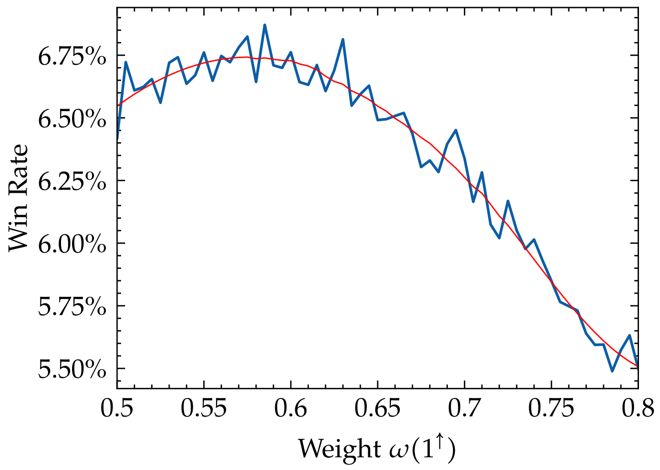

Figure 3.5: Weight search for the weighted real capacity strategy $c = c_r$, with weights $\omega(1^{\uparrow}) = \omega(1^{\downarrow})$ and $\omega(2^{\uparrow}) = 1 = \omega(2^{\downarrow})$. Sample size $n = 250\,000$ with $\omega(1^{\uparrow})$ step 0.005.

The weighted perceived capacity strategy with weights $\omega(1^{\uparrow}) = 0.675 = \omega(1^{\downarrow}), \omega(2^{\uparrow}) = 1 = \omega(2^{\downarrow})$, has a win rate of 5.2% with on average 15.4 cards remaining at the end of the game. By weighting the perceived pile capacities with good weights, a multiplicative increase of 30% can be achieved for the win rate. Figure 3.6 shows how the behavior of the average pile cards for won games changes compared to the unweighted case. As expected, the speeds in which the different piles grow changes. The growth of the more advanced piles, which are most of the time piles $1^{\uparrow}$ and $1^{\downarrow}$ weighted with $\omega(1^{\uparrow}) = \omega(1^{\downarrow}) = 0.675$, is not plateauing as much as the unweighted advanced piles at around 70 to 60 cards remaining. The slower piles, most of the time the more heavily weighted piles $2^{\uparrow}$ and $2^{\downarrow}$ weighted with $\omega(2^{\uparrow}) = \omega(2^{\downarrow}) = 1$, are growing slower than in the unweighted case leaving on average more cards playable for later on. The pile cards of the four piles seem to be on average further away from each other, covering a wider range of cards with a closer distance. The risk of larger steps might therefore be reduced, leading to more cards played overall and a higher win rate.
The weighting has an even bigger impact on the real capacity strategy $c = c_r$. In that strategy a win rate of 6.8% with on average 14.9 cards remaining can be reached. This means that the weighted real capacity strategy is 30.8% better than weighted perceived strategy and 100% better than the unweighted real capacity strategy. The real capacity strategy was worse than the perceived capacity strategy, but after weighting the capacities, the real capacity strategy it is even better than the weighted perceived capacity strategy. When comparing the average course of won games between the two

weighted capacities see Figure 3.7, the advanced piles grow a little faster from 60 to 40 cards remaining and the slow piles develop faster where less than 25 cards remain. This figure does not really indicate why the weighted real capacity is superior to the weighted perceived capacity. Leading to the question why weighting the real capacity has such a large impact on the win rate. The reason for this increase might be that the problem of the unweighted real capacity, where densely packed areas of unplayed cards are more favored than sparsely packed areas, has been greatly reduced. The two advanced piles increase faster through weighting, this might leave more evenly distributed smaller gaps of cards behind. There are less densely packed regions of cards leading to a more informed decision, instead of just stepping blindly over cards. Now that the problem is reduced the strength of the expressiveness of the real capacity shows to be advantageous by counting only cards that can still be played against the capacity. Both, reducing the problem of the real capacity strategy, and its expressiveness, seem to be the reason over the sudden superiority over the perceived capacity.

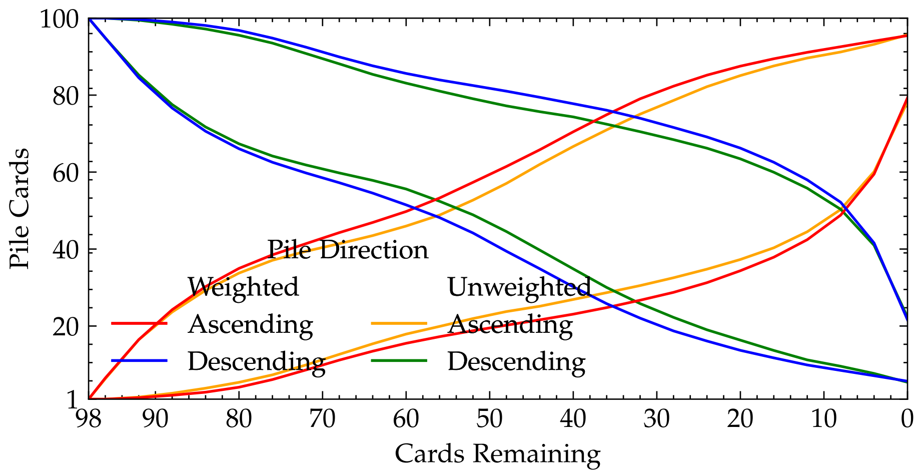

Figure 3.6: Comparison of average course of the pile cards for won games between the perceived capacity strategy in the weighted case and in the unweighted case. The weights have been chosen according to subsection 3.3.2. Compared to Figure 3.2, the standard deviation regions have been omitted in this figure since they are roughly the same and just clutter.

