Project 2: Guess my Hand

Frank Fan Kristofer Bjornsson Larry Davis

October 21, 2024

Contents

Introduction

The *Guess my Hand* project, one in Columbia University's COMS 4444 - Programming and Problem Solving course for the 2024 fall semester, involves simulating two pairs of players playing a simple card game. The two pairs sit opposite

to each other, North & South play East & West, with the players sharing the standard deck of cards amongst themselves.

Players take turns playing one card each per round, with all players privately guessing the remaining cards in their teammate's hand after everyone has played a card for that round. In response, the simulator tells each player how many cards they guessed correctly, which is their *cVal* for the round. North plays first at the start of every round, then East, then South, then West. Then, all players privately send the simulator their guesses, receiving their *cVal*s in return.

For a total of 12 rounds, each player has an opportunity to score $\sum_{i=1}^{13} 13 - i = 78$ points: 12 for the first round, 11 for the second round and so on until the 12th round, where each player makes a guess of one card. After the game's conclusion, the team's score is the sum of both teammates' points, allowing for a total of 156 points maximum per team. The team with the most points wins.

Notably, while teammates can agree on a strategy before hand, no external communication is allowed between players during a game. This means that the only way for a player to convey information to their teammate about their hand is through the card they play.

This report outlines group 3's solution for the project, composed of Frank Fan, Kristofer Bjornsson & Larry Davis.

Strategies

Initial ideas

The ultimate goal of the game is for a team to guess as many cards that their teammate has. On average, a player which guesses at random should guess every fourth card correctly. This is because out of 52 cards in the deck, they get points for guessing 13 cards correctly [1]. A slightly better, but still random player will know not to guess their own cards, increasing their accuracy to every third guessed card being correct. This is as good as a randomly guessing player can get, which sets the lowest standard for every group's performance at $\frac{1}{3} \cdot 156 = 52$ points.

A variety of strategies were explored throughout the first few discussion classes of the project, where the goal was to explore any strategy *better than random* and go from there. Most students were initially drawn towards conveying information about their hand through suits. A player is dealt 13 cards and wants their teammate to guess more than 4 out of the 12 they'll have left

[1] Actually, this isn't quite true for the purpose of simplicity. In reality, on round i, the player gets points for guessing $13 - i$ cards correctly out of a deck of $52 - 4 \cdot i$ remaining cards, as cards played throughout the rounds can't be guessed.

at the round's end. Chances are, they'll likely have more than 4 of a single suit [2].

The initial ideas included a Player mapping suits together so that playing one suit meant that the player had the most of another suit which the played suit mapped to. For example, both players agree that a Diamond maps to Hearts and Hearts map to Diamonds. Player A has 5 Diamonds and one Heart, so they play their Heart. Meanwhile, Player B has 5 Hearts and one Diamond, so they play their Diamond. Seeing Player A play a Heart, they will guess all the Diamonds that haven't been played, while Player B will guess all the Hearts that haven't been played. Both players will get 5 points each, totalling 10 points for the first round. They're off to a good start.

"The linguistic bias"

While the initial strategies, which utilized suits and values, were good starting points, our group was surprised to see how many groups stuck to those well into the project. We felt that most groups fell victim to the discussed *linguistic bias* of the problem, and saw no reason why abiding to the given suits and values would improve our guessing performances over creating our own way of categorizing and grouping cards together.

For example, there is no good reason to abide by the default categorization of *four* suits for this problem. Unless the original creators of the card deck knew something the rest of the world did not, the combination of four suits, each with 13 cards can be thought of as random — at least in terms of optimal grouping.

The group was convinced early on that relying too heavily on the classical categorization of suits and values could likely create unnecessary pitfalls. We wanted our strategy to suit the problem generally: Imagine any n pairs of player dividing an entirely new deck containing d random cards into $2n$ equally large hands, each containing $\frac{d}{2n}$ cards. Our goal was a good general strategy that would perform well on any of these cases, of which, a deck containing $d = 52$ cards with 2 sets of players is just one case.

Best Seed

Our strategy, *best seed*, maps each card to a specific seed value, known to both teammates, which is used to create a random permutation of the entire deck. When deciding which card to play, a player creates the permutation for each card in their hand. For each permutation, they check how many of their n cards appear in the permutation's first n cards, playing the card corresponding to the seed which gave the permutation with most of their cards.

[2] Exact probability, derived in Appendix A, is 0.628.

Figure 1 visualizes the algorithm where a player holds 8 cards and shows the process creating and evaluating permutations, ultimately selecting the card corresponding to the permutation which reveals most of the player's cards — Eight of Clubs revealing 6 cards in this example.

