Elicitation of Strategies in Four Variants of a Round-robin Tournament: The case of Goofspiel (Supplementary File)

Moshe Dror and Graham Kendall, *Senior Member, IEEE* and Amnon Rapoport.

I. Preamble

This is a supplementary file to the paper of the same name/authors as appears above.

Appendix A provides more details of the strategies that are briefly presented in the paper. The descriptions are those handed in by the students, in accordance with their requirements of their course.

The supplement also contains the 32 tables that were omitted from the paper for reasons of space.

M. Dror is with The University of Arizona, USA. e-mail: (mdror@eller.arizona.edu).

G. Kendall with the ASAP Research Group, School of Computer Science, University of Nottingham, UK and University of Nottingham Malaysia Campus. e-mail: (graham.kendall@nottingham.ac.uk or graham.kendall@nottingham.edu.my)

A. Rapoport is with the Graduate School of Management, University of California, Riverside, USA. e-mail: (amnonr@ucr.edu).

Manuscript received mmm dd, yyyy; revised mmm dd, yyyy.

Appendix A

Strategy Descriptions

Strategy 0201

Goofspiel Strategy

My Goofspiel strategy was to be the best and win everything. Obviously this is not always possible, but it was a goal worth working towards. At its fundamentals, a good goofspiel strategy needs to be adaptive, hard to predict, and intelligent in the choices it makes. This is by no means an easy task, and there are various paths one can take in formulating a goofspiel strategy. Here I outline the path I have taken.

Before any choices can be made of what card to play, a player should try to calculate some data that could help them make more informed decisions. The provided information included with the goofspiel C++ code was not enough to make intelligence decisions. Thus, I built several methods within the code to capture the following data:

- What cards from the middle deck have been played within a particular round
- What cards I have played during a particular round
- What cards my opponent has played within a particular round
- The total "value" of cards that have been played by my opponent, myself, and the middle deck. Better cards such as Kings and Queens have more value than low cards like 2 or 3.
- In carry over, I also capture the total value points both players have won from round to round, as well as the last middle card played. I can check if there was a tie in a particular round by observing whether or not there was a change in the points won by either player. I can use the knowledge of the last middle card to make a more intelligent decision in the event of a carryover.

Data extrapolated from these methods could help me better predict what cards my opponent may have, my position in a round relative to the middle deck and my opponent, and what strategy might maximize my chance of winning a particular hand.

After methods to collect the aforementioned data were built, I began to develop an algorithm that would make use of such data to play cards. The goal of my strategy is to identify the relative position of the total value of cards I possess against the value of the middle deck and my opponent's deck; this is identified in every hand. Essentially, during every hand, I count the value of every card I have, what is left in the middle deck, and what my opponent has left. Then I will have a general sense of every decks position and strength, letting me make a basic prediction about what my opponent might play and what might be left in the middle deck. For example, if the opponent has used up more valuable cards than both me and the middle deck, I know it is most likely safe to pursue a more aggressive strategy in obtaining high value middle cards. Conversely, if the value of my deck is less than my opponent's but more than the middle deck, I may just continuously play my lowest cards and save my highest for the one or two high cards left in the middle deck.

I then created some arbitrary rules that adjust the value of my bets by a few cards solely to throw off predictive algorithms used by my opponent that may try to learn how much I over- or under-bet

each middle card. For example, every four rounds, I will draw all random cards for the entire round. While this may cause me to lose that round against someone with a simple deterministic strategy, it could help me greatly against those using a long-term data collection and prediction algorithm. Other times, I may substitute what is normally a calculated decision with a random one. For example, I may normally play conservatively if the value of my cards is less than my opponents, but I may randomly play a high card in these scenarios to try and obfuscate my decision making. My code has functionality for occasional randomness as another defense against pattern-seeking strategies.

Before turning in this assignment, I had the opportunity to run this against the strategies developed by Xiao Liu and Justin Williams. Justin Williams employed a more predictive strategy, and I tuned my arbitrary randomness rules to what gave me the best performance against him. I was able to beat him consistently, about every 4 out of 5 times. My defenses against predictive algorithms appeared to work as I caused his strategy to make bad predictions. Xiao Liu used a more deterministic strategy that played by some simple rules. As I did not use any sort of long-term learning and prediction, I was not able to overcome her simple strategy. Her strategy always made more precise decisions based on card value, rather than my more general value approach that has randomness. However, when she ran her strategy against Justin Williams, he was able to quickly predict her strategy and beat it consistently. In a sense, our different strategies almost resembled a form of rock-paper-scissors.

Overall, this was an interesting problem to work on and I appreciate how difficult it is to create an optimal goofspiel strategy. It appears that every strategy can be effectively countered with another strategy, and that there is no one type of strategy that can consistently win; there is no optimal solution for goofspiel. This was a very memorable project, and now I will live the rest of my life as a self-proclaimed goofspiel expert.

Strategy 0202

Goofspiel Description

I will summarize my strategy by first describing what I consider to be the base strategy (no carryover, maximizing points), then explaining how I modify that base strategy to account for the different scenarios of the game.

Base Strategy

The base strategy is based on a set of comparison ratios calculated repeatedly throughout the game. During each turn, I assess the state of the game by calculating a measure of the following four key values:

1. Value at stake: in most cases, this value is simply the value of the up card.
2. Value of my remaining cards: the value of the cards I have remaining in my hand, i.e., my remaining spending power.
3. Value of the opponent's remaining cards: the value of the cards the opponent still has in his or her hand, i.e., the opponent's remaining spending power.
4. Value remaining in the deck: the value of all cards in the middle deck which have not yet been played.

These values are combined and compared each round in a series of ratios intended to establish how "daring" I can be during the current turn, given all of the values at stake. These ratios include:

1. Up card to the remaining value in the deck: compares the value at stake to the value still available in later turns. The calculation of this value is weighted such that higher values will eventually lead to greater risk in the card I choose to play.
2. My spending power to the remaining value in the deck: compares my spending power to the value still available in later turns.
3. Opponent's spending power to the remaining value in the deck: compares the opponent's spending power to the value still available in later turns.
4. My spending power to the opponent's spending power: compares my spending power to that of the opponent.

These ratios are combined in a weighted average to produce a "daringness" score, which is value from 0 to 1 that rates how daring I should be with my next card played. This daringness score includes the following properties:

1. The score is higher if there is much value at stake, and increasingly higher as the value at stake increases. This means, for example, that when there are many points at stake, the daringness score is at or near 1, indicating that I should be very aggressive in trying to win the turn. When the value at stake is lower, other factors have more influence (like whether I or my opponent have more spending power, for example).
2. The score is specially standardized so that the values from a given turn are considered in the context of the rest of the potential value, i.e., that if there is a queen lurking in the deck near the

end (which I would know because it hasn't yet been played), the daringness score will reflect that and try to hold some higher cards in reserve in order to provide the spending power necessary to win the lurking card.

3. The score also considers my spending power as compared to that of my opponent so that, for example, if I see that my opponent has spent several high-value cards near the beginning of the game, I won't need to be as aggressive for some cards later in the game because the opponent has less potential for winning later turns.

This daringness score is used to determine which of my remaining cards I am to play. The decision simply uses the daringness score as a 100-point scale, applies the scale to an ordered set of my remaining cards, and selects the card that corresponds to the proper amount of daringness, after rounding to the nearest available index.

This flexible "daringness" framework allows me to easily adjust the strategy according to the different scenarios required, which I describe in the following sections.

Carry-over vs. No Carry-over

In games with carryover, my daringness score adapts to account for the potential of much more value at stake (in the case of a previous tie). It adds all of the carryover value (allowing for multiple sequential ties) to the value of the up card, and this additional value at stake weighs heavier in the calculation of daringness. Usually carryover produces a rather aggressive daringness score, since many more points are at stake.

Points vs. Wins

Assuming that the base case is maximizing points, I had to adjust my algorithm to allow for maximizing wins regardless of points. Again, since my "daringness" framework was flexible, I was able to add in a weighting factor to strategically account for the maximizing of wins. This consisted of establishing a "critical value" needed for the win (equal to half of the available points, plus one), and then keeping track of how close my cumulative score was to that critical value. (In the case of no carryover, maximizing wins, the value of any ties is subtracted from the critical value.) Until my score reaches the critical value, my weighting structure applies a heavier weight to the daringness score, encouraging more aggressive play until the critical value is reached. This usually means that the first half of the game is more aggressive and includes more gambles, since my daringness score is trying to get my cumulative points to the critical value.

Strategy 0203

1 Overview

In this report, I'll show the flow of my mind on solving Goofspiel, the game of pure strategy.

Main contribution of this report includes: (a) a reasonable good strategy; (b) a general way to refine any given strategy.

I am not going to explain rules of the Game, as you can always find better explanation online when searching for "Goofspiel".

Theoretical approach of solving the problem will be explored first. Then followed empirical approach based on the concept of "doing the best you can" and "If you find something is good to do, do it now".

If you are not interested in my thought flow but only the final solution, please jump to the last page directly and read the conclusion.

To narrow down my topic, a bunch of assumptions are states as follow.

2 Assumptions

1. Try to win at the end of all rounds is equal to try to win at each round. In another word, I treat opponent in each round as a new one, and delete all memory from round I played before, so I am not going to consider how to guess opponent's strategies across all rounds.
2. There are only two players in each round, me and the opponent.
3. Since my strategies do not make any difference on carryover or nocarryover case, I'll just discuss my strategies and attach simulation results on both cases.

3 Theoretical Approaches

In the approach, I am not going to guess or crack the strategy my opponent chooses, but try to find a better strategy that beats most strategies known with theoretical proof as support.

A. antiRandom

To start with, let's first assume that the opponent is play in a random way as given in the software.

Ross proved that one can win with a expect value of 28 point against opponent playing randomly by play card equal to every upcard (Ross, 1971).

This algorithm is implemented in my software with source file antiRandom.cpp. To call it, set the strategy name as "antiRandom".

Let Alice use random, and Bob use antiRandom strategy, run 100 rounds, we got the results:

```
Up Card: 8H (value = 8)
Calling gxkStrategy001
Calling antiRandom
Alice played: QC
Bob played: 8D
        Alice gets 8 points
        Alice's points at round 100: 25 (Total for all rounds: 2833)
        Bob's points at round 100: 66 (Total for all rounds: 5707)

Alice's total points: 2833
Bob's total points: 5707
```

Figure 1: carryover case

```
Up Card: KH (value = 13)
Calling gxkStrategy001
Calling antiRandom
Alice played: AC
Bob played: KD
        Bob gets 13 points
        Alice's points at round 100: 32 (Total for all rounds: 2715)
        Bob's points at round 100: 59 (Total for all rounds: 5555)

Alice's total points: 2715
Bob's total points: 5555
```

Figure 2: nocarryovercase

As we can see, Bob won an average of around 28 points in each round on both cases, which verifies Ross's proof.

B. antiRandomKiller

An easy way to against antiRandom is play (upcard+1) if upcard is not equal to 13 and play 1 when the upcard is equal to 13. So you'll only lose 13 and win 1 to 12, which sum up to 78. Thus the expect value of win is 65 points.

This algorithm is implemented in my software with source file antiRandomKiller.cpp. To call it, set the strategy name as "antiRandomKiller".

Let Alice use antiRandom while Bob uses antiRandomKiller, we got the following result for 100 rounds:

```
Up Card: AH (value = 1)
Calling antiRandom
Calling antiRandomKiller
Alice played: AC
Bob played: 2D
        Bob gets 1 points
        Alice's points at round 100: 13 (Total for all rounds: 1300)
        Bob's points at round 100: 78 (Total for all rounds: 7800)

Alice's total points: 1300
Bob's total points: 7800
```

Figure 3: carryover case

```
Up Card: AH (value = 1)
Calling antiRandom
Calling antiRandomKiller
Alice played: AC
Bob played: 2D
        Bob gets 1 points
        Alice's points at round 100: 13 (Total for all rounds: 1300)
        Bob's points at round 100: 78 (Total for all rounds: 7800)

Alice's total points: 1300
Bob's total points: 7800
```

Figure 4: nocarryover case

As we can see Bob won 65 points on average of each round on both cases.

Addtionally, I am also interested the performance of antiRandomKiller against random players. So Iet Alice use random and Bob use antiRandomKiller, we got the following result for 100 rounds:

```
Up Card: 7H (value = 7)
Calling gxkStrategy001
Calling antiRandomKiller
Alice played: 8C
Bob played: 8D
        Played the same card - no points awarded
        Alice's points at round 100: 33 (Total for all rounds: 3407)
        Bob's points at round 100: 51 (Total for all rounds: 5030)

Alice's total points: 3407
Bob's total points: 5030
```

Figure 5: carryover case

```
Up Card: KH (value = 13)
Calling gxkStrategy001
Calling antiRandomKiller
Alice played: 8C
Bob played: AD
        Alice gets 13 points
        Alice's points at round 100: 40 (Total for all rounds: 3320)
        Bob's points at round 100: 51 (Total for all rounds: 5021)

Alice's total points: 3320
Bob's total points: 5021
```

Figure 6: nocarryover case

As we can see, antiRandomKiller is also far way better than random strategy.

In all, we can say that theoretically, antiRandomKiller is better than random and antiRandom Strategy.

C. antiRandomKillerKiller

Same way as how I against antiRandom by antiRandomKiller, I can also win antiRandomKiller by antiRandomKillerKiller, which play (upcard+2)mod 13 when upcard is not equal to 11 and play 13 when upcard is equal to 13.

Let Alice use antiRandomKiller and Bob use antiRandomKillerKiller, we got the result for 100 rounds:

```
Up Card: 7H (value = 7)
Calling antiRandomKiller
Calling antiRandomKillerKiller
Alice played: 8C
Bob played: 9D
        Bob gets 7 points
        Alice's points at round 100: 12 (Total for all rounds: 1200)
        Bob's points at round 100: 79 (Total for all rounds: 7900)

Alice's total points: 1200
Bob's total points: 7900
```

Figure 7: carryover case

```
Up Card: 4H (value = 4)
Calling antiRandomKiller
Calling antiRandomKillerKiller
Alice played: 5C
Bob played: 6D
        Bob gets 4 points
        Alice's points at round 100: 12 (Total for all rounds: 1200)
        Bob's points at round 100: 79 (Total for all rounds: 7900)

Alice's total points: 1200
Bob's total points: 7900
```

Figure 8: nocarryover case

We can see that antiRandomKillerKiller won an average of 67 points each round, this is because antiRandomKillerKiller only loses when upcard is equal to 12.

Remember that antiRandomKillerKiller should beat all strategies before to be a better strategy.

Let Alice use antiRandom and Bob use antiRandomKillerKiller, we got the following results for 100 rounds:

```
Up Card: 6H (value = 6)
Calling antiRandom
Calling antiRandomKillerKiller
Alice played: 6C
Bob played: 8D
        Bob gets 6 points
        Alice's points at round 100: 25 (Total for all rounds: 2500)
        Bob's points at round 100: 66 (Total for all rounds: 6600)

Alice's total points: 2500
Bob's total points: 6600
```

Figure 9: carryover case

```
Up Card: 5H (value = 5)
Calling antiRandom
Calling antiRandomKillerKiller
Alice played: 5C
Bob played: 7D
        Bob gets 5 points
        Alice's points at round 100: 25 (Total for all rounds: 2500)
        Bob's points at round 100: 66 (Total for all rounds: 6600)

Alice's total points: 2500
Bob's total points: 6600
```

Figure 10: nocarryover case

We can see that antiRandomKillerKiller won an average of 41 points each round since it only loses when upcard=12 and upcard=13.

Let Alice use random strategy and Bob use antiRandomKillerKiller, we got the following results for 100 rounds:

```
Up Card: TH (value = 10)
Calling gxkStrategy001
Calling antiRandomKillerKiller
Alice played: 4C
Bob played: QD
        Bob gets 10 points
        Alice's points at round 100: 44 (Total for all rounds: 3957)
        Bob's points at round 100: 47 (Total for all rounds: 4406)

Alice's total points: 3957
Bob's total points: 4406
```

Figure 11: carryover case

```
Up Card: AH (value = 1)
Calling gxkStrategy001
Calling antiRandomKillerKiller
Alice played: 9C
Bob played: 3D
        Alice gets 1 points
        Alice's points at round 100: 43 (Total for all rounds: 3825)
        Bob's points at round 100: 38 (Total for all rounds: 4552)

Alice's total points: 3825
Bob's total points: 4552
```

Figure 12: nocarryover case

We can see antiRandomKillerKiller is still better than random strategy.