3.4 Penalize Playability

The real capacity in the weighted case in contrast to the unweighted case illustrate the importance of making sure that some cards should not be neglected by skipping over

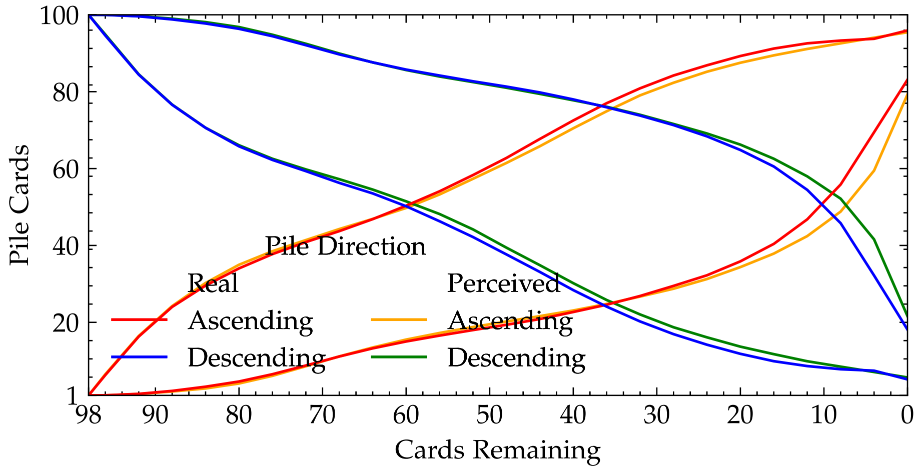

Figure 3.7: Comparison of average course of the pile cards for won games between the weighted real capacity strategy and the weighted perceived capacity strategy. Similar to Figure 3.6.

them. This leads to the idea, that neglecting cards should come at a price, or in other words, a penalty. We want to make sure that every card should be kept playable on at least one pile in order to keep the game winnable.

3.4.1 Penalty Function

We introduce a function $p_\# : S \times C \to \{0, 1, \ldots, p_{\#\uparrow} + p_{\#\downarrow}\}$, which counts the number of piles a cards c can be played onto for a state s. This function is given by

$$p_\#(s,c) = |\{p \in P \mid playable(s,c,p)\}|,$$

with $playable(s,c,p)$ as in subsection 3.2.1.
This counting function is then used in a penalty function $f : \{0, 1, \ldots, p_\uparrow + p_\downarrow\} \to \mathbb{R}_0^+$, where $f(p_\#(s,c))$ gives a penalty for the playability of card c. Penalizing all cards still in play yields the value function

$$v(s) = -\sum_{c \in C \setminus L(s)} f(p_\#(s',c)),$$

for our penalize playability strategy. Notice that the value function is negative so that the Procedure 1 minimizes the sum of penalties over all cards still in play.

3.4.2 Good Penalty Function

We want to find a fitting penalty function f for our value function. We start with the assumption, that $f(0) = 1$ since penalties are relative to each other and we definitively want to penalize cards that cannot be played any more. It is reasonable to assume that the penalty function should be monotonically decreasing because having a card playable on more piles should have a smaller penalty than a card playable on fewer piles. Therefore, a good candidate for this penalty function seems to be $f(x) = e^{-\alpha x}$. Using this function, we search for a fitting parameter α yielding the highest win rate. See Figure 3.8 for a parameter search for $\alpha \in [0.5, 3.5]$. The best parameter can be found at roughly $\hat{\alpha} = 1.5$. Notice that a small α penalizes cards playable on more piles too much, leading to a lower win rate. An α larger than $\hat{\alpha} = 1.5$ seems to only really penalize cards which are playable on very few piles. The penalty mostly lies at $f(0)$ but as seen with $\hat{\alpha} = 1.5$ penalizing $f(x)$ with $x > 0$ higher seems to more optimal.

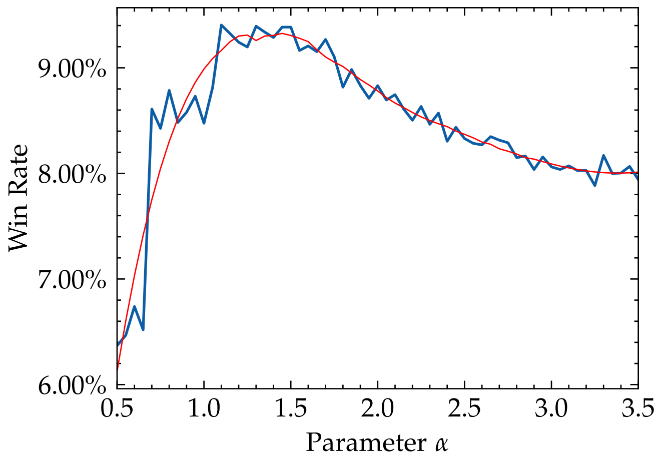

Figure 3.8: Parameter search for α of the penalty function $f(x) = e^{-\alpha x}$ of the penalize playability strategy. Sample size $n = 250\,000$ with α step 0.05.