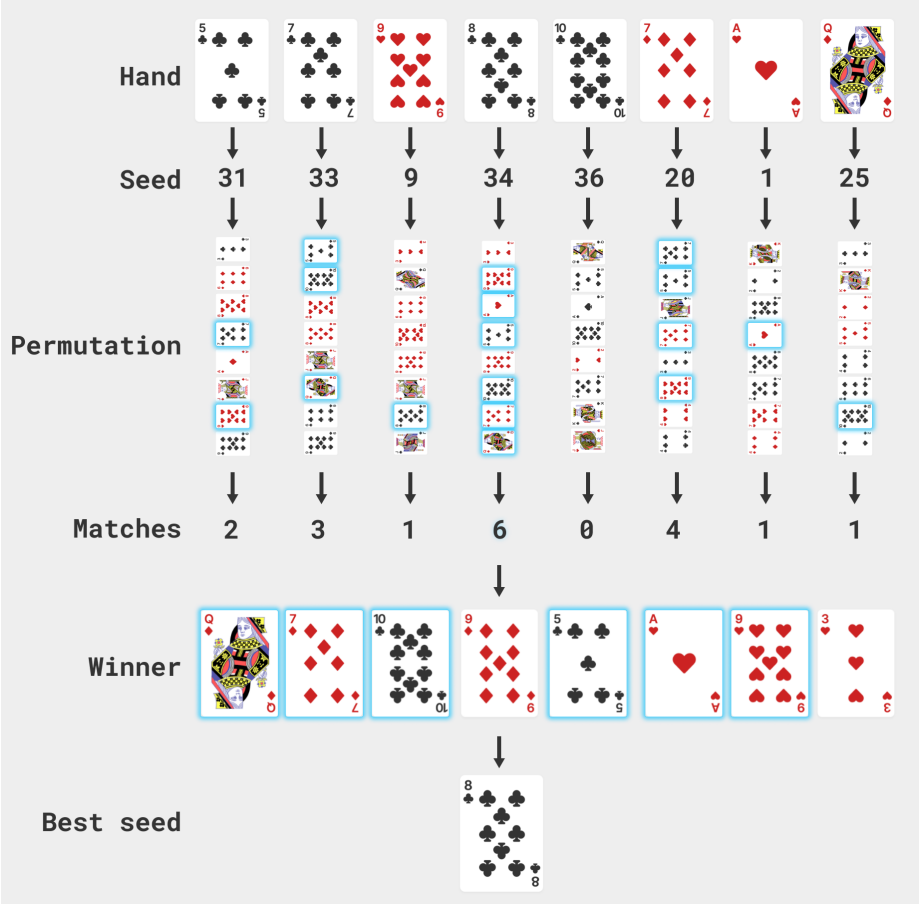

Figure 1: Visualization of selecting best seed

Mapping cards to seeds

As long as both players agree on a consistent mapping and every card corresponds to a unique value, any mapping will work. This requires a deterministic, one-to-one function, receiving a card as input and returning a seed.

In our case, we used the mapping in Listing 1 to convert our cards to seeds. It may look cryptic, but it essentially multiplies the card value with $13 \cdot \text{suit}$, where each suit is mapped from 0 to 3.

```
ALL_VALUES = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q
    ", "K", "A"]
CARD_VALUE = {
    "A": 1,
    "J": 11,
    "Q": 12,
    "K": 13,
}

def get_seed(card: Card):
    return int(CARD_VALUE.get(card.value, card.value)) + 13 * (
        list(card.map.keys()).index(card.suit)
    )
```

Listing 1: Mapping cards to seeds

Creating permutations

With a mapping from card to seed which both players agree on, we generate a shuffled version of the deck based on each card's seed. This shuffle creates a deterministic permutation of the deck. To determine permutations that correspond with each card, we take a subset of the pseudo-randomly ordered deck, containing only the first $13 - i$ cards, where i is the round number.

Measuring permutations

The idea of using shuffles seeded by individual cards serves to allow the players to work with a permutation of possible card orders. For each card's associated permutation, a score is assigned. This score takes into account the number of cards shared between the permutation and the player's hand, and also considers impossible cards— cards that have already been played or the card that will be played to indicate the permutation. The resulting permutation with the highest score is the one whose corresponding card will thus be played.

Playing Unlikely cards

Playing the best seed is naturally only possible in the first round. In the second round, each teammate is bound to playing their second best seed, as their best seed has already been played. Following the strategy throughout the entire 12 rounds would mean by the end, each player would be playing their *worst seed*.