So we can say that so far antiRandomKillerKiller is better than all strategies ahead.

D. A **worse case: antiRandomKillerKillerKiller**

Similarly, I consider strategy antiRandomKillerKillerKiller, which play (upcard+3)mod 13 when upcard is not equal to 10 and play 13 when the upcard is equal to 10.

To save the length of report, let's do some math instead of take a picture of the results.

antiRandomKillerKillerKiller win antiRandomKillerKiller at expect value of 69 points since it loses only when upcard=11.

antiRandomKillerKillerKiller win antiRandomKiller at expect value of 45 points since it loses only when upcard=11 and upcard=12.

antiRandomKillerKillerKiller win antiRandom at expect value of 19 points since it loses only when upcard=11, 12 or 13.

But how does antiRandomKillerKillerKiller against random strategy?

Let Alice use random strategy and Bob use antiRandomKillerKillerKiller, we got the result for 100 rounds:

```
Up Card: 5H (value = 5)
Calling gxkStrategy001
Calling antiRandomKillerKillerKiller
Alice played: QC
Bob played: 8D
        Alice gets 5 points
        Alice's points at round 100: 53 (Total for all rounds: 4493)
        Bob's points at round 100: 30 (Total for all rounds: 4028)

Alice's total points: 4493
Bob's total points: 4028
```

Figure 13: carryover case

```
Up Card: JH (value = 11)
Calling gxkStrategy001
Calling antiRandomKillerKillerKiller
Alice played: JC
Bob played: AD
        Alice gets 11 points
        Alice's points at round 100: 41 (Total for all rounds: 4432)
        Bob's points at round 100: 40 (Total for all rounds: 4083)

Alice's total points: 4432
Bob's total points: 4083
```

Figure 14: nocarryover case

As we can see, win is not guarantee now. So we can't say antiRandomKillerKillerKiller is better.

E. Conclusion

For theoretical approach, antiRandomKillerKiller is the best.

4 Empirical Approaches

In this approach, I am going to somehow crack the strategy my opponent chosen and react accordingly.

A. antiTheoretical

To crack all theoretical strategies above, one should notice that, given a short sequence (say sequence length equal to 3 or 4), it's hard to find if it is random, but it's reasonable easy to find it is correlate to some other known sequence.

Let playedCardOpp donates the card my opponent played given an upcard, and playedCardMe donates the card I play given a upcard. To beat any strategy mentioned in "Theoretical way", just play

```
If (playedCardOpp+1) is not equal to 13
        playedCardMe  = (playedCardOpp+1)mod 13
Else
        playedCardMe=13
```

Algorithm 1

In order to guess what card the opponent will play given a certain upcard, we need to guess the strategy the opponent uses.

To guess the strategy of the opponent, when the opponent only take strategies among those in "theoretical way", just calculate the correlation between the played cards sequence of opponent and should played cards sequence corresponding to each strategies.

In all, the antiTheoretical strategy plays like this, play 1, 2, 3 regardless of the value of first 3 upcards.

Use algorithm 1 to play after find the strategy of the opponent. If calculated playedCardMe is equal to some card already been played, choose the closet but large unplayed card instead.

Further, to beat random strategy, just let antiTheoretical play upcard when no correlation has been found between opponent played cards sequence and any antiRandom strategies should have played cards.

While the longer the sequence I choose to calculate correlation the accurate my calculation will be, but the reason to choose only first 3 is because I'll lose 11+12+13=36 points at first in the worst case. But I'll still win in the end. If I choose first 4, then I'll lose 10+11+12+13=46 point at first in the worst case, then I got no chance to win afterwards.

Let Alice use antiRandom, antiRandomKiller, antiRandomKillerKiller, and random strategy, and Bob use antiTheoretical, we got following results for 100 rounds:

```
Up Card: 8H (value = 8)
Calling antiRandom
Calling antiTheoretical
Alice played: 8C
Bob played: QD
        Bob gets 8 points
        Alice's points at round 100: 24 (Total for all rounds: 2863)
        Bob's points at round 100: 57 (Total for all rounds: 5953)

Alice's total points: 2863
Bob's total points: 5953
```

Figure 15: carryover case

```
Up Card: JH (value = 11)
Calling antiRandom
Calling antiTheoretical
Alice played: JC
Bob played: KD
        Bob gets 11 points
        Alice's points at round 100: 38 (Total for all rounds: 2947)
        Bob's points at round 100: 53 (Total for all rounds: 5995)

Alice's total points: 2947
Bob's total points: 5995
```

Figure 16: nocarryover case

```
Up Card: 3H (value = 3)
Calling antiRandomKiller
Calling antiTheoretical
Alice played: 4C
Bob played: 3D
        Alice gets 3 points
        Alice's points at round 100: 55 (Total for all rounds: 3464)
        Bob's points at round 100: 25 (Total for all rounds: 5329)

Alice's total points: 3464
Bob's total points: 5329
```

Figure 17: carryover case

```
Up Card: 7H (value = 7)
Calling antiRandomKiller
Calling antiTheoretical
Alice played: 8C
Bob played: TD
        Bob gets 7 points
        Alice's points at round 100: 26 (Total for all rounds: 3725)
        Bob's points at round 100: 63 (Total for all rounds: 5207)

Alice's total points: 3725
Bob's total points: 5207
```

Figure 18: nocarryover case

```
Up Card: 2H (value = 2)
Calling antiRandomKillerKiller
Calling antiTheoretical
Alice played: 4C
Bob played: 9D
        Bob gets 2 points
        Alice's points at round 100: 20 (Total for all rounds: 4025)
        Bob's points at round 100: 71 (Total for all rounds: 4498)

Alice's total points: 4025
Bob's total points: 4498
```

Figure 19: carryover case

```
Up Card: 4H (value = 4)
Calling antiRandomKillerKiller
Calling antiTheoretical
Alice played: 6C
Bob played: 9D
        Bob gets 4 points
        Alice's points at round 100: 17 (Total for all rounds: 4171)
        Bob's points at round 100: 61 (Total for all rounds: 4489)

Alice's total points: 4171
Bob's total points: 4489
```

Figure 20: nocarryover case

```
Up Card: 7H (value = 7)
Calling gxkStrategy001
Calling antiTheoretical
Alice played: 6C
Bob played: JD
        Bob gets 7 points
        Alice's points at round 100: 18 (Total for all rounds: 3603)
        Bob's points at round 100: 63 (Total for all rounds: 4801)

Alice's total points: 3603
Bob's total points: 4801
```

Figure 21: carryover case

```
Up Card: JH (value = 11)
Calling gxkStrategy001
Calling antiTheoretical
Alice played: 2C
Bob played: JD
        Bob gets 11 points
        Alice's points at round 100: 40 (Total for all rounds: 3536)
        Bob's points at round 100: 51 (Total for all rounds: 4899)

Alice's total points: 3536
Bob's total points: 4899
```

Figure 22: nocarryover case

So far, we can see that antiTheoretical is better than random strategy, antiRandom, antiRandomKiller and antiRandomKillerKiller.

B. probabilityTransferMatrix

When searching on the internet, I found Glenn and Laurent claims that the solved the Goofspiel by some probablityTransferMatrix.

The matrix is listed below (The entry in row i column j is the probability with which you should play card i when card j is the initial upturned card.):

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	.0519	.0309	0	0	0	0	0	0	.0145	0	.0100
2	.4144	.2266	0	.0204	.0558	.0727	0	.0338	.0473	.0361	0	.0304	0
3	.0897	.0216	.1784	.0952	.0357	.0025	.0694	.0215	.0015	0	.0410	0	.0370
4	.4959	.2991	.0340	.0610	.0873	.0985	0	.0536	.0648	.0673	0	.0561	0
5	0	.0976	.2301	.1343	.0667	0	.1237	.0395	0	0	.0803	0	.0655
6	0	.3551	.0929	.1073	.1242	.1846	0	.0767	.1198	.0985	0	.0819	0
7	0	0	.2740	.1754	.1014	.0018	.1675	.0602	.0012	0	.1016	.0080	.0875
8	0	0	.1386	.1544	.1654	.2184	.0209	.1028	.1422	.1384	.0156	.0987	0
9	0	0	0	.2210	.1481	.0451	.2021	.0917	.0295	.0151	.1229	.0280	.1262
10	0	0	0	0	.2155	.2664	0	.1436	.1766	.1696	0	.1243	.1288
11	0	0	0	0	0	.1100	.3967	.1506	0	.0635	.2530	.0646	0
12	0	0	0	0	0	0	0	.2262	.4171	.2410	.0228	0	0
13	0	0	0	0	0	0	0	0	0	.1704	.3482	.5081	.6610

What strategy probabilityTransferMatrix does is playing the unplayed card with highest probability according to the matrix when given an upcard.

However, simulation shows this strategy is not so good or not even close.

Let Alice use random strategy, and Bob use probabilityTransferMatrix. We got the results for 100 rounds:

```
Up Card: KH (value = 13)
Calling gxkStrategy001
Calling probablityTransferMatrix
Alice played: 2C
Bob played: AD
        Alice gets 13 points
        Alice's points at round 100: 43 (Total for all rounds: 3970)
        Bob's points at round 100: 39 (Total for all rounds: 4409)

Alice's total points: 3970
Bob's total points: 4409
```

Figure 23: carryover case

```
Up Card: 9H (value = 9)
Calling gxkStrategy001
Calling probablityTransferMatrix
Alice played: 2C
Bob played: Error in get_suit_descriptionError in get_suit_description
        Alice gets 9 points
        Alice's points at round 100: 48 (Total for all rounds: 4022)
        Bob's points at round 100: 31 (Total for all rounds: 4350)

Alice's total points: 4022
Bob's total points: 4350
```

Figure 24: nocarryover case

As shown in the figure above, probabilityTransferMatrix slightly won random strategy.

Let Alice use antiRandom and Bob use probabilityTransferMatrix. We got the results for 100 rounds:

```
Up Card: 8H (value = 8)
Calling antiRandom
Calling probablityTransferMatrix
Alice played: 8C
Bob played: Error in get_suit_descriptionError in get_suit_description
        Alice gets 8 points
        Alice's points at round 100: 42 (Total for all rounds: 4930)
        Bob's points at round 100: 49 (Total for all rounds: 3885)

Alice's total points: 4930
Bob's total points: 3885
```

Figure 25: carryover case

```
Up Card: 5H (value = 5)
Calling antiRandom
Calling probablityTransferMatrix
Alice played: 5C
Bob played: 2D
        Alice gets 5 points
        Alice's points at round 100: 48 (Total for all rounds: 4832)
        Bob's points at round 100: 43 (Total for all rounds: 4020)

Alice's total points: 4832
Bob's total points: 4020
```

Figure 26: nocarryover case

As shown in the figure above, probabilityTransferMatrix can't beat antiRandom, which clarifies that probabilityTransferMatrix is not a better strategy.

C. furtherRefined

So far, the best strategy is antiTheoretical. But how do we further refine it?

Notice that there are some conditions that we should win some cards.

For example, if the current upcard is larger than any remained upcard, while your unplayed card is no less than opponent's unplayed card, you should play your largest unplayed card to win the current upcard.

Now, an intuitive way of refining antiTheoretical is to add some detectors, which detects if some conditions meet, then do corresponding actions. Example above is one of these conditions.

Add this condition and reaction to antiTheoretical and name the new strategy furtherRefined.

Let Alice use furtherRefined and Bob use antiTheoretical. We got the result for 100 rounds:

```
Up Card: 9H (value = 9)
Calling antiTheoretical
Calling furtherRefined
Alice played: 9C
Bob played: 9D
        Played the same card - no points awarded
        Alice's points at round 100: 0 (Total for all rounds: 94)
        Bob's points at round 100: 0 (Total for all rounds: 201)

Alice's total points: 94
Bob's total points: 201
```

Figure 27: carryover case

```
Up Card: 5H (value = 5)
Calling antiTheoretical
Calling furtherRefined
Alice played: 9C
Bob played: 9D
        Played the same card - no points awarded
        Alice's points at round 100: 13 (Total for all rounds: 1252)
        Bob's points at round 100: 36 (Total for all rounds: 1632)

Alice's total points: 1252
Bob's total points: 1632
```

Figure 28: nocarryover case

The result demonstrates that furtherRefined is better than antiTheoretical.

To ensure that furtherRefined is so far the best one, run it against random, antiRandom, antiRandomKiller, antiRandomKillerKiller and probabilityTransferMatrix. We found furtherRefined beat all other. Thus, furtherRefined is the best strategy I know so far.

5 Conclusion

Strategy furtherRefined is the best.

6 Further work

The way I refined antiTheoretical to furtherRefined, is a general method to refine any given strategy, including strategy furtherRefined.

Thus, the further work could be discover more qualified conditions, similar to the example, and add it the existing best strategy, then a better strategy is developed.

Reference

ROSS, S. M. (1971). Goofspiel: The Game of Pure Strategy. Journal of Applied Probability. 8,.

GLENN C. RHOADS AND LAURENT BARTHOLDI (2012), http://arxiv.org/pdf/1202.0695.pdf

Strategy 0204

Games of Pure Strategy: Goofspiel

Goofspiel, or the games of pure strategy, refers to a type of card games in which two players are involved. In the standard version of Goofspiel, two players are each endowed with a deck of cards (13 standard cards for each deck), together with an additional deck as the prizes. In each round of the game, a card from the prize deck is upturned (randomly drawn from the prize deck); after reading this prize card, two players are bidding simultaneously by showing one card from the remaining cards at hand. The player with the higher bid wins the prize in the current round, and then the other cards in the current round are discarded. When the players' bids tie each other, the prize card is also discarded in this standard version of Goofspiel. One variation is a Goofspiel with carryover, in which case the prize cards in tied rounds are carried over to the following rounds until one player successfully beats the other player.

To fix idea, suppose two players A and B, with deck a and b at hand respectively, are competing to win cards in deck c, with $a, b, c \in \{1, 2, 3, \dots, 13\}$. In round i, prize card c_i is upturned; and after observing this prize card, players A and B are bidding with a_i and b_i respectively. Comparing a_i and b_i and according to the rule stated above, the players' payoffs are determined. In the standard version of Goofspiel, the player with higher total payoff (calculated as the sum of payoffs from all rounds) wins the whole game. Again, one revision of it might be to maximize the total number of rounds in which player wins the prize card, irrespective of card values. To summarize a bit, there are four variations to consider here:

1) To maximize the total payoff without carryover;
2) To maximize the total payoff with carryover;
3) To maximize the number of rounds won without carryover;
4) To maximize the number of rounds won with carryover.

In this report I am going to discuss the "good" strategy in each of these cases, as there is no "best" strategy (or pure strategy Nash equilibrium).

1. To maximize the total payoff without carryover

For the first variant, in which case the players are aiming to maximize the total payoff and prize cards are discarded in tied rounds, it is not hard to see that when the opponent is playing in a random fashion, the best response to this naïve strategy is to match the prize cards, *i.e.*, $a_i = c_i$ suppose you are player A. However, upon rational players predicting this strategic interaction, *tit-for-tat* strategy is inevitable then. For example, if your opponent has the expectation that you play matching the prize cards, then his or her best response will be to play $b_i = c_i + 1$. Similar analysis can be conducted in this fashion. So one possible strategy is to identify your opponent's strategy in the beginning few rounds, then playing the best-response strategy to win over. One drawback of this identification strategy is that suppose the opponent's strategy does not follow this trail, then you will identify nothing in the first few rounds. Then there will be no ex ante good strategy in the sense of prisoners dilemma, where there is no pure strategy equilibrium.

However, Rhoads and Bartholdi (2012) show that we can use computer to help with solving the "best" strategies. The Goofspiel game in their setting is slightly different from the standard version. Not discarding the prize cards in tied rounds, the authors allow the two players to evenly split the value displayed in the prize cards. However, we can see that these two versions should be equivalent in terms of "best" strategies. I am going to proceed without proof, but the intuition is that when we are choosing the best strategies we care about the difference in total points won. In tied rounds the difference in the points won between the two players are the same in the standard version and in the variant discussed in Rhoads and Bartholdi (2012). Thus, with high probability we will end up with the same choices of "best" strategies. So, I will use the strategy simulated from the computer, which is showed to win most of the time in the paper. The strategy is as follows: we have the following matrix, each entry p_{ij} is the probability of playing card i when the prize

card is j, where i is index of the row and j the index of column. For example, when the prize card is shown to be 1, then a player should play card 2 with probability 0.4144.