3.4.3 Results

Through penalizing playability of cards another increase in the win rate can be realized. Using $f(x) = e^{-1.5x}$ as the penalty function, the win rate reaches 9.4% with 13.2 cards remaining on average. This is another 38% multiplicative increase in the chance of winning in comparison to the weighted real capacity.
The comparison of the average course of the game for won games between the penalize playability strategy and weighted real capacity strategy is noticeably different, see

Figure 3.9. The penalize playability strategy seems to allow faster growth for the advanced piles at around 80 cards remaining, and more steady growth of the slow piles across the course of game. Because most cards are still playable on other piles and keeping the penalties balanced across the cards, the piles seem to grow earlier overall. When faced with a bad decision the penalized playability strategy can give up on a pile for a single card more easily than the weighted real capacity strategy, which tries to keep as many cards playable as possible over all piles. This seems to be the biggest strength of using penalties.

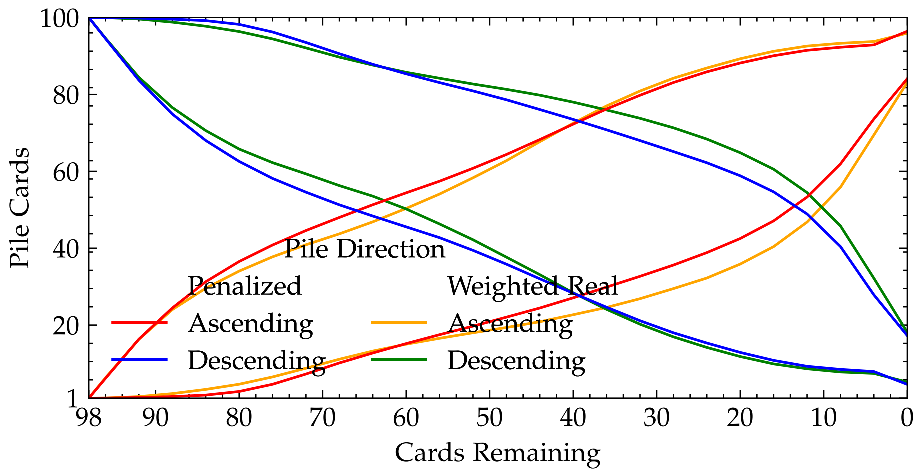

Figure 3.9: Average course of won games for the penalized playability strategy compared to the weighted real capacity strategy using a good penalty function and good weights.

3.5 Penalty with Recoverability

A strategy where the playability of cards is penalized has been introduced in the previous section 3.4. This value function did not directly consider the case that a card can be made playable again by doing backwards tricks. If a card might be recoverable in the future we want to reduce the penalty in proportion to the likelihood that the card can be recovered.

3.5.1 Simplifying the Penalty

In the section 3.4 an exponential function is used which takes the number of piles a card can be played onto as penalty function. We want to have more flexibility in the penalty function when we introduce a recoverability term in it. We establish a simplified penalty function $\pi : S \times P \times C \to [1, \ldots \infty)$, which expresses a penalty for a card c on pile p of state s. By taking the exponential function as example, the overall penalty for a card c is the product of the penalty $\pi(s, p, c)$ over all piles in P, $\prod_{p \in P} \pi(s, p, c)$. Analogous to the penalized playability, using this simplified penalty yields

$$v(s) = - \sum_{c \in C \setminus L(s)} \prod_{p \in P} \pi(s, p, c),$$

as value function.
Using then the simplified penalty function

$$\pi(s, c, p) = \begin{cases} 1 & \text{if } \mathit{playable}(s, c, p) \\ \alpha & \text{otw.} \end{cases} \tag{3.1}$$

with the right constant $\alpha \in [1, \infty)$ should be roughly equivalent to the previous penalize playability strategy. In Figure 3.11 a penalty search for the constant $\alpha \in [2, 5]$ can be seen. Choosing an α too small penalizes bad playability too little and choosing it too large seems to lead to bad choices. The strategy should probably sometimes concede on highly penalized cards rather than keeping them. The highest win rate can be achieved for roughly $\hat{\alpha} = 3.5$. Using $\hat{\alpha}$ realizes a win rate of 9.3% with 13.2 cards remaining on average at the end of the game. These numbers are equivalent to the numbers of the previous penalized playability strategy where the exponential function was used. The advantage of the new penalty is that we can modify the penalty of a card for a single pile more easily.

3.5.2 Recovery Using Distance

We want to soften the penalty in Equation 3.1 for unplayable cards by considering the distance of the card to the pile card. It seems to be exponentially less likely to recover a card the further it is away from the pile card since it requires the use of more consecutive backwards tricks. A new penalty function is therefore given by

$$\pi(s, c, p) = \begin{cases} 1 & \text{if } \mathit{playable}(s, c, p) \\ \alpha - \beta \cdot e^{\gamma \cdot (|p_c(s,p) - c| - 1)} & \text{otw.,} \end{cases} \tag{3.2}$$

with $\beta \in [0, \alpha - 1]$ the magnitude of the penalty reduction and $\gamma \in (0, \infty)$ the falloff weighting the distance between the card and the pile card.