Therefore, this strategy is only effective for a certain amount of rounds. After multiple days of simulations, we saw that an ideal number of *seed rounds* was 2, before a new strategy would supersede. For more information on the number of seed rounds, refer to appendix C.

The superseding strategy has the simple goal of maintaining the effectiveness of the best seed strategy. Namely, the strategy looks at the first two permutations its player indicated to its teammate in the first two rounds, and plays

the cards that did not appear in those permutations. Essentially, on the third round onwards, the player starts playing their most unlikely cards, leaving the likely cards for their teammate to guess.

In order to determine which card in our hand is deemed the most unlikely, the strategy looks at the environment from their teammate's point of view, determining which cards they would be likely to guess. This is done by seeing the initial two permutations generated by the player and estimating a $cProb$, using a corrected $cVal$, considering information the guesser has had access to since those two rounds. The playing player can know the lower bound of the guesser's $cVal$ for the first two rounds, as they know how many of their cards appeared in the permutation. However, the $cVal$ has to be corrected, calculating a new probability after removing cards that have been played since. The $cProb$ is the corrected $cVal$ divided by the amount of total unexposed cards from the permutation. If a card appears in multiple permutations, its final $cProb$ will be an average of all its $cProb$'s.

Our unlikely card strategy has room for exploring performance gains, as we should have a set of a few unlikely cards to choose from. However, in the span of the project, we couldn't find a way to utilize this freedom.

Guessing

In order to interpret the seeded card that the guessing player's partner has presented, the guesser first finds the permutation with which the card is associated. After determining that permutation, the guesser then removes all impossible cards, which are cards that have already been played by any player or that exist in the guesser's current hand. On the first guess, the guesser supplements these removed cards with random possible cards to ensure the guess is always complete. On subsequent turns, the guesser utilizes the $cVal$ of previous guesses to adjust their current guess by calculating a corrected $cVal$, which is computed by taking into account cards that were correctly guessed but have since been played and are thus no longer relevant. The guesser then determines the most likely cards by considering the likelihood of each card in their partner's hand, which is done by combining the number of times each valid card has been included in previous guesses and the corrected $cVal$. After generating the most likely cards, the guesser forms their final guess by selecting a subset from the list corresponding to the maximum size of the current guess.

Tournament Analysis

Overview of Tournament Setup

The class tournament was designed to pit all teams' playing and guessing strategies against each other, to determine how each team fared comparatively. In order to establish a fair testing scenario for all teams, each of the ten teams

was matched up against each of the other ten teams (including itself), with the opportunity to play as both the North-South and the East-West team, which constitutes 100 different pairings. Further, in order to ensure fairness among gameplay (i.e. some teams perform better given a particular arrangement of the cards), 1000 unique arrangements were assigned to each pairing. Thus, the total number of rounds in this tournament was $100 \times 1000 = 100,000$. Over the course of each round, the final scores of both teams were recorded, and a list of all rounds and their respective scores was posted following the conclusion of the tournament.

Exploding opponents

While developing our solution, we found a nasty approach to *explode* our opponents. Essentially, we could alter their environment to force them to crash. The attack could be defended against, and so some group members believed that if brought up in class, this could be allowed. However, when raised in class, the professor banned the approach, considering it to be cheating.

Idea
The attack utilizes the fact that both teams run in the same program, with a shared environment. This means that both teams share the standard library functions and global variables. Now, all groups, including ours, rely on standard library functions for their players to function correctly, but every player is free to overwrite the standard library functions to whatever they want.

This was very useful while comparing our players with different groups. We saw that most groups had placed `print()` statements throughout their player code, which made it difficult for us to test our player against theirs with multiple simulations. Therefore, to *silence* their `print` calls, we opened their player file and overwrote Python's `print` statement to the one in listing 2. This removed all our opponent's `print` statements, allowing us to run our player against other groups and without parsing the results.

```
1 # overwrite Python's global print function
2 def print(*args, **kwargs):
3     return None
```

Listing 2: Overwriting Python's print

To crash other players does not require any other clever tricks. For example, `random.sample` seemed to be very popular, used by groups 1, 4, 5 & 6, although the full list can be found in figure 2. To explode `random.sample`, we must simply overwrite it to crash when called. Listing 3, shows how easy this is.