Table 1 Matrix of Bidding Probabilities

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0*	0	.0519	.0309	0	0	0	0	0	0	.0145	0	.0100
2	.4144	.2266	0	.0204	.0558	.0727	0	.0338	.0473	.0361	0	.0304	0
3	.0897	.0216	.1784	.0952	.0357	.0025	.0694	.0215	.0015	0	.0410	0	.0370
4	.4959	.2991	.0340	.0610	.0873	.0985	0	.0536	.0648	.0673	0	.0561	0
5	0	.0976	.2301	.1343	.0667	0	.1237	.0395	0	0	.0803	0	.0655
6	0	.3551	.0929	.1073	.1242	.1846	0	.0767	.1198	.0985	0	.0819	0
7	0	0	.2740	.1754	.1014	.0018	.1675	.0602	.0012	0	.1016	.0080	.0875
8	0	0	.1386	.1544	.1654	.2184	.0209	.1028	.1422	.1384	.0156	.0987	0
9	0	0	0	.2210	.1481	.0451	.2021	.0917	.0295	.0151	.1229	.0280	.1262
10	0	0	0	0	.2155	.2664	0	.1436	.1766	.1696	0	.1243	.1288
11	0	0	0	0	0	.1100	.3967	.1506	0	.0635	.2530	.0646	0
12	0	0	0	0	0	0	0	.2262	.4171	.2410	.0228	0	0
13	0	0	0	0	0	0	0	0	0	.1704	.3482	.5081	.6610

* It is clear that probabilities in a column add up to 1, but not necessarily true for a row.

There might potentially be a problem with bidding according to Table 1, which is that there are some overlapping in each row of the matrix. For example, if in the first round card 2 is upturned, then according to the matrix you will play card 4 with probability 0.2991. Then if you indeed played card 4 in the first round and in the second round card 1 is upturned, then card 4 will not be in your hand now (discarded in the previous round). To overcome this difficulty, we can simply renormalize the probability distribution. In the example, now we simply renormalize the probability of choosing card 2 to be $0.4144/(0.4144 + 0.0897) = 0.8221$, and choosing card 3 to be $1 - 0.8221 = 0.1779$.

2. To maximize the total payoff with carryover

If the prize cards are carried over in those tied rounds, then there will be a high accumulated points in some rounds with strictly positive probability. As an example, in the first round the probability of drawing the king card from the prize deck is

$1/13$, and the probability of tied in the first round is $1/13 = 13/13^2$; then the probability of having greater than 13 points in the second round is $1/169 = 1/13 \cdot 1/13$, which is strictly positive although quite small in this case.

Under this situation, in order to maximize the total points won, a player is expected to play aggressively in those rounds with high cumulative prize[1]. As in the above example, if we happen to have a draw and the king card is upturned in the first round, then a rational player would probably wish to bid as much as he has, *i.e.*, to bid the highest card at hand in the second round. Based on this idea, I propose a strategy as follows:

1) Start with playing according to Table 1 in the first round;
2) In the later rounds, if the previous round is tied and the cumulative point is bigger than your highest card at hand, then bid the highest card; if the cumulative point is less than your highest card value at hand, then bid as if this cumulative point is the value of the upturned card in Table 1.

Here we are facing the same issue as in part 1, which is discussed right under Table 1, the overlapping issue. We adopt the same solution to that issue here.

3. To maximize the number of rounds won without carryover

In this case, the value of prize card is not as important as in the previous cases. It is possible that the player, whose total payoff earned is higher, might lose more rounds than his or her opponent. Then what would be the key consideration here? I would argue that the rankings of remaining cards at hand (both mine and opponent's) play key roles in determining who would win more rounds. As a simple example, in a certain round if it is true that my highest card is greater than the opponent's highest card, then I am going to win for sure by bidding my highest card irrespective of the prize card. I am proposing the following the strategy in this case:

1) Start with playing according to Table 1 in the first round;

[1] To calculate the cumulative prize, we simply add up the points carried over from the previous round and the upturned card value in the current round.

2) After bidding in each round, recording the opponent's bidding card so that I will be able to figure out the remaining cards at the opponent's hand before next round bidding;
3) In the remaining rounds, first comparing the highest card at my hand with the opponent's highest card. If mine is greater than the opponent's, then I bid my highest card; if my highest card is lower than his or hers, then I bid the card which has a value just above the expectation of the opponent's remaining cards, from my remaining cards of course.

For the last part in the last bulletin, let us consider the following hypothetic numeric example. Suppose in round 10, the opponent's remaining cards are $\{1, 4, 10, 11\}$ and my remaining cards are $\{2, 6, 7, 10\}$, then mean value of the opponent's remaining cards is 6.5. In this case, I will bid 7 in this round.

The justification of the last bulletin in this situation is purely probabilistic. Bidding higher than the expectation value of the opponent's remaining cards yields, on average, higher winning probability in the current round. The opponent might predict this strategy, and bid aggressively. For example, the opponent might be able to bid always the card with a value greater than the mean of remaining cards. But the apparent drawback of this response is that bidding aggressively in this way in the current round sacrifices with higher probability of losing in the future rounds. However, the strategy proposed above for this situation (maximizing number of rounds won without carryover) is not quite clear whether it will yield higher probability of winning or not, because as analyzed above the opponent's response is not quite clear either. But this strategy definitely beats the opponent's strategy of randomizing over the remaining cards.

4. To maximize the number of rounds won with carryover

As discussed in last section, the value of prize cards does not play such important role as in maximizing the total points won situation. Although in the tied rounds the prize card is carried over to the following rounds, the analysis of game strategy is not quite different from that in last section. So I will use the same strategy here as in the last section.

Reference

[1]. Dror, M., T. Chan, and (Ed.) M. Hazewinkel (2012) Goofspiel, *Encyclopedia of Mathematics*, Supplement IV, Springer (in press).

[2]. Rhoads, G. C., and L. Bartholdi (2012), Computer Solution to the Game of Pure Strategy, *Games*, 3: 150-156.

Strategy 0205

At first glance, Goofspiel is a very simple game. Each player has 13 cards worth 1 to 13 points which they use to bid on randomly ordered cards valued at 1 to 13 points. Nothing is hidden – you know what cards you have, what cards your opponent has, and what cards remain in the deck. When trying to decide which card to play, however, the complexity rises quickly.

One sensible strategy is to play a card equal to the value of the up card – winning an ace has very little value so you should not wager many points on it, and winning a king is very valuable so it is seemingly worth it to risk more points. Using this thinking to consider how to beat your opponent, however, quickly leads to an arms race. If you think your opponent will match the up card – a 10 for example – it makes sense to attempt to beat your opponent by one by playing a Jack. However, your opponent may realize this and decide to beat your Jack with a Queen. As the values continue to rise, at some point you may assume that your opponent will not want to waste such a high card and will instead "throw away" the hand with a low card such as a 2. This gives the nice opportunity to win a valuable card with little expenditure by playing a 3. This circular logic could continue ad nauseam.

As described in this this scenario, naïve strategies that only consider the value of the up card can easily become overwhelmed. For this reason, my strategy involves a number of more sophisticated elements described in this paper, including:

- Tiered valuation of cards
- A learning algorithm based on monitoring of historical play
- Bounded random play
- Mitigations against learning algorithms

Four versions of my goofspiel algorithm were created: maximize points (carryover), maximize points (no carryover), maximize wins (carryover), and maximize points (no carryover). Each version uses the same basic set of strategies with a few unique features as discussed later in this report.

Starting the Game

To clarify the terminology as I will be using it in this report:

- Hand – When a new up card is revealed and each player bids. There are 13 hands per round
- Round – A series of 13 hands, during which each card A-K is randomly revealed as the up card
- Game – A series of several rounds played in succession

In a normal game of Goofspiel (no carryover, maximize wins), the goal is to win one more than half of the available points in a given round. At the start of each round, there is a total of 91 available points in the deck, so the goal is to win 46 points. As the game progresses, however, ties may occur causing points to be discarded and thus lowering the total number of points needed to win. Therefore at the start of each hand we first calculate the number of points needed to win:

Winning Score = (My Score + Opponent Score + Points remaining in deck)/2

For convenience, we then determine the number of outstanding points needed to win the game:

Points I need = My Score – Winning Score

Tiered Valuation

Once we have determined how many points are needed to win, we want to find out how to get as quickly as possible. To do this, calculate the value of winning each card in the deck (Figure 1). At the start of each round the number of points needed is always 46, and these values are always the same:

```
K: 0.28 Q: 0.26 J: 0.24 T: 0.22
9: 0.20 8: 0.17 7: 0.15 6: 0.13
5: 0.11 4: 0.09 3: 0.07 2: 0.04 A: 0.02
```

```
Value of winning 1: 0.0217391
Value of winning 2: 0.0434783
Value of winning 3: 0.0652174
Value of winning 4: 0.0869565
Value of winning 5: 0.108696
Value of winning 6: 0.130435
Value of winning 7: 0.152174
Value of winning 8: 0.173913
Value of winning 9: 0.195652
Value of winning 10: 0.217391
Value of winning 11: 0.23913
Value of winning 12: 0.26087
Value of winning 13: 0.282609

 found a set of winners with 4 cards!
            Needed to win: 46
Need this many more points: 46
```

Figure 1

To find the shortest path to the winning score, the values of each card are added to find combinations that have a sum greater than or equal to 1. In the first round, we need 46 points to win, which is most quickly accomplished with four cards: **K, Q, J, 10.** Using a **tiered valuation system**, these cards are stored in a **winners** array, because they are the cards that can win the game in the minimum number of plays. Once the **winners** have been determined, a second valuation tier called **helpers** is created. Any two **helper** cards may be used to replace any one **winner** card up to the highest winner. In the first hand of a round, the **helpers** are **9, 8, 7, and 6**. The sum of any two helpers must be greater than or equal to the highest winner – e.g. five is not a helper because despite 9 + 5 >= 13 being true, 6 + 5 >= 13 is not true. The final tier of the valuation system is **tossers**, which are cards that would require three or more to be useful. Unless the learning algorithm determines we can easily win them (as described later), the bounded random algorithm will dispose of these by playing the lowest available card. Ultimately we wind up with three arrays configured as shown in Figure 2.

```
          123456789TJQK
Winners: 0000000001111
Helpers: 0000011110000
Tossers: 1111100000000
```

Figure 2

As the game progresses, however, the number of points needed to win changes for three reasons:

- Ties reduce the number of points needed to win
- You may lose some of the cards you need for the shortest path to victory
- You may win smaller cards, which expand your options for shortest path to victory

```
              123456789TJQK
myCard Status: 1111111010110: 60
opCard Status: 0111111011110: 69
dkDeck Status: 1111111001110: 61
              123456789TJQK
MyWins Status: 0000000010000: 9
OpWins Status: 0000000000000: 0
  Ties Status: 0000000100001: 21

Value of winning 1: 0.037037
Value of winning 2: 0.0740741
Value of winning 3: 0.111111
Value of winning 4: 0.148148
Value of winning 5: 0.185185
Value of winning 6: 0.222222
Value of winning 7: 0.259259
Value of winning 8: 0
Value of winning 9: 0
Value of winning 10: 0.37037
Value of winning 11: 0.407407
Value of winning 12: 0.444444
Value of winning 13: 0

 found a set of winners with 3 cards!
            Needed to win: 36
Need this many more points: 27

         123456789TJQK
Winners: 0000001001110
Helpers: 0000010000000
Tossers: 1111100000000
```

Figure 3

In Figure 3, we show a situation where three cards have been turned up: 8, 9 and K. Ties occurred on 8 and K, thereby reducing the total points available in the game to 70, and the winning score to 36. We won the 9, so the total points we need to win is 27.

Starting with the remaining cards of the highest value and working down, we find that the shortest path to 27 is **Q, J, 10** (34 points) using **three cards**. Now we look for other solutions using three cards, and find another: **7, 10, J** (28 points).

Of course there are other three card solutions such as **6, J, Q** (29 points), however, the winner cards should be such that any combination using the minimum number of cards to win is greater than or equal to the winning score. In this example, if six were included, we see that some of the three card combinations (such as **6, 7, 10** = 23 points) would not win, so 6 is demoted to "helper" status.

These tiered values are recreated at the start of each hand and provide a reasoned baseline for decisions for the learning algorithm and bounded random strategies described below.

Monitoring Historical Play

This strategy uses a learning algorithm which monitors what the opponent has played against each up card. Of course this information becomes more valuable as more history is recorded and can be dangerously inaccurate when little historical data is known, so we wait for a set number of games (10% or 10 games, whichever is smaller) before starting to use this information. Until this threshold is reached, and again if historical data is insufficient to make an informed decision, we used the bounded random strategy described in the next section. For games with very few rounds we may start using the historical data pre-maturely (with as few as one previous observation), but we attempt to offset this by checking (described in the "sanity check" section) to ensure that the played card is not significantly higher than the card that would have been played were we not using the learning algorithm.

Figure 4 shows the state of the learning algorithm after 35 rounds. We see that with a 10 as the up card, the opponent has played ace 16% of the time, three 25% of the time, and ten 58% of the time. The algorithm takes only cards currently in the opponent's hand into consideration, so if the opponent

```
Op against a 10? 1 - 0.166667 Cumulative: 0.166667
Op against a 10? 2 - 0 Cumulative: 0.166667
Op against a 10? 3 - 0.25 Cumulative: 0.416667
Op against a 10? 4 - 0 Cumulative: 0.416667
Op against a 10? 5 - 0 Cumulative: 0.416667
Op against a 10? 6 - 0 Cumulative: 0.416667
Op against a 10? 7 - 0 Cumulative: 0.416667
Op against a 10? 8 - 0 Cumulative: 0.416667
Op against a 10? 9 - 0 Cumulative: 0.416667
Op against a 10? 10 - 0.583333 Cumulative: 1
Op against a 10? 11 - 0 Cumulative: 1
Op against a 10? 12 - 0 Cumulative: 1
Op against a 10? 13 - 0 Cumulative: 1
```

Figure 4

typically plays a jack against a ten, but does not currently hold the jack, it will be excluded from the calculation since they obviously could not play that card. If we have seen a single card played 50% of more of the time, we assume this is the card they will play thus will play that card +1 (in this case, we would play an 11). If we do not have the desired card to play, we try the next highest card until we find one that is available. Of course this can lead to situations of overbidding, which are mitigated by the sanity check process described later in this report.