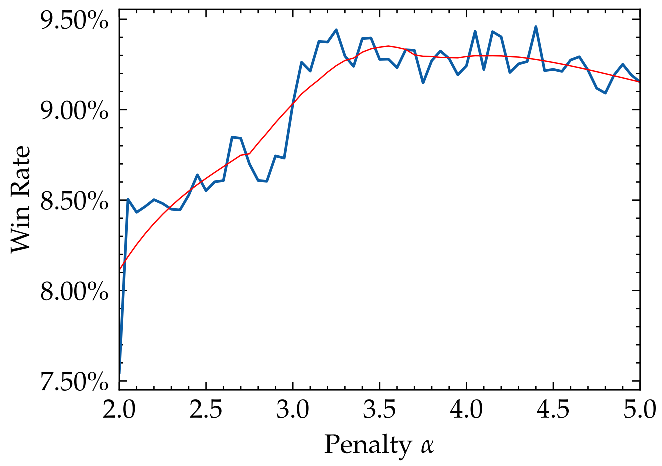

Figure 3.10: Parameter search for constant α of Equation 3.1 for the simplified penalty strategy. Sample size $n = 250\,000$ with α step 0.05. The slight decrease from $\alpha \in [3.5, 5.0]$ continues to decrease for larger penalties α.

Choice of Magnitude β and Falloff γ

We leave for the constant penalty $\alpha = 3.5$ and try to find fitting β and γ. Finding optimal parameters in a two-dimensional parameter is difficult to do. The magnitude β has to be fixed before searching for the falloff γ. Without the right β the effect of the γ will be small. Only a few magnitudes have been checked and $\beta = 1$ seems to work fair enough. It reduces the penalty by not too much and not too little but there are certainly better choices. Searching for a γ, see Figure 3.11, reveals that $\hat{\gamma} = 0.03$ has the highest win rate. Searching now for a new β or α with this falloff did not improve the chance of winning by too much and has therefore been abandoned.

Results

As already seen in the parameter search of γ in Figure 3.11 the win rate increased with using the recovery using distance strategy. Using $\alpha = 3.5$, $\beta = 1$, and $\gamma = 0.03$ a win rate of 12.7% has been achieved with on average 11.2 cards remaining. This is another multiplicative increase of 35% compared to the win rate of the strategy using a penalty without the recovery term.
This shows that the penalty for unplayable cards should not always be constantly large and has to be softened the more likely a card is to be recovered on a pile. Having the recovery term in the penalty allows the strategy to make a better decision on which cards to abandon if there is no other choice. Cards which can be recovered with a better

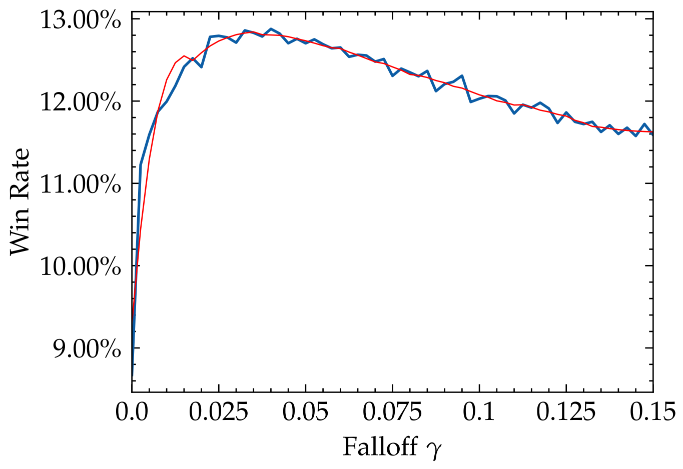

Figure 3.11: Falloff γ search for the penalized playability with recovery using distance strategy of subsection 3.5.2. The magnitude $\beta = 1$ with penalty $\alpha = 3.5$ has been chosen for this search. A sample size of $n = 250\,000$ with γ step 0.0025 has been used.

chance are quicker abandoned than those who have a have a lesser chance of recovery. A possible way to improve this strategy is by taking a similar approach than for the perceived and real capacities. Instead of weighting the perceived distance one could weight the number of cards, which have a backwards trick partner between the current unplayable card c and the pile card in the respective order. This approach has not been tried in this thesis since simulating it seems too computationally expensive. There are numerous positions that are evaluated where potentially a lot of cards are unplayable leading to a lot of iterations per unplayable card.

3.5.3 Recoverability Estimation Using Single Bridges

We have seen that softening the penalty function of Equation 3.1 for unplayable cards, which are closer to the pile card improves the chance of winning the game. In the recovery using distance a very rough estimation has been used to achieve this. We want to quantify the recoverability of a card more precisely.
We want to estimate the likelihood of recoverability $\rho(s,c,p) \in [0,1]$ for a card c onto pile p of state s and use it to soften the penalty in proportion to it. The resulting penalty function is

$$\pi(s,c,p) = \begin{cases} 1 & \text{if } \mathit{playable}(s,c,p) \\ 1 + (\alpha - 1) \cdot (1 - \rho(s,c,p)) & \text{otw.,} \end{cases} \tag{3.3}$$

with α chosen as previously for Equation 3.1. $\rho(s,c,p)$ should be the probability of recovering the card c on pile p, meaning that if a card can always be recovered $\rho(s,c,p) = 1$ no penalty should be given $\pi(s,c,p) = 1$, and when it is impossible to recover the card $\rho(s,c,p) = 0$ the full penalty $\pi(s,c,p) = \alpha$ should be carried. Ideally, $\rho(s,c,p)$ would be the exact probability of recovery. This seems impossible to calculate since it involves the unknown cards to draw, the multiple ways the card can be recovered over the rest of the game, and the cards the strategy will actually play. We therefore need a way to approximate this probability.