- Group 1: `random.sample`
- Group 2: `random.choice`

- Group 4: `random.seed, random.sample`
- Group 5: `random.seed, random.sample`
- Group 6: `random.seed, random.sample`
- Group 7: `np.random.choice`
- Group 8: `random.randint, random.choice`
- Group 9: `random.seed, random.shuffle`
- Group 10: `np.random.choice`

Figure 2: Vulnerable groups

```
import random
# Exploding random.sample

def explode(*args, **kwargs):
    raise Exception("CRASH")

# any subsequent calls to random.sample() will be
# equivalent to explode(), causing the caller to crash
random.sample = explode
```

Listing 3: Exploding a caller of `random.sample()`

Prevention

To prevent your player from being attacked by the method explained above, you must create a local copy of the global methods your player uses on initialization. Then, throughout the game, use the local copies of these methods. As you create your local copies on initialization, your opponent has not had a chance to play and run the malicious code. This assumes that groups are not allowed to modify the simulator's initialization phase, which we will agree *is* cheating.

Tournament Results

In the last class period before the final player submission, each player was paired up against one opposing player, with the average and standard deviation displayed after all rounds between two players had been completed. From this "mini precursor tournament", our player seemed to land in approximately third place out of all ten players, based on average score. As the group with the highest average below the two groups that performed in the mid nineties, we felt we were leading the second pack of strategies. Following the official tournament and the posting of the final scores, it seems that some groups were able to improve their strategies before the final player submission, surpassing our final submission and landing us slightly below what we had expected while awaiting the tournament results.

In fact, throughout the course of this project, we generated a strategy early on that produced a relatively high mean score compared to the other groups. However, progress on our strategy plateaued as we tried to determine ways to optimize it. Some of the techniques we attempted to implement did not work as predicted, which is further discussed later on in this report, and ultimately, we were only able to come up with small improvements that would only marginally increase our score. Thus, we weren't surprised to find our final performance middling in the scope of all the groups, in spite of the fact that we had consistently performed better in the preliminary trials during the discussion periods of this project.

Comparison with other players

In analyzing the total wins and average scores of all groups, our player seemed to rank fifth out of ten players. The top three players all averaged around 95 cards each, with fourth place averaging around 92. Our group averaged approximately 89 cards, with sixth through ninth place averaging between about 87 and 88 cards, and finally tenth place with an average of 82 cards. Figure 3 shows the range of mean scores with their standard deviations among the ten players.

In comparing the strategies of each group, cases can be made for both marked similarities and differences. However, the performance of each group's implementation speaks for itself, so let us analyze what may have contributed to some groups' successes and other groups' pitfalls. Of the groups who performed better than us, groups 1 and 9 (who placed second and third, respectively) both utilized a "fake suit" system, where cards were sorted into groups after the initial distribution of cards. The upside of this strategy is the avoidance of the previously discussed "linguistic bias", and it instead redefines a suitable language and tokenizes cards based on the initial card arrangement. Since this strategy provided both of these groups with a mean score of around 95 cards, it is possible that a proper implementation of this strategy might prove to produce a higher score than some of the strategies employed by other groups, such as our current seed-selection strategy.

Another common strategy was the min-max strategy, where players would oscillate between playing the minimum and the maximum card in their hand, with the goal being to bound their partner's guesses to a particular range. Three groups seemed to implement this: group 6 (who performed better than us, with a mean of 92 cards), and groups 7 and 10 (who performed slightly worse than us, with respective means of 88 and 87 cards). However, each group's implementation of such a strategy differed. Groups 6 and 7 both utilized a "greedy min-max" strategy for a set number of rounds, which dynamically adjusts the range of valid cards by considering the difference between the cards current value and the min/max range. Group 10 utilizes a wrap-around min-max strategy, which intends to bound the cards more tightly without adhering to the upper and lower edges set by the linguistic bias. This strategy ended up putting these

three groups around the middle of the class, but since there is a decent variance between their performances, the implementation of such a strategy may determine its comparative efficacy, particularly compared with group 3's strategy.

Group 4 executed a technique similar to ours by employing permutations of unguessed cards, which mirrors our initial seed generation style. However, unlike our style, they follow this seed technique by then playing the lowest or highest card for a set number of turns based on the turn number, and end the game during the last few turns by generating other permutations, and playing the card corresponding to the permutation with the least or most similarity depending on the turn number. Assuming that our two initial seed strategies are similar enough to generate comparable results, it seems that group 3's later strategy of playing the unlikeliest card was ultimately more effective in generating points in the later rounds. However, group 4's creation of new seeds in the later rounds is something that our group had considered using to optimize our current strategy, so it may be a viable option to employ for a future iteration of this strategy.