Often our opponent will not have played a single card consistently. For this reason we also look at the cumulative likelihood of each card. In Figure 5 we show the historical play with nine as the up card. In this case, the opponent has played ace 26% of the time, two 3% of the time, three 7% of the time, ten 23% of the time, jack 20% of the time, and queen 20% of the time. No single card stands out as the single most likely play. By looking at the cumulative history, I see that if I really want to win this nine, I have a very good chance of winning with a king, about 80% chance of winning with a queen, 60% chance of winning with a jack, and so on.

```
Op against a 9? 1 - 0.266667 Cumulative: 0.266667
Op against a 9? 2 - 0.0333333 Cumulative: 0.3
Op against a 9? 3 - 0.0666667 Cumulative: 0.366667
Op against a 9? 4 - 0 Cumulative: 0.366667
Op against a 9? 5 - 0 Cumulative: 0.366667
Op against a 9? 6 - 0 Cumulative: 0.366667
Op against a 9? 7 - 0 Cumulative: 0.366667
Op against a 9? 8 - 0 Cumulative: 0.366667
Op against a 9? 9 - 0 Cumulative: 0.366667
Op against a 9? 10 - 0.233333 Cumulative: 0.6
Op against a 9? 11 - 0.2 Cumulative: 0.8
Op against a 9? 12 - 0.2 Cumulative: 1
Op against a 9? 13 - 0 Cumulative: 1
```

Figure 5

Sanity Check

One problem with using the historical play percentage is that it may lead you to play far too high of a card. This is especially prevalent in later rounds when the opponent might have fewer appropriate cards to play against a given up card, which can dangerously skew the results. For this reason, at the end of card selection, a sanity check occurs. If the card you are playing is more than three higher than the value of the up card and the up card is not in the winner array, the recommendation is discarded and the bounded random strategy takes over. For example, in Figure 6 the up card is a five and there is no value the opponent has consistently played against a five (they are pretty evenly split between 3, 6, 8, and 10). In this case we look at the cumulative score, which indicates we should play a jack to win the five. In most cases the jack in our hand is much more valuable than a five, so the sanity check will discard this recommendation. If, however, the five is in my "winner" array, I will play whatever is necessary to win the card - for example, if I only need four more points, I will gladly play the jack to win the five.

```
Op against a 5? 1 - 0 Cumulative: 0
Op against a 5? 2 - 0 Cumulative: 0
Op against a 5? 3 - 0.275862 Cumulative: 0.275862
Op against a 5? 4 - 0 Cumulative: 0.275862
Op against a 5? 5 - 0 Cumulative: 0.275862
Op against a 5? 6 - 0.241379 Cumulative: 0.517241
Op against a 5? 7 - 0 Cumulative: 0.517241
Op against a 5? 8 - 0.241379 Cumulative: 0.758621
Op against a 5? 9 - 0 Cumulative: 0.758621
Op against a 5? 10 - 0.241379 Cumulative: 1
Op against a 5? 11 - 0 Cumulative: 1
Op against a 5? 12 - 0 Cumulative: 1
Op against a 5? 13 - 0 Cumulative: 1
```

Figure 6

Bounded Random Play

At the start of the game, the learning algorithm has no data to act on, so we use a strategy of bounded random play. Additionally, if the sanity check determines the learning algorithm as picked a seemingly

bad card (i.e. bidding Jack for 5), the bounded random strategy is used. Using the tiered valuation of each card to determine upper and lower bounds, we pick a random card from a reasonable subset of options. The bounds used are:

- If the card is in the **winners array**, play one of the highest available cards, pseudo randomly selected with the up card as the lower bound and the highest available card as the upper bound. However, if my opponent has cards higher than all of the cards in my bounded subset, I will play my lowest available card rather than playing a valuable card when defeat is likely. For example:
    - K – Play King or Lowest Available Card
    - Q – Play Queen, King or Lowest Available Card
    - J – Play Jack, Queen, King or Lowest Available Card
    - T – Play Ten, Jack, Queen, King or Lowest Available Card
- If the card is in the **helpers array**, randomly play up card, up card + 1, or up card + 2.
- If the card is in the **tossers array**, play lowest available card

This bounded random play strategy ensures that the card played is reasonable considering the value of the up card, but introduces enough randomness into the play to make it difficult for learning algorithms to predict my play. It also takes into account likely losses so we do not play a high card when a loss is very likely.

Finally, on the first up card of the games before the learning algorithm is trained, there is very little information available to make a rational choice. Since we have no good data to make a decision on, we build on the work of Rhoads and Bartholdi (2012) in picking a statistically reasonable card (Figure 1). This strategy is only used on the first card of the round, and only on rounds before the learning algorithm is trained, so this strategy is employed between one and ten times.

	upcard												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	0	.052	.031	0	0	.020	0	0	0	.014	0	.010
2	.414	.227	0	.020	.056	.073	0	.034	.047	.036	0	.030	0
3	.090	.022	.178	.095	.036	.002	.069	.021	.002	0	.041	0	.037
4	.496	.299	.034	.061	.088	.098	0	.054	.065	.067	0	.056	0
5	0	.098	.230	.134	.067	0	.124	.039	0	0	.080	0	.065
6	0	.355	.092	.107	.124	.185	0	.077	.120	.098	0	.082	0
7	0	0	.274	.175	.101	.002	.168	.060	.001	0	.102	.008	.087
8	0	0	.139	.154	.165	.218	.021	.103	.142	.138	.016	.099	0
9	0	0	0	.221	.148	.045	.202	.092	.029	.015	.123	.028	.126
10	0	0	0	0	.215	.266	0	.144	.177	.170	0	.124	.013
11	0	0	0	0	0	.110	.397	.151	0	.063	.253	.065	0
12	0	0	0	0	0	0	0	.226	.417	.241	.023	0	0
13	0	0	0	0	0	0	0	0	0	.170	.348	.508	.661

Figure 7: Rhoads and Bartholdi optimal strategy for first move, N=13

Mitigation Against Learning Algorithms

In additional to using a bounded random strategy to thwart learning algorithms, in games where we are not trying to maximize points, we initiate a “suicide mode” once we have definitely won or lost the

game. With this strategy, once we have won or lost, we play our lowest available card on every hand. In this way we attempt to corrupt the data of the opponent's learning algorithm.

Special Cases

There are two special cases of the game we consider: carryover and points maximization.

Carryover

When playing with carryover instead of discarding the up card when a tie occurs, the up card "carries over" to the next round, increasing the size of the pot. For example, if a two carries over onto a three, the total value of the pot is five, which is not terribly exciting. However, if a medium size card such as a seven carries over only a medium size card such as an eight, the pot size is increased to fifteen, which is a very valuable pot.

Carryover can create significant changes in scorekeeping because not only are values much higher than 13 feasible, but values may also be repeated during a round (for example, you may have a 5 up card, but also a 2/3 carryover with a value of 5). When playing carryover games, I always consider the total value of the pot as the up card. I think this is the most reasonable strategy overall, and while some opponent strategies may consider either the combined value or the latest up card value, I believe it would be counterproductive to attempt to accommodate these individually. For pots that are over 13, we play just as if the value was 13 since there is unfortunately no way to play more aggressively to win a pot with 18 points than how the normal strategy plays to win a pot with 13 points.

One very interesting characteristic of carryover is the **value of ties**. When ties occur, two lower value cards are combined to potentially make a high value pot – for example turning a 7 and 9 into 16 points. With this in mind, my algorithm attempts to **actively create ties** if the following three criteria are met:

- The first up card is not already in the winner array
- We have a card that is higher than my opponents highest card
- The learning algorithm believes there is at least an 70% chance of being able to tie

The first point is important because if we win the cards in the winners array, getting the big tie values don't really matter. If we can win a winner, we should just go ahead and do it instead of being greedy. The second point is important because if we do not know for certain that the opponent can be beat, we do not want to intentionally create a big pot for them to win. The third point is important because we want to be reasonably sure we will be able to create a tie if that is what we intend to do.

As I write this, I realize one strategy that might have worked well would be to play less aggressively in the early hands of carryover games in order to increase the probability of having strong cards when large carryover pots are created. Unfortunately I realized this very late and was not able to modify my strategy in time for the competition.

Max Points

At first glance the best strategy for maximizing points would seem to be winning as many hands as

possible; however, I believe this will ultimately lead to big problems. Instead, I propose a similar strategy to what is used for maximizing wins, but with a higher point target. The reason for this is that even when playing for max points, if you are playing against a reasonable opponent you will still have to sacrifice hands occasionally to get the big wins. Therefore, we keep my same winners/helpers/tossers scheme, but instead of going for half+1 of the available points, we go for about 2/3 of the available points. Since it is not reasonable to attempt to win 90/91 points, we instead shoot for 63/91 points. As shown in Figure 8, this changes the winner array on the first hand to **K, Q, J, T, 9, 8**.

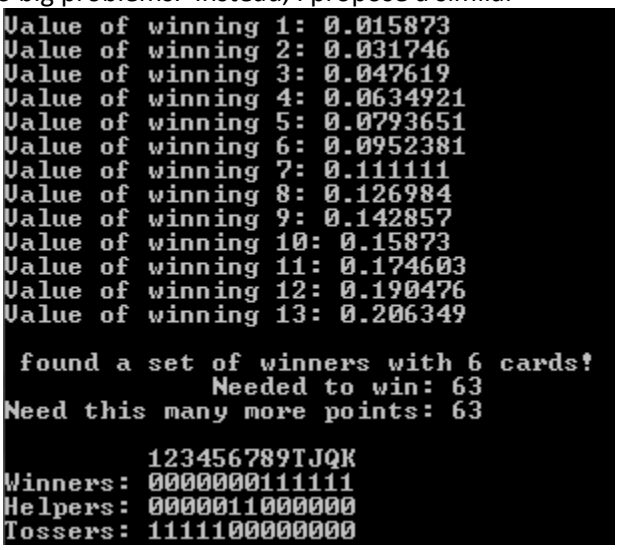


Figure 8

One other difference in max point games vs. regular games is that we do not go into the "suicide mode" described earlier as we do in a max wins game. While this does reduce our defense against AI opponents, it is sensible to not unnecessarily give up points.

Conclusion

While goofspiel seems to be a simple game at first, constructing a strategy that can win against a variety of opponents is a significant undertaking. The strategy described in this paper uses a number of approaches including learning algorithms, bounded random play, strategies to disrupt other AI strategies, strategies to increase value of carryover games, and more. The more you think about a game such as this, the more you think about special cases that can make a big difference such as the value of ties in carryover games and how valuable machine learning techniques can be. It also helps illustrate that machine learning algorithms can sometimes go awry and create big problems (for example, bidding large cards on low value up cards) – for this reason it is important to build in checks and balances to account for deviations such as this.

References

Rhoads, Glenn C., and Laurent Bartholdi. "The Game of Pure Strategy is solved!." *Retrieved from http://arxiv.org/pdf/1202.0695.pdf November 12, 2012.*

Strategy 0206

Goofspiel Competition

Introduction

Goofspiel, also called Game of Pure Strategy (GOPS) is a two-person game. A bet consists of turning up the next card from the middle pile and then the two players 'bet' on the upturned card, each player choosing one card and then simultaneously displaying it to the other player. The player showing the highest card wins the value of the upturned card. These three cards are then discarded. The game ends after 13 bets and the winner is the person who obtained the most points (46 points or better are in need to win).

Goofspiel also varies according to the rules when two players bet the same cards for an upcard.

If two players bet the same card for up card, the up card remains on the table and add up to the points of next card till two players have different bets. This is goofspiel with carryover. On the other hand, if two players bet the same card for the middle deck, neither of them get the points and three cards are discarded. This is goofspiel with no carryover.

The different rules in goofspiel games make the strategy to bet on card different. Also strategies are different when players want to gain maximum points and maximum wins.

We will explore the strategies for goofspiel with carryover and goofspiel with no carryover to maximize the points gained and to maximize the number of wins.

Goofspiel with carryover

Maximize the points

Goofspiel game with carryover to maximize the point aims to gain as many points as possible.

A measure called “Point Quality” is used to evaluate how efficient past cards in one round are spent. Both my point quality and opponent’s points quality are calculated each bet. Every new card I play is aiming to enhance the difference of two players’ point quality.

$$Point Quality(Ai) = \frac{\sum_{j=0}^{j=i} A's\_reward\_in\_bet(j)}{\sum_{j=0}^{j=i} A's\_bet(j)}$$

Point Quality (Ai) refers to the Point Quality of player A after the i_{th} bet.

Eg. The table below shows how Point Quality is computed in 1 round.

Upcoming Card	10	7	2	11	6	12	3	4	9	13	1	5	8
Player A 's bet	12	9	1	13	8	2	3	4	11	10	5	6	7
Player B's bet	11	7	2	12	9	13	4	5	10	8	1	6	3
Point quality (A)	10/12	17/21	17/22	28/35	28/43	28/45	28/48	28/52	37/63	50/73	51/78	51/84	64/91
Point quality(B)	0/11	0/18	2/20	2/32	8/41	20/54	23/58	27/63	27/73	27/81	27/82	27/88	27/91

CarryOver points are recorded in order to adjust the number of points (bet reward) I can get for this bet.

I arbitrarily set up the boundary that for bet reward smaller than 5 I give my smallest card in my deck.

For bet reward larger than five and smaller than 10, I will compare my points quality measure with my opponent's. If my quality measure is better, I can bet more aggressively, which means I can pay 2 points more than the bet reward if the card is not played yet. Otherwise, I will play at most 1 point more than the bet reward.

For rewards larger than 10, I will check the largest card in hand of both my opponent and me. If my opponent has the larger card, I will bet conservatively. If I have the larger card, I will bet one more than my opponent's largest card.

The result shows that having a measure of point quality is helpful to adjust the aggressiveness of the strategy in terms of maximizing the points I can get.

Maximize the number of wins

Goofspiel with no carryover to maximize the wins aims to gain as many wins as possible.

CarryOver points are recorded in order to adjust the number of points I bet for (bet reward). I set up the goal to get 46 points initially.

When I am close to the target, I will play more aggressively in order to win this round, which means I will bet 2 points more than the bet reward and even sometimes I will use my largest card in order to get the points to win the game.

When a carryover happens, I check my largest card and my opponent's largest card to see if I can get the points of this round by using my largest card. If my opponent is about to win, I use my largest card anyway. If not, I will bet a conservative number.

Since the goal of each round is just to win, as long as I can get enough points to win, it doesn't matter how many points my opponent gets. The measure of "Points in Need to win" helps me adjust the aggressiveness of my strategy to approach the wins.

Here I also arbitrarily set up the boundary that for bet reward smaller than 5, I will just give my smallest card.

Goofspiel with no carryover

Maximize the points

Goofspiel game with no carryover to maximize the point aims to gain as many points as possible.

A measure called “points Quality” is used to evaluate how efficient my cards are spent. Every new card I play is aiming to enhance the value of this measure.

$$Point Quality(Ai) = \frac{\sum_{j=0}^{j=i} A's\_reward\_in\_bet(j)}{\sum_{j=0}^{j=i} A's\_bet(j)}$$

Point Quality (Ai) refers to the Point Quality of player A after the i_{th} bet.

Eg. The table below shows how Point Quality is computed in 1 round.

Upcoming Card	10	7	2	11	6	12	3	4	9	13	1	5	8
Player A 's bet	12	9	1	13	8	2	3	4	11	10	5	6(Tie)	7
Player B's bet	11	7	2	12	9	13	4	5	10	8	1	6(Tie)	3
Point quality (A)	10/12	17/21	17/22	28/35	28/43	28/45	28/48	28/52	37/63	50/73	51/78	51/84	59/91
Point quality(B)	0/11	0/18	2/20	2/32	8/41	20/54	23/58	27/63	27/73	27/81	27/82	27/88	27/91

I arbitrarily set up the boundary that for upcard smaller than 5, I will just give my smallest card.

For cards larger than five and smaller than 10, I will compare Points Quality measure with my opponent. If my quality measure is better, I can play more aggressively, which means I can pay 2 points more than the upcard if the card is not used yet. Otherwise, I will play at most 1 point more than the up card.

For cards larger than 10, I will check the largest card in hand of both my opponent and me. If my opponent has the larger card, I will play less aggressively which means I at most bet 1 point more than the up card. If I have the larger card, I will bet one more than my opponent’s largest card.

The result shows that having a measure of points quality is helpful to adjust the aggressiveness of the strategy in terms of maximizing the points I can get.

Maximize the number of wins

Goofspiel with no carryover to maximize the wins aims to gain as many wins as possible.

Discarded points are recorded in order to adjust the number of points I need to win and monitor my opponent's progress. I set up the goal to get 46 points initially.

When I am close to the target, I will play more aggressively in order to win this round, which means I will bet 2 points more than the up card and even sometimes I will use my largest card in order to get the points to win the game.

Since the goal of each round is just to win, as long as I can get enough points to win, it doesn't matter how many points my opponent gets. The measure of "Points in Need to win" helps me adjust the aggressiveness of my strategy to approach the wins.

Here I also arbitrarily set up the boundary that for upcards smaller than 5 I will just give my smallest card.

Conclusion

We did a small experiment with 3 people.

My strategy is easier to predict because in most situation, my upper bound for a card is 2 points more than the upcard. This can be adjusted in the future to increase ambiguity in opponent's prediction.

When compared to strategies without prediction on my upper bound, my strategy performs better.

Strategy 0207

Goofspiel, the name given to The Game of Pure Strategy (GOPS), is a popular tool for game theorists. Here is a quick summary of the game rules:

- Suits of cards are separated and one of the four suits are set aside
- Two players are then given a complete suit each and the third remaining suit is placed in a central common pile
- Each player can see and choose from any of their own cards, but can't see their opponents cards until they are played.
- The common pile is placed face down and the top card is flipped up one at a time.
- That card is determines a point value of that round, with A=1 point and K=13 points.
- Each player plays one card each simultaneously and the player with the higher score wins the value of that central card.
- The first player to 46 points wins the hand.

For this project, I have implemented a strategy that brings together 4 separate and linked strategies. I will outline each of the four below.

Strategy 1: Low Early

This simple strategy sorts through and tracks which are the highest unplayed UpCard, MyCard and OpCard. Then it looks to see if the highest MyCard is greater than higest OpHand card. Only then does it play the high card. In all other cases it plays the low card.

This is based on the idea that if you can gain a systematic advantage over the opponent, you will be able to win routinely. By throwing low hands early, you are hoping to catch the opponent throwing out their K. After that card is played, you will have the higher card on all of the remaining hands, and thus an advantage.