Single Bridges

The single bridges estimation of $\rho(s,c,p)$ assumes that cards can only be recovered using a single backwards trick. If a backwards trick can be made at pile p to recover card c with a hand card, the card is recoverable with the probability $\rho(s,c,p) = 1$. If the backwards trick can not be done using the cards at hand, we calculate the probability of drawing cards that are needed for doing this backwards trick next turn and use it for our estimation. This assumes that the card has to be immediately recovered next turn, which is not always the case since the player can play onto other piles in the meantime and wait for the draw.
We want to recover a card c in range $p_c(s,p) - b < c < p_c(s,p)$ for ascending piles and in range $p_c(s,p) < c < p_c(s,p) + b$ for descending piles, using cards from the deck. Let B be the set of the existing single bridges which can be used to recover this card. A bridge is a pair of unplayed cards used in a backwards trick. The set B is given by

$$B = \begin{cases} \{(k, k+b) \mid k \in \{p_c - b, p_c - b + 1, \dots, c\} \wedge k, k+b \in C \setminus L(s)\} & \text{if } p \in P^{\uparrow}, p_c = p_c(s,p) \\ \{(k, k-b) \mid k \in \{c, c+1, \dots, p_c + b\} \wedge k, k-b \in C \setminus L(s)\} & \text{if } p \in P^{\downarrow}, p_c = p_c(s,p), \end{cases}$$

which is a set of tuples containing both endpoint cards of the bridges. Count the number of bridges s_b where only a single card is missing from the hand

$$s_b = |\{(e_1, e_2) \in B \mid e_1 \in D(s) \oplus e_2 \in D(s)\}|,$$

and count the number of bridges d_b where a double draw of the same bridge is necessary to recover the card

$$d_b = |\{(e_1, e_2) \in B \mid e_1 \in D(s) \wedge e_2 \in D(s)\}|.$$

After the turn $t_{min} = 2$ cards are drawn from deck $D = D(s)$. We want to calculate the probability, that we draw either one of the missing s_b single endpoints of the bridges, or one of the d_b pairs where both of the endpoints of a bridge are missing. We establish a probability tree for finding this probability as seen in Figure 3.12. The probability to draw a bridge and our estimation for the recoverability $\rho(s,c,p)$ is therefore

$$\rho(s,c,p) = P(\text{drawn}) = \frac{s_b}{|D|} + \frac{d_b}{|D|} \cdot \frac{s_b + 1}{|D| - 1} + \frac{|D| - s_b - d_b}{|D|} \cdot \frac{s_b}{|D| - 1}.$$

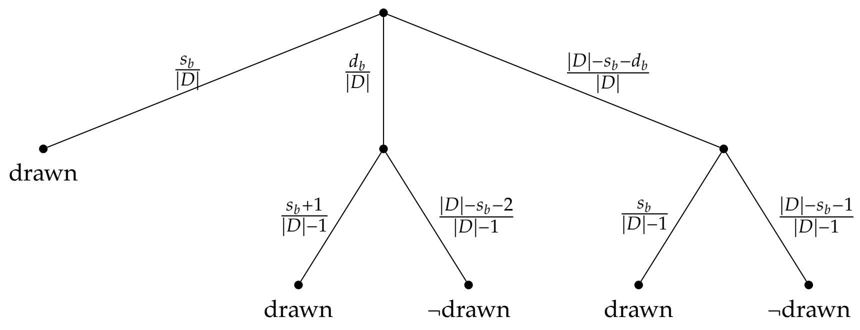

Figure 3.12: Probability tree used for the recoverability estimation $\rho(s,c,p)$, with s_b, d_b as in section 3.5.3 and $D = D(s)$ of state s.

Results of Single Bridges

As expected, the win rate of using the single bridges increases compared to the strategy using only the penalty. A win rate of 10.8% is achieved with 12.8 cards remaining. Using the single bridge estimation is worse than the recovery using distance penalty strategy where 12.7% are won. The reason for this might be, that the single bridges strategy softens only the penalty of unplayable cards which are directly reachable with a single backwards trick. And even for those, the estimation for the probability of recoverability $\rho(s,c,p)$ might be too conservative. The probability assumes, that the card has to be recovered immediately in the next turn. Most of the time, the player can play other cards from the hand onto other piles leading to more unconsidered draws where a bridge can be found. The estimation of $\rho(s,c,p)$ has to consider this in order to reach a higher win rate.

Single Bridges with Amplified Recoverability

The estimation $\rho(s,c,p)$ is too modest and has to be amplified in some way to be a more precise estimation. A way to magnify the recoverability is to use an exponent similar to a square root. The resulting penalty function is then

$$\pi(s,c,p) = \begin{cases} 1 & \text{if } \mathit{playable}(s,c,p) \\ 1 + (\alpha - 1) \cdot (1 - \rho(s,c,p)^{\lambda}) & \text{otw.,} \end{cases}$$

with $\lambda \in (0,1]$ and $\rho(s,c,p) \in [0,1]$ the single bridges recoverability estimation.
A search for the exponent λ can be seen in Figure 3.13. Note the increasing win rate with smaller exponent reaching the peak at $\hat{\lambda} = 0.2$. Even smaller exponents amplify the recoverability estimate by too much leading to a complete removal of the penalty when there is the slightest chance of a bridge.