The range of standard distributions may further provide insight into the efficacy of each group's strategy. Standard deviations for each group can also be seen in Figure 3. Seven groups achieved a standard deviation between 7 and 8 cards, while three groups achieved a standard deviation above 10. These latter three, groups 2, 7, and 10, were ranked among the bottom four for the overall tournament. Conversely, the two lowest standard deviations were achieved by groups 1 and 5, who ranked as the top two scoring groups for the overall tournament. Our group ended up with the sixth lowest standard deviation, merely 0.872 points behind group 1's lowest standard deviation. While these standard deviations are too close in value to definitively determine a direct correlation to player efficacy, a general trend still exists between performance and variance, which suggests that better strategies may often tend to produce more consistent results.

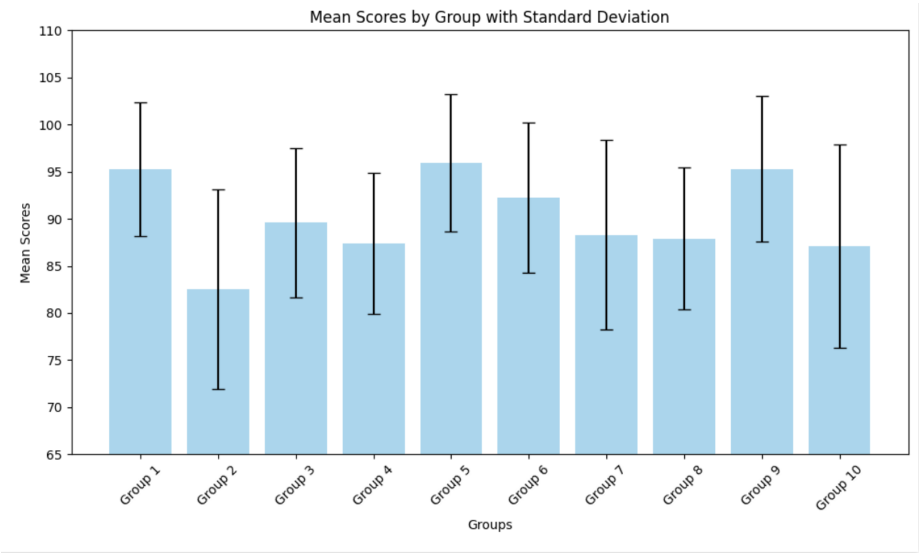

Figure 3: The means and standard deviations of each group

Tournament Discussion

The element of luck was certainly a small factor in performance on a round-by-round basis. While the design of the tournament was intended to reduce bias by having all player-pairs run 1000 card arrangements, there still exist fortunate and unfortunate arrangements when comparing individual rounds. For example, the arrangement generated by number 350 (seed 350, but here "seed" refers to the seed for running the simulation) produced some of the worst scores with our strategy, including the lowest achieved score in all 19,000 of our trials (62 cards). In fact, excluding us playing against ourselves, we only won one instance of this arrangement: as East-West opposing group 2's North-South (who was also the lowest scoring group on average). In determining an explanation for this behavior, it is likely that this particular initial distribution of cards happens to not match up well with our determined permutation, and we are thus forced to select a card to convey the best permutation which may not necessarily hold many of the cards in our current hand. For more information on permutation variance, refer to appendix B.

As aforementioned, the tournament was structured to allow each group to face off against all ten players with the ability to play as both teams: North-South and East-West. In comparing our group's performance in each role, we discovered that playing in the East-West partnership produced a slightly better mean score in almost every instance (with the one exception of opposing group 6, whose difference in mean score between North-South and East-West is only 0.019 points). In speculating the reason for this activity, we come to the conclusion

that since our East player has already seen the card that North has revealed, it allows them to choose a slightly more optimal permutation for the West player. With the ability to eliminate one more card, the East player can more effectively select a permutation that would not only provide the West player with more cards in East's hand, but also reduce the number of impossible and extraneous cards in the predetermined permutation.

Summary

Overall, we devised a strategy that seemed promising in the early stages of this project, and ultimately stacked up to some but not all of the implementations of the other groups. However, our strategy certainly had its individuality, as despite being mimicked by only group 4 in their initial technique, it was relatively unique as compared with the strategies of the other nine groups.

The strengths of our strategy include a flexible balance between pseudo-randomization and probability adjustments, which allow for an adaptive approach that still allows valuable information to be conveyed between partners. By using a deterministic shuffling method based on card seeds, this strategy ensures that both players can infer the logic behind each card play. Furthermore, exposing cards that are least likely to be guessed by the guesser balances revealing information early and saving it for future point-scoring opportunities. On the guessing side, updating card likelihoods allows the guessing phase to be informed by both empirical data and probability.