Strategy 2: Plus 3

This strategy begins with the idea that a player would want to play their best card (King) only when they would win the most points (upCard = King). However, this incentive applies to both players, and thus could result is no points being awarded when both players play a King. So instead of playing the Upcard, this strategy plays the upcard+3. This means that when the upcard is a 10, you will play your king. When it's a 9 you will pay the Queen. This means that you are likely to be giving up the Jack, Queen and King because when those are the upcard, you will be playing you're A, 2 or 3.

In addition to this basic rule, the strategy also implements a random card being thrown if you are over 46 points. So than any learning efforts made by the opponent are confused a bit.

Strategy 3: Random

This strategy is very simple. Play a random, unplayed card.

Strategy 4: UpCard +/- 1

This strategy is quite simple and similar to Strategy #2 above. In this case, the it looks to play the UpCard, but if that isn't available, it looks to play UpCard+1. And if that's not available, it plays UpCard-1. And if that's not available, it throws an available random card.

Linkage between Strategies

Each of these 4 strategies is linked together with a simple learning algorithm. The default strategy is #2 above. This strategy is played all of the time to start. However, if after 40% of the rounds are played, the strategy is losing, the system moves to the first alternate strategy. In this case, it trys the UpCard +/-1 strategy. If that trend hasn't reversed by the time 50% of the cards are played, then the system switches to the alternate strategy, lowest cards early followed by highest cards later. And if by the time 75% of the cards have been played the system is still behind it switches to a random available cards the rest of the time.

Testing

This strategy has been quite successful. When testing against basic strategies like play = upcard and or play = randomcard this strategy wins. It didn't matter when using carry over or no carry over, keeping score of points or wins.

When tested against various editions of other student strategies, it was either routine the winning strategy in all cases or it was very close and competitive in all cases.

One Single strategy

The project was developed with the idea that several different strategies will be used to applied across each of the different game conditions (carry over, no carry over, keeping score of points or wins). However during testing it was determined that a single comprehensive strategy was very effective in all conditions. As such this same integrated strategy is submitted to be played in all game testing conditions.

Strategy 0208

1. Overview

I have two major algorithms designed for Goofspiel without considering carryover at the first place. One is for maximizing the number of winning (denoted as **Algorithm.1**), and another one is for maximizing the total points of earning (denoted as **Algorithm.2**). Then I am going to discuss how modified version of both algorithms is still applicable when carryover is allowed.

I will present my both algorithms in *section 3* and *section 4* as the manner below:
First, I will start with talking about my interpretation of Goofspiel and the motive of my design.
Second, specific concepts and methods used by the algorithm will be introduced in this part.
Third, the breakdown of algorithm with detailed description, flowchart and pseudocode.
Last, weakness of the algorithm and further work will be discussed

At the last part of this report, you can find the competition results I obtained from running my algorithms against other existent Goofspiel algorithms and corresponding analysis.

2. Methodology and Definition

Before we go to the next section, let me introduce the methodologies and definitions I have kept using in my algorithms.

2.1 Ideas

The ideas I use are summarized as below:

a) Performance evaluation: take into account only my own performance during the game and ignore how my opponent behaves; (used **by Algorithm.1**)
b) Countermeasure: keep track of opponent's behavior during the game and react accordingly; (used **by Algorithm.2**)
c) Randomness: integrate randomness into the algorithm to make it difficult for my opponent to predict; (used **by Algorithm.1** and **Algorithm.2**)

2.2 Definition

For convenience, I have following definitions:

- ***value cards***: denotes the cards used by two player in the game;
- ***point cards***: denotes the cards from the third suit to turn up.

There is some calculation in my algorithms. Here are the notations:
Sets:

$I \in \{1, 2, ..., 13\}$: Standard deck of card for two players (**value cards** or **point cards**);

$k \in \{1, 2, 3, ..., 13\}$: There are 13 rounds to play for each game;

$n \in \{1, 2, ..., N\}$: There are N games to play for both players before one quits.

Variables:

v_i : Value of card (value cards) for a player to play ($v_1 = 1$, $v_2 = 2$, ..., $v_{13} = 13$);

p_i Point of card (point cards) from the third suit ($p_1 = 1$, $p_2 = 2$, ..., $p_{13} = 13$);

$ownotal_k^n$: My total points granted at n_{th} game till round k;

$opptotal_k^n$: My opponent's total points granted at n_{th} game till round k;

$ownravg_k^n$: Average value of my remaining hands at n_{th} game till round k;

$oppravg_k^n$: Average value of my opponent's remaining hands at n_{th} game till round k.

3. Maximizing number of winning (Algorithm.1):

If the setting is to maximize total number of winning, one is just trying to beat his/her opponent in a game but he/she does not care about the differential (could be 1, 2, 10, …or any positive number). Hence, his/her strategy might not necessarily be very aggressive as long as he/she is still leading at the end of each round. In other words, the user has no utility for extra points, which is:

$$u(lose) = 0, \quad u(win) = 1;$$

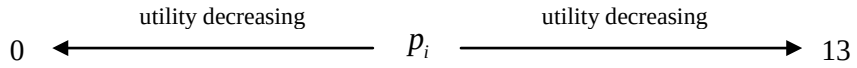
So I propose the following algorithm in order to get the optimal number of winning:

3.1 Utility function

First and foremost, I come up with this utility function to calculate the utilities for the value cards available to play at each round.

Regardless of anything, it should always be true that the higher value card I uses, the better chance will I have to collect the points at a round. On the other hand, I do not want to waste my "good" value card for a "bad" point card, say I don't want to use my King for a measly 1 or 2 points. So the utility has something to do with both the value card to play and point card upturned,

$$u(v_i, p_i) = \frac{1}{|v_i - p_i| + 1}, \text{ if } v_i \neq p_i \text{ ; otherwise 1.} \qquad (1)$$



It should make sense as utility is decreasing at both sides when value card is getting bigger and lesser, and biggest utility is attained when value card matches point card [1].

Clearly, I didn't use any dynamic data from the game in this utility function. In order to make it specific to different situation, I need to incorporate more data generated from the game to revise the function. Consider several scenarios decomposition below for **revision**:

3.2 Scenario decomposition

First scenario: when I am falling behind in terms of total points gathered but I hold better remaining hands, i.e.

$$owntotal_k^n < opptotal_k^n,\ \ ownravg_k^n > oppravg_k^n;$$

Then it is logical that I may be willing to play a higher value card to beat my opponent for the upturned point card. So, I make the revision:

$$f^{s_1}(v_i, p_i) = \frac{1}{v_i - p_i} \times (opptotal_k^n - owntotal_k^n) \times (ownravg_k^n - oppravg_k^n), \text{ if } \ v_i > p_i$$

Second scenario: when I am falling behind by total points and unfortunately at the same time my remaining hands is worse than my opponent's, i.e.

$$owntotal_k^n < opptotal_k^n,\ \ ownravg_k^n < oppravg_k^n$$

This time, I should be very careful and manage to make a good use of my value cards as the value cards I own are not as good as my opponent's. The revision here is:

$$f^{s_2}(v_i, p_i) = \frac{1}{v_i - p_i} \times \frac{(opptotal_k^n - owntotal_k^n)}{(oppravg_k^n - ownravg_k^n)}, \text{ if } \ v_i > p_i$$

Third scenario: when I am both leading and happen to have better remaining hands at the same time, i.e.

$$owntotal_k^n > opptotal_k^n,\ \ ownravg_k^n > oppravg_k^n;$$

Then I am even able to squander! And,

$$f^{s_3}(v_i, p_i) = \frac{1}{p_i - v_i} \times \frac{(ownravg_k^n - oppravg_k^n)}{(owntotal_k^n - opptotal_k^n)}, \text{ if } \ v_i < p_i$$

Fourth scenario: when I am leading but my remaining hands are not as good as my opponent's i.e.

$$owntotal_k^n > opptotal_k^n,\ \ ownravg_k^n < oppravg_k^n$$

Since I am still leading, I maybe become a little conservative, as there is no urgency for catching up points score.

$$f^{s_4}(v_i, p_i) = \frac{1}{p_i - v_i} \times (owntotal_k^n - opptotal_k^n) \times (oppravg_k^n - ownravg_k^n), \text{ if } \ v_i < p_i$$

I will just eliminate the item(s) when any of them is equal to 0 because it makes the function meaningless (0, or infinity) and the utility function follows the same pattern as above.

3.3 Monte Carlo simulation

Monte Carlo simulation here follows the ideas of "**I want to know the effect for the next round if I play this value card at the current round**".

As the utility function created can be used for any round to assess the utilities of the value cards at hand, it is possible that I look at multiple rounds instead of just the current round. In order to do this, I have to simulate the descendant rounds and calculate the utilities of the value cards. However, due to the intricate dynamics of Goofspiel, I here only consider two rounds in a row to for simple calculation.

When I am still at the current round with the outcome not yet revealed, I need to know more for calculating the utility of each value card for the next round. Therefore, some **assumptions** would be necessary (*please notice that these assumptions are only valid for the calculation during Monte Carlo simulation*):

- My opponent wins the current round for whatever value card I play at the current round;
- My opponent's remaining hands stay the same (it doesn't get either better or worse off).

We can think of these assumptions like a recourse process, which should be plausible where I stay pessimistic and do not relax when I am in the middle of the game.

Another mystery is that I don't know which point card is going to appear at the next round, and here is where Monte Carlo comes to play. *It is always true that each of the remaining point cards shows up with the same probability at the next round*. Hence we can use uniform (0~1) distribution to determine which in each iteration during Monte Carlo simulation. In other words, one iteration is one realization of the next round with different possible point cards.

In each iteration, summation of the utilities at the next round is calculated **for each value card available at the current round** (*eliminate one value card one time from hands when its corresponding summation of the next round is being calculated*).

$$\sum_{i \in cur.hds, i \neq j} f^{s_w}(v_i^k, p_i^k \mid v_j^{k-1}, p_j^{k-1})$$

Easily, we can get the average utility at the next round **for each value card available at the current round** in each iteration. Then after many times of iterations (say 10000 times), we will be able to acquire many (10000) average utilities **for each value card available at the current round**. Summing them up and averaging them by iterations (10000), I obtain the "**future utility"** of a currently available value card (which means if I play this value card I will have that much "future utility" for the next round).

As the calculation is nested and quite cumbersome so I will create a **numerical example** to make it clearer to the reads. Consider when it is the 9th round of a game:

i. The upturned point card is "8" and four hidden point cards are "2", "6", "9", "Q", and each of four could be turned up with equal probability .25 at the next round (10th round);
ii. My current value cards (current hands) at this round (9th round) are "1", "7", "8", "10", "J".

I can play any of them at the current round, and the other four will be left for the next round.

In one iteration: a uniform (0~1) random number generated is .61 then "9" is the point card at 10th round. I am going to calculate the "future utilities" for my each current value card "1", "7", "8", "10", "J".

Calculating the "future utility" of "1" implies that "1" is played at 9th round and "7", "8", "10", "J" are left available to play at 10th round. It shouldn't be difficult to calculate the total utilities of "7", "8", "10", "J" for 10th round. Then the average is the "future utility" of "1" at 10th round when the point card is "9". Similarly, I can calculate the "future utilities" for "7", "8", "10", "J" at the 10th round when point card is "9" by doing the same thing **in this iteration**.

Assume there are 10000 iterations. In each iteration, the "future utilities" for "1", "7", "8", "10", "J" are computed. Finally after 10000 iterations, there are 10000 "future utilities" for each. Then we calculate the averages as their ultimate "future utilities" for "1", "7", "8", "10", "J".

Pseudocode:

```
for ( i=1; i<= iterations; i++)
{
        decide point card of next round applying uniform(0~1);
        for (all the value cards available at the current round)
        {
           delete the value card played at the current round from hands;
           calculate the summation of the utilities of the value cards for the next round;
            calculate the average of the summation of the utilities;
            adds up the average.
        }
}
obtain "future utility" for each value card available at the current round;
```

3.4 Algorithm

Step.1: Initialize random number generator, set the number of iterations for Monte Carlo simulation;
Step.2: Update my hands, upturned point card P_i and obtain the total points, average value of my remaining hands as well as my opponent's for the current round, then decide the scenario accordingly;
Step.3: Calculate the utility $f^s(V_i, P_i)$ of each value card of the current round, and run Monte Carlo simulation to obtain the "future utility" for each value card available at the current round. Plus these two utilities, then V_i with the highest total gets to play;
Step.4: If it is round 13 in a game, then the game is over, otherwise come back to Step.2 and repeat;

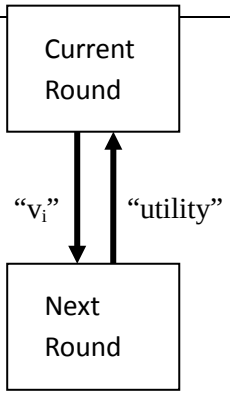


Figure 1 Monte Carlo demonstration

3.5 With carryover

I am borrowing the algorithm from above. When carryover is allowed, we expect no big different algorithm as the utility function I come up with in the last section is still capable to represent the player behavior patterns under this circumstance:

- If points for one round are greater than 13, both players will high likely fight for this round; multiplier function (1) is decreasing from "13" to "1", which is the same idea.
- Otherwise, points for one round are less than 13, and then it is just like a regular round.

Besides, it is worth mentioning that the probability that one round is tied is not going to be high unless both player are using the same algorithm. I concern the stability if I drastically alter my strategy. So basically what I do is keep the algorithm.

3.6 Weakness of this algorithm

Notice that this utility function is made up personally under strong assumptions and lacks of scientific proof. Hence, it may not be well-defined. Besides, I am only simulating two rounds in row, which makes decision a bit "short-sighted". If I could simulate the game covering more rounds (longer future) the decision should then be better and more reliable.

4. Maximizing points of earning(Algorithm.2)

4.1 Statistical behavior database

In order to maximize points of earning, my goal has been tightened from just winning as many games as possible to earning as many points as I can. Since the context here is a lot more distinct than the previous one, something must be paid attention to as one is trying to develop his/her algorithm.

I will treat as a long series of rounds instead of game by game different from the previous algorithm where it is more based on two rounds in a row. Consequently I will look at the game "globally".

Actually, I am doing so by learning my opponent's behavior in a long run through constructing

a **statistical database** which record the average of my opponent's value card played for each upturned card (here 1, 2, 3…, 13). This may take a while to build up a reliable database, say 10 or more games. The structure of database consists of different averages:

$$A = \{A_1, A_2, ...A_{13}\};$$

Let's assume I have an A={1.4, 2.6, 4.9, 5.3, **1.1**, 6.4, 7.6, 9, 7.4, 12, 11.1, **9.3**, 13} after 10 games. Then for the next game I know which value card to play for a particular point card, say point card is Queen, I can try to play 10 > 9.3 to beat my opponent according to the history data. Or if point card is 5, I can try to play 2 > 1.1 to win this round.

4.2 Stochastic model

As long as my database is successfully created, every time a card is turned up from the third suit (pile). I can try to find a card to play which "matches" my opponent's play. However, it would be too easy to predict if my opponent has the similar algorithm. So in order to disturb my opponent, it'd be better to incorporate some randomness into my algorithm.

Step.1: look up the database and get the corresponding entry A_{pi} in terms of P_i, find the value cards at hand which are not played yet and separate them into two **class G, L**: class G is greater A_{pi} and Class L is less $A_{pi.}$
Step.2: if class G is not empty, then generate a random number from uniform(0~1), if it is less than .5 we take the first value card in class G. Otherwise we generate another random number from uniform(0~1), return the second value card if it is less than .5. Repeat until we obtain a value card to return. Come back to Step.1.
Step.3: if class G is empty, the algorithm returns the least value card from Class L as I don't want to waste my value cards in a point card which I can barely win.
This can be viewed as a Markov chain with a single sink node.