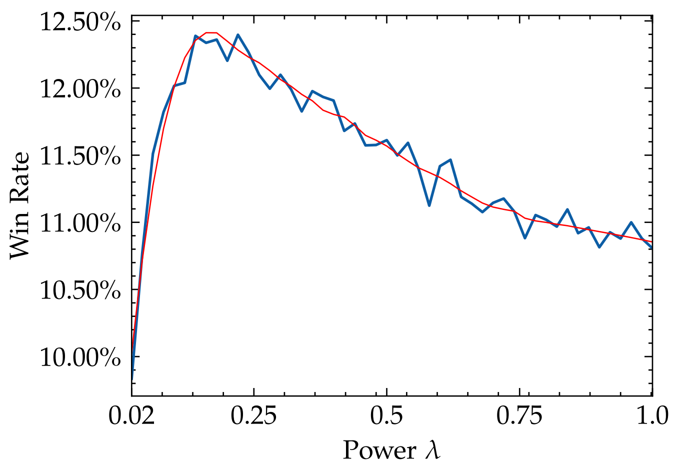

Figure 3.13: Search for exponent $\lambda \in [0.02, 1]$ of the single bridges strategy where the probability of recoverability estimation $\rho(s,c,p)$ has been amplified. A reduced sample size of $n = 100\,000$ with λ step 0.025 has been used because counting the bridges in this strategy took longer to compute.

3.5.4 Results of Single Bridges Amplified

Amplifying the recoverability probability estimation with $\lambda = 0.2$ yield a 13% increase in the win rate compared to the unamplified case. The win rate reached with this strategy is 12.2% with 13.0 cards remaining. This strategy still performs multiplicative 4% worse than penalty strategy which uses the recovery using distance term. Nevertheless, the amplified single bridges strategy is close to beating it by only softening the penalties of unplayable cards that are in reach within one backwards trick. This strategy shows a lot of potential for improvement.
An obvious possible improvement is to consider double bridges where unplayable cards are recovered using two backwards tricks. Coming up for a probability of recovery estimation for this seems much tougher since it involves much more combinations of recovering a card over more than one turn. Another potentially viable way to improve the win rate is to change how the recoverability is amplified. At the moment, it is

amplified by a constant exponent over the course of the game. The amplifying exponent should probably dependent on how many cards remain to be played, possibly having a larger amplification in the first half of the game than the second half. The amplification could also depend on the pile and its pile card, magnifying the exponent if the growth speed of pile is slow. Attempts have been made to find a good dependency, but one which increases the chance of winning has not been found.

3.5.5 Single Bridges with Recovery Using Distance

Another obvious improvement is to just fallback on the penalty function which uses the distance to soften the penalty for unplayable cards which need to be recovered using two or more backwards tricks. The resulting penalty function is therefore given by

$$\pi(s,c,p) = \begin{cases} 1 & \text{if } playable(s,c,p) \\ \alpha - \beta \cdot e^{\gamma \cdot (|p_c(s,p) - c| - 1)} & \text{if } \neg playable(s,c,p) \wedge |p_c(s,p) - c| > b \\ 1 + (\alpha - 1) \cdot (1 - \rho(s,c,p)^{\lambda}) & \text{otw.} \end{cases}$$

3.5.6 Results of Single Bridges with Recovery Using Distance

The same parameters penalty $\alpha = 3.5$, magnitude $\beta = 1$, falloff $\gamma = 0.03$, and power $\lambda = 0.2$ have yielded the best win rate in the conducted searches to find better parameters. Changing any of them does not seem to increase the chance of winning. This strategy achieves the highest win rate of all covered strategies with a 13.1% win rate and 12.4 cards remaining. Notice that this strategy seems to take more risks than the previous best strategy, which used only the recovery term involving the distance. It has a higher win rate but a worse mean when compared. This shows, that there is a strategy where win rate does not directly correlate to the average number of cards remaining at the end of the game.

Strategy Name	**Win Rate**	**Cards Remaining**	**Backwards Trick Usage of Winners**
Perceived Capacity	4.0%	17.1	18.1
Real Capacity	3.4%	17.9	16.3
Weighted Perceived Capacity	5.2%	15.4	18.0
Weighted Real Capacity	6.8%	14.9	15.7
Penalize Playability ($e^{-1.5x}$)	9.4%	13.2	15.1
Recovery Using Distance	12.7%	11.2	15.9
Single Bridges	10.8%	12.8	15.9
Single Bridges Amplified	12.2%	13.0	16.6
Single Bridges with Recovery Distance	13.1%	12.4	16.7

Table 3.2: Summary of all strategies using their best parameters found.

4 Deeper Search Depth

So far only turns of length $t_{min} = 2$ have been evaluated in the positional evaluation. As seen in the distribution Figure 4.1 where the backwards trick has been removed by setting $b = 0$ for the perceived capacity strategy, making good use of the backwards trick is necessary for winning the game. In a sample of $n = 1\,000\,000$ games only 1 game lead to a win. We want to account for longer turns, which might collect more value from the backwards trick. With a deeper search the strategy might be able to squeeze in a card before doing a backwards trick.