On the other hand, there were a number of pitfalls that likely explains why our player did not perform as well as some of the other groups. In the late game, the strategy relies more heavily on randomness when there are no more likely cards to be added to the guess. This may have cost us some points in certain rounds, particularly where our initial seed proves inefficient at delivering useful permutations. Moreover, when likely cards are being calculated, we weigh all guesses equally, not much taking into account the recency of the guesses. Later guesses are likely to be based on more information, so they should possibly be weighted more heavily.

Future Work

One of the strategies we attempted to implement to improve our overall technique was to alter the seeds for each round. The rationale behind this was that we would not be constrained to a predetermined seed for the entire round, and we could continue to play cards in our hand that could indicate alternative sequences of potential guesses. With this strategy, the guesser could utilize current and previous seeds and guesses to better narrow down which cards are truly present in their partner's hand. When we first tried to implement this, we discovered that our mean score dropped to around 65 cards per round; due

to the time constraint of the project, we scrapped the idea for our final submission. A possible reason for this behavior is that the initial deterministic seed allows each partner to have consistent expectations about the other's hand, while changing the seed may have disrupted the ability to track patterns over guesses. However, for a future iteration, this strategy seems to have merit, so it is likely worth further exploration. In order to implement it properly, it may be worth more concretely defining the seed-changing rule, such as changing the seed on particular turns according to the number of cards left in the hand or other predictable factors.

Additionally, as mentioned in the tournament analysis, the likelihood calculations could also use some further adjustment. Currently, our strategy considers all past guesses equally, but employing a higher weight to more informed later guesses might prove beneficial to assigning likelihood scores. This would also likely decrease the number of ties that arise from this strategy. In the case that there are not enough likely cards to round out a late-game guess, probability-driven guesses might introduce a degree of randomness that counteracts the information that our players have gathered over the course of the game. Utilizing a more deterministic guessing strategy based on our previous information during this stage may thus work to solve this issue.

A Probabilities of hands

Thanks to group 4's Adithi Narayan, who supplied a source with the exact probabilities of suit distribution for a 13 card hand, we have the exact probability of suit distribution for a 13 card hand [1]:

having one suit with more than four cards:
(all combinations) - (combinations with longest suit = 4)
$1 - 0.351 = 0.649$

having one suit with more than four cards, and at least one of every other suit:
(all combinations) - (combinations with longest suit = 4) - (5-4-4-0) - (5-5-3-0) $1 - 0.351 - 0.012 - 0.009 = 0.628$

B Permutation variance

Our entire algorithm relies on the fact that some permutations will reveal more of our hands, while others will reveal fewer of our cards. This can be thought of as *permutation variance*. Understanding the variance amount is helpful to understanding our algorithm's performance.

As an example, let's assume no variance in the amount of cards revealed in each permutation. On average, with the deck shared among four players, a player should expect to have a quarter of the permutation's cards be theirs. With no variance, each permutation in the first round should contain 12 cards. Of those, we own $\frac{12}{4} = 3$ of those. We play any card, since all are equally good, and our teammate guesses 4 cards correct as they know better than to guess their own cards in the permutations. This will continue on and by the game's end, each player will have guessed around a third of all cards correctly, resulting in a total score of $\frac{156}{3} = 52$ — precisely average... and not very good.

As each permutation is randomly generated, it is bound to have non-zero variance in how many of our cards appear. The higher the variance, the better our *best seed* will be, resulting in a higher score.

But what is the variance here? What can we expect our best seed to be on average? How performant should we expect our player to be without looking at any data?

Luckily the mathematical groundwork has already been done for us. The distribution of conditional probabilities for a set drawn through random sampling without replacement can be modelled by a hypergeometric distribution. In a hypergeometric distribution, samples are drawn from a population without replacement, and the probability of getting a certain number of successes can be written as follows:

$$P(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

Where:

N = Population size

K = Number of successes in the population

n = Number of draws

k = Number of observed successes

We then apply these variables specifically to our scenario. Since all cards are drawn from the same deck, N is the size of the deck, which is 52. Here, the number of draws n is the number of times we randomly sample a set of cards, which is each of the cards in our hand. Therefore, $n = 13 -$ number of turns elapsed. We can define "success" as any given card being in a hand, thus the K is the total number of cards within the deck that are in my hand, which is $(12 -$ number of turns elapsed$)$. Note that $K = n - 1$ because each time a card is used to indicate a set of cards, that card itself becomes exposed, which reduces the size of the hand by 1. Finally, k is defined by the number of randomly sampled cards that are actually in the hand. Since k is the observed accuracy, it is also the "seed score" when we find the best seed card to indicate our hand. With all this information, we can formalize the distribution of our seed card accuracies.