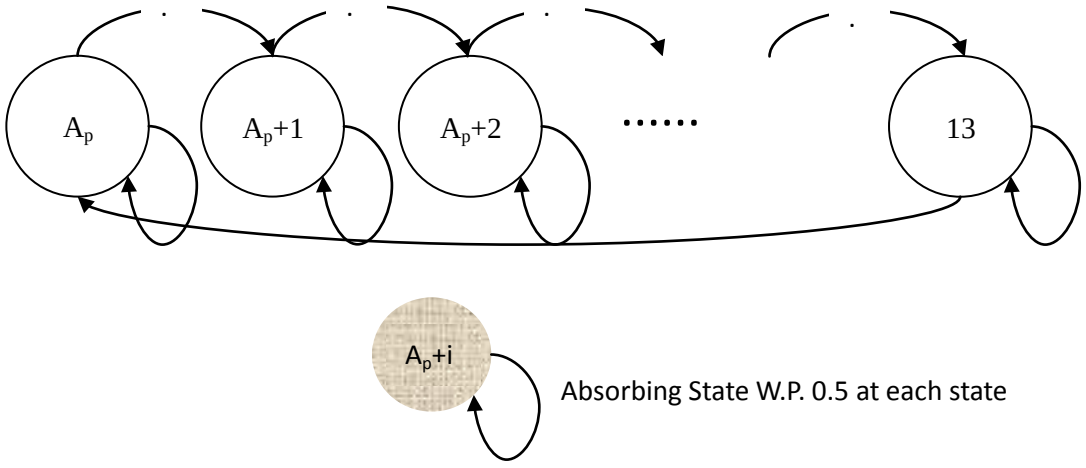


Figure 2 Markov Chain

Pseudocode (stochastic part):

```
while(1)
{
```

```
    for (Class G)
    {
      generate a random number from Unifor(0~1);
      if rand()<0.5
        return the corresponding value card;
      }
}
```

4.3 With carryover

I can easily adjust my algorithm from above by doing only two things to Algorithm.2.

First, I expand my database from containing 13 entries to 25 entries, so I am keeping track of my opponent's value card for one round when the point card is from 1 to 25.

$$A = \{A_1, A_2, ...A_{13,...}, A_{25}\}$$

Then, we follow the same stochastic algorithm to find a value card to return. However, if point of one round is greater than 25 resulting from at least three tying rounds in a row, I am going to return my biggest value card.

4.4 Weakness of this algorithm

A might sometimes not be a good indicator of my opponent's response behavior as it's possible that one average is high just because for some rounds my opponents accidentally played some great value cards but most of the time he play small value cards. A possible improvement is to use dominance which keeps track more detailed information of my opponent's play style.

5. Performance test

I run my algorithms against two typical strategies and the result is listed in the table below.

Table 1

	Matching Upturned Card	Random strategy
maximizing # of winning (NCO)	even	Better
maximizing # of winning (WCO)	even	Better
maximizing points of earning (NCO)	better	little better
maximizing points of earning (WCO)	NA	Even

The results shall be very self-reflected, as it is obvious that "Matching Upturned Card" will be beaten by my algorithm if my algorithm tries to respond its playing behavior. And "Random strategy" is worse when my algorithm tries to aggressive and conservative at the same time.

6. Conclusion

Algorithm.1 is better when my opponent uses stochastic strategy (anti-randomness) and Algorithm.2 is better when my opponent uses deterministic strategy (anti-deterministic).

7. References

[1] Ross, S.M. (1971). Goofspiel: The Game of Pure Strategy. Journal of Applied Probability. 8,.

[2] Moshe Dror, Timothy Chan. (2004) Goofspiel

[3] Thomas S. Ferguson, Costis Melodidakis, Games with Finite Resources

Strategy 0209

Goofspiel Strategies

Introduction

Goofspiel, the game of pure strategies , is two player card game. Each player takes a complete suit of cards. Another suit is shuffled and placed face down in the middle of table (deck), the last suit is discarded. The basic rules of this game are as follow:

- The cards in the middle of the table are turned up one at a time.
- Both players make secret bids and show one card at the same time.
- The player showing the highest value of card wins the value of upturned card.
- Once a card has been played from a player's hand it is discarded
- If both players select the same card from their hand, nobody wins the face up card, there are two ways to deal with the upturned card in our game
    - It is discarded.
    - Its value is carried over to next rounds till one player wins
- The winner is the player who has the most points once all the cards on the table have been played.
- The cards are valued from low to high as ace=1, 2, 3, ..., 10, jack=11, queen=12, and king=13

In our project, we will need different strategies to maximize either the total winning points or total number of winning.

Analysis

Goofspiel is described in Luce and Raiffa (1957) and also studied by other researchers, such as Ross (1971), Dror, M. (1989), Rhoads and Bartholdi(2012) and etc. There are no optimal strategies for this game when both players aim to optimize their winning due to the unpredictable strategies of each player. However, as stated by Ross (1971), the best strategy is to play card that matches the upturned card when opponent chooses card randomly.

To analyze this game, the number of distinct ways that the middle cards could be ordered and the number of distinct betting sequences for each player are all N factorial. Therefore, the number of possible ways of playing this game is $f(N) = (N!)^3$, which exceeds the computational possibilities. Assuming both players adopted an optimal solution that could maximize the winning profits, Rhoads and Bartholdi(2012) described a method to break the game down to small games and use recursive rule to express the value of a game as a function of the values of smaller games, they also proposed to solve the computational complexity issue by using a bottom-up approach storing the values of the smaller game and use these stored values when computing the larger games. However, the memory requirement still hindered the replication of such method on our local machine.

To come up with optimal strategies, we need to anticipate how our components will play and take the appropriate responses. To make it more clear, the notions involved in my strategies are listed below:

- X- the set of cards in my hand
- Xi- the card played by player in play i
- Y- the set of cards in opponent hand
- Yi- the card played by opponent in play i
- P- the set of cards in the deck
- Pi- the card upturned in the deck in play i
- Vx,Vy,Vp –the value of my card, opponent card and upturned card respectively
- m==m%13 means that m is remains when m is divided by 13

Assuming my opponents do not have the computational power to derive optimal strategies by implementing the method proposed above, I consider the possible strategies played by my opponents might consist of following options:

1. opponents might choose cards randomly disregarding to the upturned card (rarely)
2. opponents might choose cards in a sequence that matches the upturned card (Vy==Vp) (he or she will consider I select my cards randomly)
3. opponents might give up the first card and choose cards in a sequence that the value of each card will be 1 greater than the value of upturned card (Vy=Vp+1)(he or she will consider that I define my strategies based on option 2)
4. opponents might choose shifting strategies. This strategy might show the value of selected cards is certain less or more than the value of upturned card in the middle. Vy=Vp+k, kϵ{2..13}, if Vk≥13, Vk==Vk%13 (remains after Vk is divided by 13)
5. My opponents might choose cards based on the recommendation by Rhoads and Bartholdi(2012)

Corresponding Strategies

The corresponding strategies are very straightforward. First, to identify what options opponent is chosen. Second, a strategy will be chosen to respond to opponent.

Operational Details

Stage 1.

In the first and second card play, the cards are chosen randomly. The purpose of this stage is to identify what possible strategy opponents are implemented.

Stage 2

Decision is made about what strategy is played by opponent by observing the first two rounds.

1. If Y1-P1==Y2-P2=m (m==m%13),

   From play 3, a card Xi=Pi+m+1 will be played

   - In case that required cards were played in play 1 and/or 2, cards that should have been played will be selected instead.
   - In case Xi>13, value of Xi will be redefined as the mode of 13 by using following equation: Xi= Xi%13
2. If Y1-P1≠Y2-P2,
   From play 3, assuming that opponents are using random strategy, cards chosen will match the upturned card from the middle of deck.
   - In case that required cards were played in play 1 and/or 2, cards that should have been played will be selected instead.
3. Different checking processes are employed based on different requirement (*Maximizing # of Winning* or *Maximizing Winning Points*) and different situations (carryover or non-carryover)

Checking Process

There are four different checking processes that are implemented for different strategies.

1. During the process, a checking activity is performed during every play.
   If sum(upturned card + player's points at this round) ≥46,
   and the largest card in player hand is larger than any of card in opponent hand, player will play his largest card.
2. During the process, a checking activity is performed during every play.
   First, cards in both players and deck are ranked. If half of cards in player hand are larger than corresponding ranked cards in opponent hand, when cards in deck is upturned, the player will play corresponding ranked card. For example,

Ranking	Player	Opponent	Comparison	Deck
2	X2	Y2	X2>Y2	P2 is upturned, X2 is played

3. During the process, a checking activity is performed during every play. If combined value of upturned card and carried over is larger than the total remaining deck card, and the largest card in player hand is larger than any card in opponent card, the player will play the largest card.
4. During the process, a checking activity similar as the one in checking process 2 is performed for every play. The difference is that when ranking the cards in deck, if there is a carryover value, this value will be added into current upturned card.

Strategies Implementations

Without Carried Over

Maximizing Winning Points

Corresponding strategy and Checking process 1 will be implemented

Maximizing # of Winning

Corresponding strategy and Checking process 2 will be implemented

With Carried Over

Maximizing Winning Points

Corresponding strategy and Checking process 3 will be implemented

Maximizing # of Winning

Corresponding strategy and Checking process 4 will be implemented

Testing and Results

Simulations were run to test corresponding strategies (C) , comparing to random selection strategy(R). The result is shown in following table.

Table 1 Simulation Testing Against Random Selection

Round #	Player Performance (comparing random selection)							
	Maximizing # of Winning (Points)				Maximizing Winning Points (Points)			
	Carried Over		Non Carried Over		Carried Over		Non Carried Over	
	C	R	C	R	C	R	C	R
1	44	47	59	32	70	21	59	14
3	151	122	173	80	174	83	151	81
5	296	183	286	149	314	125	316	94
7	425	259	373	200	452	167	385	146
9	476	373	464	262	559	224	532	177
10	546	394	540	277	646	224	594	202
15	915	450	766	421	932	407	846	356
20	1198	615	1135	524	1292	465	1168	475
Average	59.9	30.75	56.75	26.2	64.6	23.25	58.4	23.75

Discussion

Table 1 shows that all strategies outperformed random strategy. As expected, Maximizing Winning Points with carryover won with the highest average points(64.6) after 20 rounds games.

In both *Maximizing # of Winning* and *Maximizing Winning Points* categories, any strategy with carryover gained more points than corresponding strategy without carryover, as shown in Fig 1. In addition, *Maximizing # of Winning* obtained more points than *Maximizing Winning Points* under both carryover and noncarryover conditions, an example is shown in Fig 2.

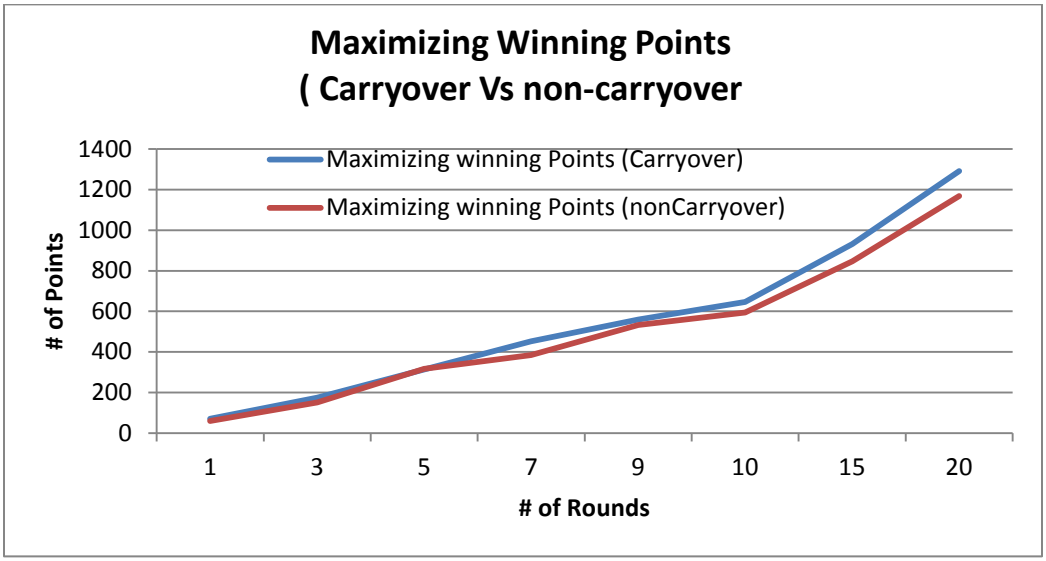


Fig 1. Maximizing Winning Points(carryover Vs non-carryover)

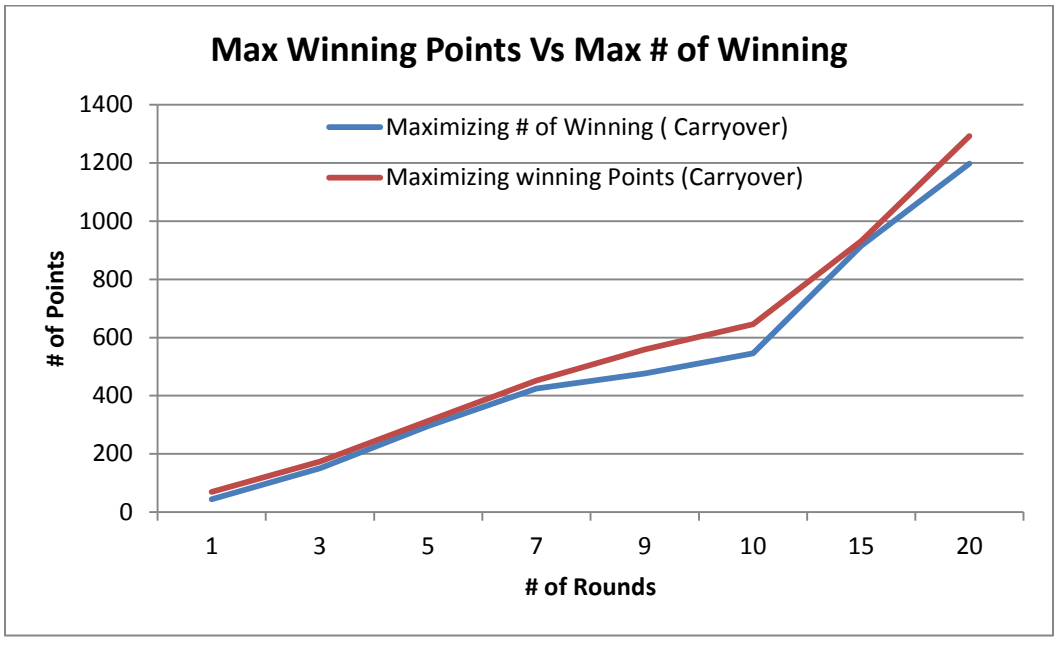


Fig 2. *Maximizing # of Winning* Vs *Maximizing Winning Points*

From all simulations , we can conclude that all proposed strategies performed much better than random strategy , and *Maximizing Winning Points* strategies do gain more points than *Maximizing # of Winning* strategies .

Reference

Luce RD, Raiffa H (1957) Games and Decisions, *Reprinted (1989) by Dover Publications*, Inc., New York.

Rhoads, Glenn C., and Laurent Bartholdi. "The Game of Pure Strategy is solved!." *arXiv preprint arXiv:1202.0695* (2012).

Rhoads ,Glenn C.(2011). Goofspiel. http://mysite.verizon.net/vze16nctz/gops.html

Dror, M. (1989). Simple proof for Goofspiel: the game of pure strategy. *Advances in applied probability*, 21(3), 711-712.

Ross SM (1971) Goofspiel—The game of pure strategy. *Journal of Applied Probability* 8:621-625.

Strategy 0210

Goofspiel Strategy

General Offensive Strategy

I believe that the heart of the Goofspiel problem is predicting what card the opponent will play. If this information were known, it would be trivial to play cards that beat the opponent's cards by one. The challenge of an offensive strategy is to build an algorithm that can predict what the opponent will play, and play a card just high enough to win.

In general, my offensive strategy assumes that my opponent is a rational player who is trying to win through use of some strategy. My strategies perform better against simple strategies (such as an opponent playing the up card + 1) than they do against completely random card selection.

General Defensive Strategy

I employ defensive strategies in order to guard against artificial intelligence algorithms that might be employed by my opponent. Any single deterministic strategy could theoretically be analyzed to discover patterns. A strategy that is predictable is easiest for a complex algorithm to beat because the played cards can be known with some degree of certainty. To guard against predictability of my own plays, I used two main defensive strategies:

1. Incorporate non-deterministic elements into the card selection. For example, if the up card is less than 7, 30% of the time I will play a low card to try to induce my opponent into beating me by several points, thereby giving me an advantage in later rounds. Because this is random, the opponent will never know when I will play a low card.
2. Use multiple strategies. I have several strategies to choose from. The strategies can play different cards for the same scenario, thereby making it harder for my opponent to find patterns in my plays. This would also help to throw off any artificial intelligence algorithms run by opponents since they would not immediately know which strategy I was using in a given round.