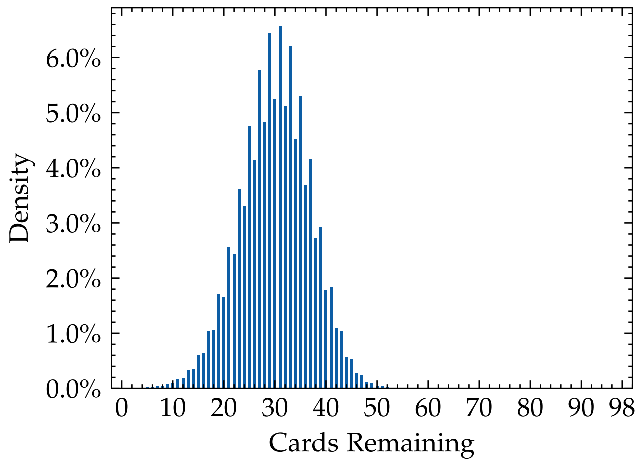

Figure 4.1: Density of the perceived capacity strategy with no backwards tricks enabled $b = 0$. Sample size of $n = 1\,000\,000$ games.

4.1 Extending the Search Depth

We want to evaluate the value of positions which use more than $t_{min} = 2$ cards in a turn. As already discussed in subsection 2.4.3, splitting a long turn and playing only the minimum required number of cards per turn t_{min} seems generally better than playing the long turn immediately. Playing short turns reveals information about the cards in the deck earlier and might therefore change the position, which holds the highest

value.
This means that the value of a position at depth $t_{min} = 2$ depends on the value of the positions, where more cards have been played. Therefore, come up with a new value function with extended depth

$$v_d(s) = \max_{\substack{s' \in \bigcup T_i(s) \\ i \in \{0,1,\ldots,d\}}} \{v(s')\},$$

with $d \in \mathbb{N}$ and $v(s)$ as a value function of a strategy. With this value function $v_d(s)$ position s will be given the highest value $v(s')$ that can be found for a state s' by playing $i \in \{0, 1, \ldots, d\}$ cards onto position s.

4.2 Results

As already mentioned, the number of possible turns grows exponentially with the length of the turn. Experiments yielded, that there are more than 10 times as many positions for turns of length 3 than length 2 at the beginning of the game. Roughly the same factor occurs with the jump from 3 to 4 cards played. The sample size had therefore be reduced by a ten fold for samples using extra depth $d = 1$.
By extending the win rate with $d = 1$, the win rate of the strategies which involve a form of capacity seems to yield only a negligible increase in the win rate, as seen in Table 4.2. This is because the capacities do not specifically gain any value by playing more cards. The slight increase might be due to random chance of picking a position, which uses more cards when doing a backwards trick, while maintaining the same capacity value. The strategies using a form of penalty for card playability on the other hand perform much better. The win rates using $d = 1$ are significantly better and see multiplicative improvements ranging from 47% to 85% compared to the win rates using $d = 0$. The penalty functions can reduce their penalty by playing a card. When a card gets played, it improves the penalty by at least 1 and even more for badly playable cards. This means, that a penalty function wants to play cards rather than keeping them as opposed to the capacity functions. This seems to be the better strategy of playing The Game, rewarding the playing of cards and preventing other card from becoming unplayable.
An experiment with $d = 2$ for the best penalty strategy achieves another improvement. In a small sample of size $n = 10\,000$, 29.7% of the games have been won with 7.1 cards remaining at the end of the game. The sample size used for $d = 2$ is much smaller and the actual win rate might deviate because of the small sample size. To give a sense of how much deviation is expected, 100 win rates of samples with size $n = 10\,000$ have been collected for the perceived capacity strategy using $d = 0$. The standard deviation of these win rates divided by the win rate measured for sample size

Strategy Name	Win Rate $d = 0$	Win Rate $d = 1$	Multiplicative Increase
Perceived Capacity	4.0%	4.0%	0.6%
Real Capacity	3.4%	3.4%	0.4%
Weighted Perceived Capacity	5.2%	5.3%	2.3%
Weighted Real Capacity	6.8%	6.8%	1.2%
Penalize Playability ($e^{-1.5x}$)	9.4%	16.2%	73%
Recovery Using Distance	12.7%	18.8%	47%
Single Bridges	10.8%	18.2%	67%
Single Bridges Amplified	12.2%	22.6%	85%
Single Bridges with Recovery Distance	13.1%	23.4%	80%

Table 4.2: Summary over all presented strategies with their best parameter choice for $d = 0$ and with extra search depth $d = 1$. A sample size of $n = 1\,000\,000$ has been used to estimate the win rate with $d = 0$, and a size of $n = 100\,000$ has been used for $d = 1$. The multiplicative increase has been calculated with the exact measured values instead of the rounded ones.