Since the distribution of seed card accuracies is a discrete distribution, we can easily get the entire distribution by calculating the P values at each k in

our possible range, which, in this case, is $[0, 12]$. From observing the hypergeometric distribution, we see that the accuracy peaks at 3 (Figure 4). In fact, the hypergeometric distribution's mean is defined by $nk/N = 12 \cdot 13/52 = 3$. This is also what we expect from a more simple way of thinking by taking $12/4 = 3$. However, this is not the interesting part of the problem.

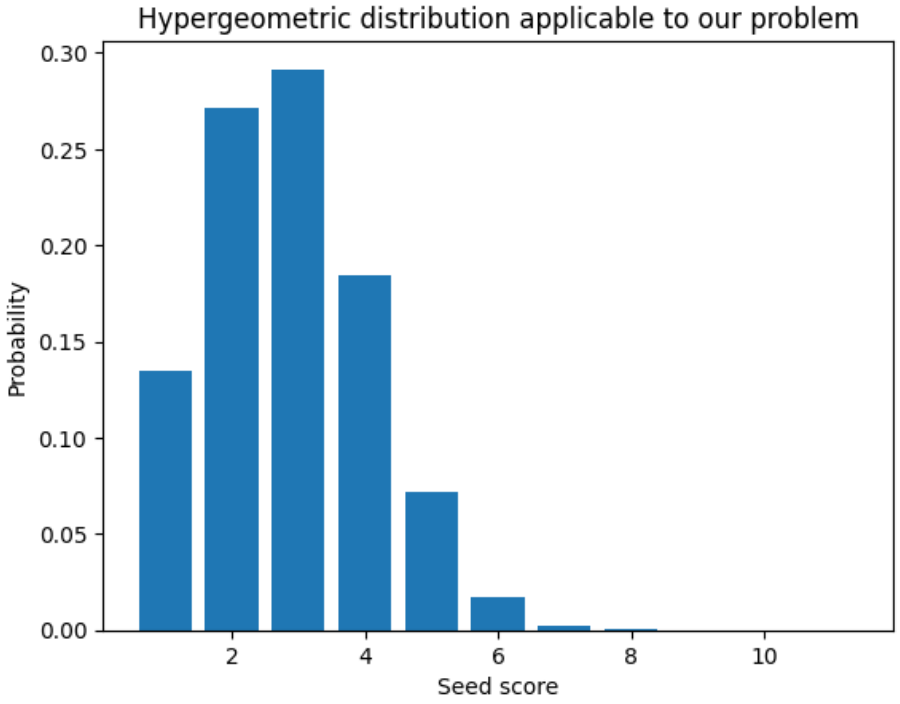

Figure 4: Distribution of seed accuracies

The second step is to find out the expected performance of the best seed card in our hand. Since we are not just playing a random card to indicate our hand, just the expected value of the accuracy is not useful; instead, we need to know the expected value of highest accuracy after k draws. Although we know that the variance of our hypergeometric distribution is $\frac{nK(N-k)(N-n)}{N^2(N-1)}$, we cannot use this to get our expected maximum because the deck is not large enough for us to assume that it is a normal distribution. Analytically, you can derive the expected maximum value by taking the derivative of the cumulative distribution function.

We can also easily do this numerically by randomly sampling K cards from our hypergeometric distribution, and retrieving the maximum accuracy (Figure 5).

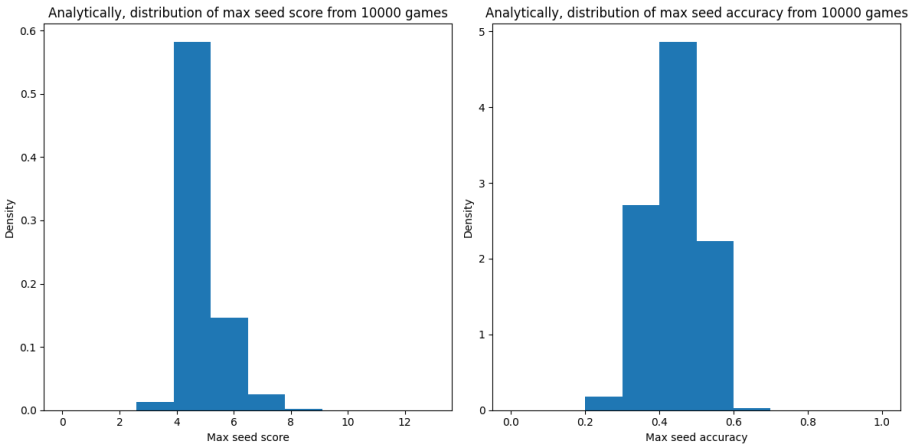

Figure 5: Maximum accuracy distribution from analytical distribution

Comparing the distribution with results from the actual game, the two results are nearly identical (Figure 6).