The end result of my defensive strategies is that the cards I play for specific up cards is not consistent from round to round, which either renders learning algorithms useless, or less useful at a minimum.

Base Algorithms Run Before Specific Strategies

I used a layered approach to strategies. My base algorithms are universal in that they look for an easy advantage. They work whether the goal is to maximize points or wins, and whether it is carryover or not. I gave each of my algorithms a code name to help keep track of them. After the base strategies run, more specific strategies are run depending on whether it is carryover or not, or if the goal is to maximize points or wins.

- **Guaranteed Good Play**. This algorithm decides whether it is a good idea to play my highest card, lowest card, or if no guaranteed optimal play can be found.
    - If my highest card can be my opponent's highest card, the following situations are analyzed:
        - If the current play value is greater than 13, I play my highest card. This would only happen in a carryover situation.
        - If the current play value is greater than half of the total remaining points, I play my highest card. This is a good strategy because if I only win one more hand in the rest of the round, I would earn the most points by winning the current hand. For example, if 10 points are carried over, consider the following set of remaining up cards:

          $$Carrover\ points = 10$$
          $$Up\ Cards = \{4,5\}$$

          If either the 4 or 5 were the current up card, it would be good to win the hand since the total value is larger than the total remaining points in the round.
        - If the up card is the highest left, I play my highest card. It would not be smart to save my highest card in order to win more points since, in this situation, the up card has the single most points that I can win.
          For example, consider a given set of remaining up cards:

          $$Up\ Cards = \{1,2,3,10\}$$

          If I could win the 10, that single hand is worth more than the sum of the remaining up cards.
        - If my opponent's highest card is less than the up card, and my highest is greater than or equal than my opponent's, I play my highest card. This prevents me from dumping a low card and letting my opponent win more points than the value of the card he plays. Consider the following cards:

          $$Up\ Cards = \{4,8,\mathbf{13}\}$$
          $$My\ Cards = \{3,9,\mathbf{11}\}$$
          $$His\ Cards = \{4,9,10\}$$

          If the 13 up card were played and I were to dumb a low card, my opponent would win 13 points by only playing 10 points. This strategy ensures that my opponent does not get an easy victory in a hand.
    - If my highest card is lower than my opponent's highest card, the following situations are analyzed:
        - If the current up card is the highest up card left in the deck, I play my lowest card. In this situation, a rational opponent would play his highest card. It is in my best interest not to waste my highest card in a situation where I am very likely to lose. For example, consider the following hands:

          $$Up\ Cards = \{5,10,\mathbf{12}\}$$
          $$My\ Cards = \{\mathbf{4},9,11\}$$
          $$His\ Cards = \{3,10,\mathbf{12}\}$$

          If the 12 up card were played, my strategy would play a 4, and I would expect

my opponent to play his 12. This would allow me to play my 11 and win the 10 up card in a future round.

- **Carryover to the Death**. This algorithm checks to see if the current play is in a high carryover situation (> 13 points). Next, it determines if my opponent and I keep playing our highest cards if I eventually win. In this situation, my I will keep playing my highest card until I beat my opponent, hopefully causing him to waste several cards.
    - For example consider the two hands:

$$My\ Cards = \{1,3,\mathbf{7},8,9\}$$
$$His\ Cards = \{1,3,\mathbf{6},8,9\}$$

      In this situation, if we both played our highest cards, we would tie on the 9, tie on the 8, and then I would win on the 7.
- **Opponent Can Beat Me In Carryover At Will**. This strategy is the opposite of Carryover to the Death. It checks to see if my opponent can keep playing his highest card and eventually beat me. If we are in a high value carryover situation and this is true, I will play my lowest card to cut my losses and try to win future hands.
- **Beat the Dump**. This strategy predicts when the opponent is likely to play his lowest card because a victory seemed infeasible. For example, consider the following cards:

$$My\ Cards = \{1,3,5,10\}$$
$$His\ Cards = \{2,4,7,9\}$$

  If the up card was a King, I would clearly have an advantage because I have the highest card. Knowing this, my opponent would likely want to play his 2 rather than waste his 9. With this in mind, it might be beneficial for me to anticipate this dump card. So, rather than playing my highest card, I would play my 3—one card higher than my opponents probably dump card. This would help me save my high cards while still winning points. One danger is that this strategy is too sophisticated, and it might backfire against more naïve players.

Strategy – No Carryover for Points

These strategies are run after the base strategies. Strategy 1 is played most of the time. Strategies 2 and 3 exist partially to throw off opponents that attempt to use learning algorithms to predict my cards. I found that if I played Strategy 2 or 3 alone, I would lose terribly against learning opponents. However, if I played them sparingly while relying on Strategy 1 for most rounds, I would win more points overall.

Strategy 1

This strategy is played 80% of the time. This strategy tries to predict the card that the opponent will play, and either try to beat it by a little bit, or lose by a lot by playing my lowest card.

- Opponent Card Prediction. This algorithm assumes a rational decision making process that a human might use. Initially, I assumed that opponents would play the up card if they had it. However, testing revealed that most computer opponents play much higher. Therefore, I ended up with the following logic:

    - If the opponent has a card 2 greater than the up card in his hand, he will play it. If it is the king or queen, the algorithm will assume that the opponent will play those cards since there are no cards 2 points higher than the king or queen.
    - If the opponent does not have a card 2 higher than the up card in his hand, he will play the next highest, but not more than 4 points greater than the up card.
    - If the opponent does not have the up card or a card within 4 points of it, he will play his lowest card
- My Card Selection.
    - For up cards > 7, my algorithm will play the lowest card in my hand 10% of the time. This should allow many future wins by sacrificing a single card while throwing off round to round card tracking employed by an opponent.
    - For up cards < 7, my algorithm will play the lowest card in my hand 30% of the time. Again, the purpose of doing this is to win more future cards while throwing off round to round card tracking employed by an opponent.
    - I then try to find a card in my hand to play that is 1 more than my opponent's predicted play. If I have to look more than 4 points higher than the up card to find a card to play, instead of overbidding for the card, I just play my lowest card.

Strategy 2

This strategy is played 10% of the time. It simply returns the up card + 1. In the case of the king, it returns 1. This is a deterministic strategy that would play the same cards each time for the same order of up cards. This strategy does not analyze the opponent's card in order to predict his play.

Strategy 3

This strategy is played 10% of the time. It returns a random card between 0 and 2 cards higher than the up card. If all three of those cards have already been played, it will play the lowest card in the hand. This should create reasonable plays while defending against artificial intelligence algorithms that try to predict what I will play. This strategy sacrifices the king and the queen in order to win more future plays.

Strategy – Carryover for Points

These strategies are run after the base strategies. My strategies build off of my strategies without carryover. If there was no tie in the last play, the strategies are carried out exactly the same. However, if there is a tie, I modified the strategies in the following ways:

Strategy 1

- Opponent Card Prediction. If the value of the current play plus the amount of carryover is greater than 13, the opponent is predicted to bid 3 points higher than the up card (unless that puts the opponent over 13, in which case 13 becomes the predicted play).
- My Card Selection. If the opponent and I are in a high value carryover situation (> 13 points), and I can eventually win a future play if we continue to play our highest cards, I will play my highest card.

Strategy 2

No change was made to this algorithm. Any efficiencies that could be gained due to carryover should be gained in the base strategies. Keeping my algorithm consistent is also a defensive measure in that it allows my Strategy 1 to gain more points against learning algorithms.

Strategy 3

No change was made to this algorithm. Any efficiencies that could be gained due to carryover should be gained in the base strategies. Keeping my algorithm consistent is also a defensive measure in that it allows my Strategy 1 to gain more points against learning algorithms.

Strategy – No Carryover for Wins

Initially, I tried creating new strategies just to maximize wins. However, these strategies were not successful. Instead, I made the following modifications to my existing strategies to maximize points.

Strategy 1

My algorithm checks for certain conditions and plays the corresponding card.

- I check to see how many more points are needed for me to win. If the current up card is less than what I need to win, and I have the highest card, I play my highest card.

Strategy 2

This strategy completely replaces the strategy for maximizing points. This strategy is played 10% of the time. It is included to try to throw off any artificial intelligence by the opponent. This strategy tries to win the cards 1-10 by bidding 3 more than the up card value for these cards. This strategy plays 1, 2, and 3 for the up cards 11, 12, and 13. Winning the cards 1-10 earns 55 points, which secures the win. The hope is that other players will try to win the high value cards and end up allowing the low value cards to be captured. This is a simple strategy, but it worked well when I played it against real human beings.

Strategy – Carryover for Wins

These strategies are run after the base strategies.

Strategy 1

My strategy is similar to the no carryover game with the following additions:

- If I can eventually beat my opponent in carryover if we both play our highest cards, and the current play value is greater than 13 or more than half of the remaining points, I will always return my highest card.
- In a carryover situation, I am willing to play up to four points higher in order to win.

Strategy 2

No change was made to this algorithm. In my estimation, overbidding for each up card by 3 points would be enough to win by a decisive margin in most situations. In the case of a tie, I could sacrifice any single

card and still have enough points left to win due to the reduction in the total points needed to win the round. Also, any easy wins that might exist due to a carryover situation would be captured in my base strategies.

Appendix B

Tables

Table 1: Score(P1-P2) - Carryover maximizing points - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	307	-	-	-	-	-	-	-	-	-	-
0205	102	188	-	-	-	-	-	-	-	-	-
0203	42	-126	-134	-	-	-	-	-	-	-	-
0210	68	-136	-27	10	-	-	-	-	-	-	-
0207	-20	154	24	-244	2	-	-	-	-	-	-
0206	186	-254	-2	66	-247	198	-	-	-	-	-
0202	136	83	-223	-17	-2	16	72	-	-	-	-
0204	212	-207	-207	-55	-142	15	-144	-86	-	-	-
0208	195	190	-19	-54	65	-22	243	204	302	-	-
0201	260	358	51	136	166	-49	256	132	213	93	-

Table 2: Wins(P1-P2) - Carryover maximizing points - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	10	-	-	-	-	-	-	-	-	-	-
0205	4	6	-	-	-	-	-	-	-	-	-
0203	0	-4	-10	-	-	-	-	-	-	-	-
0210	2	-6	0	0	-	-	-	-	-	-	-
0207	-2	10	0	-6	2	-	-	-	-	-	-
0206	10	-10	0	2	-6	8	-	-	-	-	-
0202	4	5	-4	-2	-2	2	3	-	-	-	-
0204	8	-8	-6	-2	-8	0	-8	-6	-	-	-
0208	6	8	2	-6	0	-4	8	8	10	-	-
0201	10	10	2	8	6	-6	10	6	8	6	-

Table 3: Score(P1-P2) - Carryover maximizing points - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	405	-	-	-	-	-	-	-	-	-	-
0205	346	326	-	-	-	-	-	-	-	-	-
0203	162	120	-12	-	-	-	-	-	-	-	-
0210	280	-156	-228	37	-	-	-	-	-	-	-
0207	16	268	-113	-446	-151	-	-	-	-	-	-
0206	474	-221	66	67	-398	358	-	-	-	-	-
0202	425	218	-65	-182	-15	-34	268	-	-	-	-
0204	382	-445	-216	-22	-272	64	-255	457	-	-	-
0208	399	58	-36	148	176	-245	531	185	367	-	-
0201	305	725	56	193	389	-70	404	182	299	-42	-

Table 4: Wins(P1-P2) - Carryover maximizing points - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	12	-	-	-	-	-	-	-	-	-	-
0205	16	10	-	-	-	-	-	-	-	-	-
0203	6	2	-2	-	-	-	-	-	-	-	-
0210	8	-2	-10	0	-	-	-	-	-	-	-
0207	-2	20	-6	-16	-10	-	-	-	-	-	-
0206	14	-8	2	4	-10	20	-	-	-	-	-
0202	12	12	-5	-10	-2	-4	10	-	-	-	-
0204	16	-18	-12	0	-10	4	-6	12	-	-	-
0208	14	6	-3	6	8	-14	13	8	14	-	-
0201	10	20	4	12	18	-10	12	10	18	0	-

Table 5: Score(P1-P2) - Carryover maximizing points - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	794	-	-	-	-	-	-	-	-	-	-
0205	442	580	-	-	-	-	-	-	-	-	-
0203	241	-150	-6	-	-	-	-	-	-	-	-
0210	483	-239	-200	180	-	-	-	-	-	-	-
0207	-175	494	-345	-268	-83	-	-	-	-	-	-
0206	356	-510	-343	221	-608	265	-	-	-	-	-
0202	617	172	-110	-239	-86	-58	684	-	-	-	-
0204	1009	-713	-427	-223	-391	121	-338	-397	-	-	-
0208	448	59	-347	132	332	-303	783	295	531	-	-
0201	653	1120	-99	354	477	-266	684	170	552	-224	-

Table 6: Wins(P1-P2) - Carryover maximizing points - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	28	-	-	-	-	-	-	-	-	-	-
0205	20	16	-	-	-	-	-	-	-	-	-
0203	10	-6	-6	-	-	-	-	-	-	-	-
0210	20	-6	-6	7	-	-	-	-	-	-	-
0207	-10	30	-10	-16	-10	-	-	-	-	-	-
0206	14	-22	-6	10	-16	13	-	-	-	-	-
0202	16	10	2	-8	-4	-4	17	-	-	-	-
0204	28	-28	-20	-10	-18	10	-14	-12	-	-	-
0208	14	-4	-10	4	14	-20	24	11	18	-	-
0201	26	30	-2	18	16	-12	28	10	20	-12	-

Table 7: Score(P1-P2) - Carryover maximizing points - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	1240	-	-	-	-	-	-	-	-	-	-
0205	799	894	-	-	-	-	-	-	-	-	-
0203	414	-350	-85	-	-	-	-	-	-	-	-
0210	426	-520	-187	154	-	-	-	-	-	-	-
0207	12	827	-687	-833	-96	-	-	-	-	-	-
0206	522	-733	-718	-11	-973	651	-	-	-	-	-
0202	1138	375	-522	-418	46	7	691	-	-	-	-
0204	1431	-1226	-520	-290	-655	373	-808	-1094	-	-	-
0208	958	412	-134	262	426	-548	819	437	1014	-	-
0201	916	1545	-194	347	860	-433	1125	377	804	28	-

Table 8: Wins(P1-P2) - Carryover maximizing points - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	38	-	-	-	-	-	-	-	-	-	-
0205	24	28	-	-	-	-	-	-	-	-	-
0203	12	-20	-4	-	-	-	-	-	-	-	-
0210	22	-10	-5	4	-	-	-	-	-	-	-
0207	-9	50	-28	-24	-6	-	-	-	-	-	-
0206	14	-32	-20	6	-24	36	-	-	-	-	-
0202	38	23	-17	-18	0	1	19	-	-	-	-
0204	46	-44	-26	-10	-22	16	-28	-38	-	-	-
0208	30	14	-2	12	14	-28	26	15	30	-	-
0201	34	48	-10	20	32	-34	40	20	38	0	-

Table 9: Score(P1-P2) - Carryover maximizing wins - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	224	-	-	-	-	-	-	-	-	-	-
0205	112	116	-	-	-	-	-	-	-	-	-
0203	-90	-109	-33	-	-	-	-	-	-	-	-
0210	140	-71	-34	130	-	-	-	-	-	-	-
0207	-22	153	-29	-140	-54	-	-	-	-	-	-
0206	106	-94	-59	171	-177	101	-	-	-	-	-
0202	208	61	-199	-154	48	-10	150	-	-	-	-
0204	196	-246	-298	-34	-236	157	-118	-144	-	-	-
0208	135	106	41	116	82	-170	244	224	238	-	-
0201	164	348	-29	84	269	57	300	71	170	-32	-

Table 10: Wins(P1-P2) - Carryover maximizing wins - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	10	-	-	-	-	-	-	-	-	-	-
0205	4	4	-	-	-	-	-	-	-	-	-
0203	0	-6	-2	-	-	-	-	-	-	-	-
0210	0	-2	2	8	-	-	-	-	-	-	-
0207	-2	10	0	-8	-4	-	-	-	-	-	-
0206	4	-2	-3	4	-6	2	-	-	-	-	-
0202	8	2	-8	-9	0	0	3	-	-	-	-
0204	6	-8	-10	-2	-8	8	-4	-6	-	-	-
0208	6	4	1	4	4	-10	7	9	10	-	-
0201	6	10	-4	6	8	-2	10	4	6	0	-