$n = 10\,000\,000$ gives a standard divination of 4.7% relative to the win rate. Assuming that this relative standard divination is transferable to this penalty strategy, which has a more than 7 times higher chance of winning, we get the additive standard deviation of $29.7\% \cdot 4.7\% \approx 1.4\%$. With this result, it is safe to say that increasing the extra depth d yields an overall higher chance of winning for a penalty strategy. The strategy might be able to better plan ahead with the extra search depth. Every played card can reduce the penalty. This penalty strategy can therefore gain value by squeezing in cards before a backwards trick.
An even higher win rate could be achieved by searching for more fitting parameters. When using the extended search depth value function, the value function of the strategy might need adjustments. Searching for better parameters is much harder to do with $d > 0$ since the simulation runs much slower. One could maybe reduce the number of possible positions to consider by not naively checking every position ranging to depth $t_{min} + d$. Some positions at depth t_{min} are clearly bad choices and lead to even worse positions with an extra depth d. A similar approach to alpha-beta pruning as used for chess [KM75] could be taken, where positions, that are unquestionably worse are just skipped to speed up the search for a good position. This technique might be somewhat viable for the penalty strategies but has not been attempted in this paper. Another way might be a possible algorithm, which reduces the positions to consider

in the extra depth value function $v_d(s)$ to s and the ones, which end in a backwards trick. Only those seem to be relevant for finding a position which improves the value since backwards tricks recover cards. A search for this algorithm has been unsuccessful, since it was not clear how to deal with the vast number of possible turns, if it is allowed that cards can be played on other piles while still ending the turn in a backwards trick.

5 Conclusion

A positional evaluation has been used as basis for finding a strategy for The Game. After observing that there are too many positions to evaluate for having a large enough simulated sample, the positions to check have been reduced to a bare minimum. Only to those, which can be reached by playing two cards. We came up with two main groups of strategies of valuing a position in The Game. The capacity value functions, and the penalizing value functions. With the capacity functions, we started off by looking at a perceived distance between the pile card and the last playable card of the pile and achieved our first measured win rate of 4.0%. We tried to be more precise with the capacity and introduced a real capacity function, which led to a decrease in the chance of winning to 3.4%. We observed, that the piles' grow rates are different from each other and made an effort to change them by weighting the capacities. Doing so yielded a win rate increase to 5.2% for the perceived capacity and to 6.8% for the real capacity. It was guessed, that this sudden increase for the real capacity strategy had to do with less skipping over important cards by distributing cards, through the change in growth speed of the piles, more advantageously. Noticing that some cards are vital for the win and should not be skipped, we introduced a penalty strategy penalizing playability of cards to prevent cards from becoming unplayable. After finding a good penalty function we achieve a win rate of 9.4%. The idea of a penalty displayed great potential and was therefore simplified to introduce a recover term to soften the penalty for cases, where the cards can be more likely recovered using the backwards trick. Our first penalty function using a recovery term was the recovery using distance strategy. Similar to the concept of the perceived capacity strategy, the distance between card and pile card has been used in a negative exponential function to reduce the penalty for unplayable cards closer to the pile cards. Using this, a win rate of 12.7% was reached. Trying to make the recovery term more precise, a single bridges strategy was established to soften the penalty in proportion to a likelihood estimating the probability, that the card can be recovered using a single backwards trick next turn. The resulting win rate was only 10.8%, leading to the observation that the probability of recoverability estimate had to be amplified. After magnifying it with an exponent, a win rate of 12.2% was reached, slightly smaller than the previous best win rate. After combining the amplified single bridges strategy with the strategy using the distance for the recovery, a new best win rate of 13.1% has been achieved. Subsequently, the search depth for

positions to evaluate has been extended for all strategies by 1 card. Extending the depth did significantly affect the performance of strategies using a form of penalty, reaching a win rate of around 23.4% for best penalty strategy with extra depth 1. Finally, a small sample of 10 000 games has been collected for our best strategy with extra depth 2. In this sample, 29.7% of the games were won with an average 7.1 cards remaining at the end of the game. The Game's manual says, that a result below 10 cards remaining at the end of the game is an excellent result. Out of the 10 000 games, 6 988 games reached this point. This means, that a game result using our single bridges combined with the recovery distance strategy with extended search depth of 2, has on average a satisfactory outcome.
The found strategy performed well but there is potential for future improvements. Extending the search depth was limited because of the exponentially growing number of possible turns. Future work would entail finding an algorithm which only selects relevant positions at maximal search depth. Furthermore, the parameters of the best strategy are suboptimally chosen and could be improved by making a more thorough search. Finally, coming up with a more elaborate recovery term could yield another improvement for the win rate in future work.

List of Figures

List of Tables

Bibliography

[Ben12] S. Benndorf. *Quixx*. 2012. URL: `boardgamegeek.com/boardgame/131260/qwixx` (visited on 09/12/2019).

[Ben15] S. Benndorf. *The Game*. 2015. URL: `boardgamegeek.com/boardgame/173090/game` (visited on 09/12/2019).

[BL11] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[GGS13] V. Gabillon, M. Ghavamzadeh, and B. Scherrer. "Approximate dynamic programming finally performs well in the game of Tetris." In: *Advances in neural information processing systems*. 2013, pp. 1754–1762.

[Hay+09] S. S. Haykin et al. *Neural networks and learning machines/Simon Haykin*. New York: Prentice Hall, 2009.

[KM75] D. E. Knuth and R. W. Moore. "An analysis of alpha-beta pruning." In: *Artificial intelligence* 6.4 (1975), pp. 293–326.

[KR] J. S. Ku and M. Rudoy. "Complexity of Benndorf's "The Game"." In: ().

[Sch+11] R. W. Schafer et al. "What is a Savitzky-Golay filter." In: *IEEE Signal processing magazine* 28.4 (2011), pp. 111–117.

[Sha50] C. E. Shannon. "XXII. Programming a computer for playing chess." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314 (1950), pp. 256–275.

[Smi07] D. K. Smith. "Dynamic programming and board games: A survey." In: *European Journal of Operational Research* 176.3 (2007), pp. 1299–1318.