Table 11: Score(P1-P2) - Carryover maximizing wins - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	478	-	-	-	-	-	-	-	-	-	-
0205	388	155	-	-	-	-	-	-	-	-	-
0203	43	-209	-402	-	-	-	-	-	-	-	-
0210	374	-189	-136	168	-	-	-	-	-	-	-
0207	-8	342	-251	-298	-5	-	-	-	-	-	-
0206	133	-49	-236	158	-396	14	-	-	-	-	-
0202	347	128	-162	-260	-48	77	207	-	-	-	-
0204	592	-469	-372	-144	-351	102	-212	-421	-	-	-
0208	317	239	-213	354	350	-274	66	366	314	-	-
0201	298	759	44	114	619	-185	408	75	372	0	-

Table 12: Wins(P1-P2) - Carryover maximizing wins - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	12	-	-	-	-	-	-	-	-	-	-
0205	14	4	-	-	-	-	-	-	-	-	-
0203	0	-12	-14	-	-	-	-	-	-	-	-
0210	12	-8	-6	8	-	-	-	-	-	-	-
0207	-2	20	-10	-8	-2	-	-	-	-	-	-
0206	8	-2	-14	8	-12	-2	-	-	-	-	-
0202	12	6	-3	-10	-6	2	5	-	-	-	-
0204	20	-12	-10	-6	-10	6	-2	-14	-	-	-
0208	12	8	-1	14	14	-20	7	10	12	-	-
0201	12	20	0	6	20	-13	16	4	14	0	-

Table 13: Score(P1-P2) - Carryover maximizing wins - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	867	-	-	-	-	-	-	-	-	-	-
0205	492	74	-	-	-	-	-	-	-	-	-
0203	216	-311	-429	-	-	-	-	-	-	-	-
0210	357	-512	-19	323	-	-	-	-	-	-	-
0207	-94	471	-294	-679	-128	-	-	-	-	-	-
0206	356	-314	-353	560	-589	166	-	-	-	-	-
0202	397	91	-272	-387	188	-161	309	-	-	-	-
0204	149	-812	-553	-167	-314	12	-147	-466	-	-	-
0208	708	395	-201	472	413	-235	333	161	517	-	-
0201	429	1194	-121	504	454	-241	661	129	525	-100	-

Table 14: Wins(P1-P2) - Carryover maximizing wins - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	20	-	-	-	-	-	-	-	-	-	-
0205	16	4	-	-	-	-	-	-	-	-	-
0203	0	-18	-16	-	-	-	-	-	-	-	-
0210	16	-18	2	18	-	-	-	-	-	-	-
0207	-6	30	-12	-23	-6	-	-	-	-	-	-
0206	14	-8	-8	20	-20	10	-	-	-	-	-
0202	16	2	-12	-21	4	-10	6	-	-	-	-
0204	6	-30	-20	-6	-10	14	-10	-14	-	-	-
0208	24	12	-10	16	18	-10	16	8	14	-	-
0201	14	28	-8	20	16	-18	24	10	20	-2	-

Table 15: Score(P1-P2) - Carryover maximizing wins - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	1269	-	-	-	-	-	-	-	-	-	-
0205	860	366	-	-	-	-	-	-	-	-	-
0203	209	-363	-801	-	-	-	-	-	-	-	-
0210	434	-470	-102	338	-	-	-	-	-	-	-
0207	-115	850	-326	-992	-162	-	-	-	-	-	-
0206	394	-479	38	684	-955	452	-	-	-	-	-
0202	762	231	-614	-725	-161	129	138	-	-	-	-
0204	1341	-1210	-866	-80	-352	302	-464	-881	-	-	-
0208	931	512	252	720	731	-608	813	860	937	-	-
0201	504	1739	-138	688	695	-392	1066	253	1118	-276	-

Table 16: Wins(P1-P2) - Carryover maximizing wins - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	42	-	-	-	-	-	-	-	-	-	-
0205	32	7	-	-	-	-	-	-	-	-	-
0203	8	-16	-34	-	-	-	-	-	-	-	-
0210	8	-16	-8	8	-	-	-	-	-	-	-
0207	-2	50	-14	-36	-6	-	-	-	-	-	-
0206	20	-30	7	25	-24	12	-	-	-	-	-
0202	28	14	-21	-32	-4	4	4	-	-	-	-
0204	42	-40	-42	-2	-14	22	-16	-25	-	-	-
0208	26	19	7	24	26	-30	27	21	38	-	-
0201	20	48	-10	32	28	-32	36	8	42	-10	-

Table 17: Score(P1-P2) - No carryover maximizing points - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	192	-	-	-	-	-	-	-	-	-	-
0205	104	477	-	-	-	-	-	-	-	-	-
0203	29	131	-34	-	-	-	-	-	-	-	-
0210	85	-145	-20	8	-	-	-	-	-	-	-
0207	-60	147	-153	46	-62	-	-	-	-	-	-
0206	297	13	-22	44	-33	-169	-	-	-	-	-
0202	239	108	-141	-104	76	-57	10	-	-	-	-
0204	305	-238	-129	-53	-203	49	-248	-70	-	-	-
0208	163	93	-147	-5	167	-139	16	-22	230	-	-
0201	195	297	-97	122	215	-164	97	44	225	-5	-

Table 18: Wins(P1-P2) - No carryover maximizing points - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	10	-	-	-	-	-	-	-	-	-	-
0205	6	8	-	-	-	-	-	-	-	-	-
0203	1	2	-2	-	-	-	-	-	-	-	-
0210	4	-4	-2	-2	-	-	-	-	-	-	-
0207	-6	10	-3	4	-4	-	-	-	-	-	-
0206	10	0	0	4	-4	-10	-	-	-	-	-
0202	10	7	-8	-10	2	-4	0	-	-	-	-
0204	10	-8	-4	-1	-8	4	-8	-2	-	-	-
0208	6	6	-6	0	6	-10	-2	-2	10	-	-
0201	9	10	-8	8	10	-10	5	2	6	0	-

Table 19: Score(P1-P2) - No carryover maximizing points - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	598	-	-	-	-	-	-	-	-	-	-
0205	348	1005	-	-	-	-	-	-	-	-	-
0203	20	99	-120	-	-	-	-	-	-	-	-
0210	343	-269	43	45	-	-	-	-	-	-	-
0207	76	319	-470	-329	-109	-	-	-	-	-	-
0206	408	100	-232	78	14	-347	-	-	-	-	-
0202	370	107	-226	-214	-81	-84	-92	-	-	-	-
0204	499	-538	-408	-166	-332	48	-392	-331	-	-	-
0208	146	167	-409	-11	189	-261	118	-33	416	-	-
0201	377	578	-161	208	226	38	268	80	371	168	-

Table 20: Wins(P1-P2) - No carryover maximizing points - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	16	-	-	-	-	-	-	-	-	-	-
0205	12	20	-	-	-	-	-	-	-	-	-
0203	-2	-2	-10	-	-	-	-	-	-	-	-
0210	14	-14	0	1	-	-	-	-	-	-	-
0207	5	20	-12	-8	-9	-	-	-	-	-	-
0206	16	6	-10	6	4	-20	-	-	-	-	-
0202	11	7	-16	-10	-4	-4	-9	-	-	-	-
0204	16	-18	-13	-8	-10	3	-12	-13	-	-	-
0208	9	10	-18	-2	10	-11	8	2	12	-	-
0201	12	20	-5	12	14	-1	18	2	14	8	-

Table 21: Score(P1-P2) - No carryover maximizing points - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	619	-	-	-	-	-	-	-	-	-	-
0205	587	1605	-	-	-	-	-	-	-	-	-
0203	246	70	-231	-	-	-	-	-	-	-	-
0210	410	-544	-125	175	-	-	-	-	-	-	-
0207	-98	448	-674	-427	-118	-	-	-	-	-	-
0206	536	91	-299	135	192	-523	-	-	-	-	-
0202	589	143	-304	-318	244	-117	-207	-	-	-	-
0204	850	-746	-413	-28	-399	45	-724	-703	-	-	-
0208	524	273	-421	204	274	-512	160	-8	415	-	-
0201	377	709	-443	277	553	-231	268	181	492	41	-

Table 22: Wins(P1-P2) - No carryover maximizing points - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	28	-	-	-	-	-	-	-	-	-	-
0205	22	30	-	-	-	-	-	-	-	-	-
0203	8	-8	-9	-	-	-	-	-	-	-	-
0210	21	-20	-8	3	-	-	-	-	-	-	-
0207	-10	30	-20	-12	-12	-	-	-	-	-	-
0206	21	2	-8	10	5	-30	-	-	-	-	-
0202	21	4	-17	-22	10	-16	-15	-	-	-	-
0204	30	-26	-14	-2	-18	5	-23	-28	-	-	-
0208	23	11	-19	12	17	-28	12	1	18	-	-
0201	20	27	-16	20	19	-24	14	6	23	3	-

Table 23: Score(P1-P2) - No carryover maximizing points - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	1034	-	-	-	-	-	-	-	-	-	-
0205	1028	2580	-	-	-	-	-	-	-	-	-
0203	251	199	-389	-	-	-	-	-	-	-	-
0210	415	-791	-453	272	-	-	-	-	-	-	-
0207	-41	744	-1203	-1150	-152	-	-	-	-	-	-
0206	1177	65	-909	164	310	-808	-	-	-	-	-
0202	861	326	-525	-416	196	-158	-159	-	-	-	-
0204	1500	1319	-987	-239	-599	82	-1034	-969	-	-	-
0208	901	487	-685	-181	595	-722	250	194	738	-	-
0201	830	1505	-571	431	598	-295	516	384	762	68	-

Table 24: Wins(P1-P2) - No carryover maximizing points - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	38	-	-	-	-	-	-	-	-	-	-
0205	42	50	-	-	-	-	-	-	-	-	-
0203	8	-16	-22	-	-	-	-	-	-	-	-
0210	19	-39	-23	9	-	-	-	-	-	-	-
0207	1	50	-32	-30	-12	-	-	-	-	-	-
0206	35	0	-28	15	9	-50	-	-	-	-	-
0202	34	20	-34	-30	0	-15	-9	-	-	-	-
0204	48	46	-31	-6	-24	6	-38	-34	-	-	-
0208	27	28	-25	-7	28	-42	19	12	32	-	-
0201	36	50	-21	32	30	-30	28	16	35	8	-

Table 25: Score(P1-P2) - No carryover maximizing wins - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	344	-	-	-	-	-	-	-	-	-	-
0205	146	335	-	-	-	-	-	-	-	-	-
0203	19	-136	-77	-	-	-	-	-	-	-	-
0210	144	-175	-6	87	-	-	-	-	-	-	-
0207	36	151	-67	-239	-111	-	-	-	-	-	-
0206	138	13	-88	-11	14	-36	-	-	-	-	-
0202	95	21	-139	-123	47	-48	-180	-	-	-	-
0204	308	-217	-114	-64	-157	29	-46	-167	-	-	-
0208	194	48	-135	87	178	-197	62	32	176	-	-
0201	183	331	-138	141	120	54	40	80	286	64	-

Table 26: Wins(P1-P2) - No carryover maximizing wins - 10 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	10	-	-	-	-	-	-	-	-	-	-
0205	4	10	-	-	-	-	-	-	-	-	-
0203	2	-8	-4	-	-	-	-	-	-	-	-
0210	6	-6	0	4	-	-	-	-	-	-	-
0207	5	10	-4	-9	-9	-	-	-	-	-	-
0206	6	0	-3	-3	4	-1	-	-	-	-	-
0202	4	-2	-6	-6	1	-2	-8	-	-	-	-
0204	10	-8	-8	-8	-8	4	-4	-4	-	-	-
0208	6	1	-7	0	8	-8	4	1	7	-	-
0201	5	9	-4	8	6	-1	0	6	10	5	-

Table 27: Score(P1-P2) - No carryover maximizing wins - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	658	-	-	-	-	-	-	-	-	-	-
0205	270	592	-	-	-	-	-	-	-	-	-
0203	184	-163	-243	-	-	-	-	-	-	-	-
0210	351	-220	-44	254	-	-	-	-	-	-	-
0207	-71	336	-297	-482	-17	-	-	-	-	-	-
0206	161	89	130	67	41	-130	-	-	-	-	-
0202	231	137	-225	-272	41	-96	-314	-	-	-	-
0204	499	-465	-482	-8	-248	96	-229	-443	-	-	-
0208	371	113	-234	45	173	-298	-68	110	343	-	-
0201	273	537	-191	192	315	-66	175	239	420	117	-

Table 28: Wins(P1-P2) - No carryover maximizing wins - 20 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	20	-	-	-	-	-	-	-	-	-	-
0205	8	20	-	-	-	-	-	-	-	-	-
0203	10	-12	-12	-	-	-	-	-	-	-	-
0210	14	-14	-1	10	-	-	-	-	-	-	-
0207	-8	20	-14	-16	-7	-	-	-	-	-	-
0206	9	8	6	-2	2	-8	-	-	-	-	-
0202	8	8	-13	-16	2	-8	-18	-	-	-	-
0204	16	-18	-16	0	-14	9	-16	-20	-	-	-
0208	19	8	-14	3	7	-19	-4	8	12	-	-
0201	13	20	-11	10	10	-10	10	14	18	6	-

Table 29: Score(P1-P2) - No carryover maximizing wins - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	906	-	-	-	-	-	-	-	-	-	-
0205	558	902	-	-	-	-	-	-	-	-	-
0203	200	-134	-341	-	-	-	-	-	-	-	-
0210	434	-434	-325	210	-	-	-	-	-	-	-
0207	4	461	-295	-776	-64	-	-	-	-	-	-
0206	288	167	192	76	151	-285	-	-	-	-	-
0202	452	194	-374	-511	-1	-171	-639	-	-	-	-
0204	77	-918	-575	-84	-540	10	-260	-612	-	-	-
0208	653	214	-368	193	223	-457	125	149	464	-	-
0201	596	799	-327	424	518	-198	140	333	499	-47	-

Table 30: Wins(P1-P2) - No carryover maximizing wins - 30 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	28	-	-	-	-	-	-	-	-	-	-
0205	24	30	-	-	-	-	-	-	-	-	-
0203	4	-6	-16	-	-	-	-	-	-	-	-
0210	20	-24	-15	14	-	-	-	-	-	-	-
0207	-7	30	-10	-28	-3	-	-	-	-	-	-
0206	16	13	11	12	12	-19	-	-	-	-	-
0202	15	14	-20	-24	0	-12	-26	-	-	-	-
0204	6	-30	-24	-1	-22	2	-16	-24	-	-	-
0208	24	17	-19	10	14	-22	8	12	22	-	-
0201	26	28	-8	24	19	-11	5	18	20	0	-

Table 31: Score(P1-P2) - No carryover maximizing wins - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	1507	-	-	-	-	-	-	-	-	-	-
0205	708	1626	-	-	-	-	-	-	-	-	-
0203	216	-420	-591	-	-	-	-	-	-	-	-
0210	538	-640	-496	480	-	-	-	-	-	-	-
0207	-27	833	-706	-1333	-149	-	-	-	-	-	-
0206	480	-95	-25	171	254	-252	-	-	-	-	-
0202	778	349	-685	-674	78	-196	-1073	-	-	-	-
0204	1291	-1507	-753	-301	-574	-1	-505	-1068	-	-	-
0208	781	137	-903	402	773	-867	-39	280	1165	-	-
0201	783	1416	-685	743	664	-399	402	227	954	199	-

Table 32: Wins(P1-P2) - No carryover maximizing wins - 50 rounds

P1 \ P2	Random	0209	0205	0203	0210	0207	0206	0202	0204	0208	0201
Random	-	-	-	-	-	-	-	-	-	-	-
0209	48	-	-	-	-	-	-	-	-	-	-
0205	34	50	-	-	-	-	-	-	-	-	-
0203	7	-35	-30	-	-	-	-	-	-	-	-
0210	17	-28	-21	19	-	-	-	-	-	-	-
0207	-5	50	-24	-44	-5	-	-	-	-	-	-
0206	15	5	7	10	8	-15	-	-	-	-	-
0202	34	23	-37	-35	4	-18	-47	-	-	-	-
0204	46	-45	-30	-18	-28	7	-24	-35	-	-	-
0208	30	-4	-43	17	32	-48	-16	19	42	-	-
0201	29	47	-28	37	30	-26	15	21	40	9	-