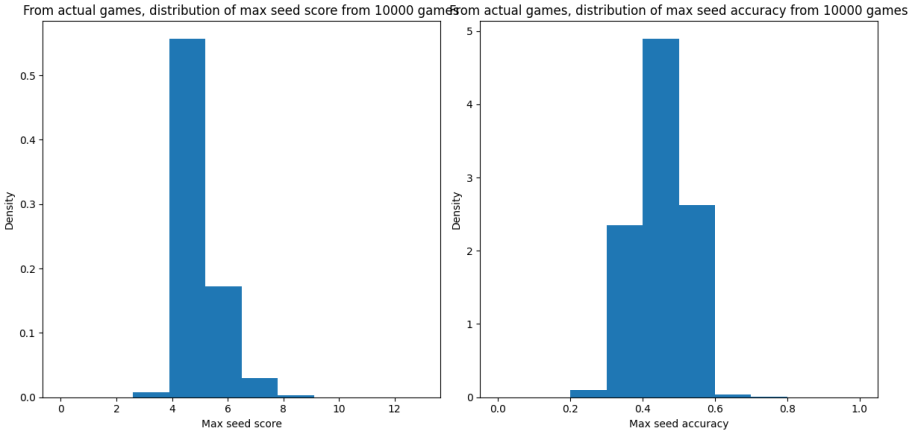

Figure 6: Maximum accuracy distribution from analytical distribution

Looking at our solution analytically is a very helpful way of understanding why our method is performant. Using statistics to our advantage, we turn a uniform distribution of cards into a probability distribution that heavily favors a large number of successful guesses even in just the first round. In fact, in the first round, the number of correct card guesses converges to 4.97, nearly half of our initial hand of 12 cards (excluding the card played).

C Seed rounds

As discussed in the report, the player cannot utilize the best seed strategy throughout the entire game, as the best seed can only be played once. After playing the best seed, the player is forced to play their second best seed, and so on. Effort were made towards creating a new mapping from cards to seeds for each round, but the results weren't so good as discussed in the Future Work section.

Therefore, we had a variable `SEED_ROUNDS`, which indicated for how many rounds a player should play their best current seed before switching strategies. Initially, we played the best seed for 3 rounds before switching, but 2 or even 1 round would yield similar results.

To find out, we used our Google Cloud credits to run simulations in a VM instance over a couple of days, getting average scores for each of the three proposed values: 1, 2, or 3. The results can be seen below in Listings 4, 5 & 6 for `SEED_ROUNDS` = 1, 2, 3 respectively. Our player played as NS, against the default player and had a mean score of 88.05, 88.39 and 87.02 for `SEED_ROUNDS` 1, 2 and 3 respectively. From these results, we saw that `SEED_ROUNDS` = 2 gave slightly better results than for 1 round, ultimately selecting 2 as the number of rounds. However, for more optimized following strategies, the extra round might be worth the $88.39 - 88.05 = 0.34$ points per game drop gotten from selecting `SEED_ROUNDS` = 1.

```
starting 524288 rounds at Sat Oct 12 01:30:16 UTC 2024
Scores over 524288 simulations:
NS Mean: 88.05 | NS Std Dev: 9.76
EW Mean: 25.00 | EW Std Dev: 4.14
Finished 524288 rounds at Sat Oct 12 08:52:20 UTC 2024
```

Listing 4: Simulations with SEED_ROUNDS 1

```
starting 524288 rounds at Thu Oct 10 09:03:05 UTC 2024
Scores over 524288 simulations:
NS Mean: 88.39 | NS Std Dev: 9.29
EW Mean: 25.00 | EW Std Dev: 4.14
Finished 524288 rounds at Thu Oct 10 17:49:07 UTC 2024
```

Listing 5: Simulations with SEED_ROUNDS 2

```
starting 524288 rounds at Fri Oct 11 04:29:50 UTC 2024
Scores over 524288 simulations:
NS Mean: 87.02 | NS Std Dev: 9.27
EW Mean: 25.00 | EW Std Dev: 4.14
Finished 524288 rounds at Fri Oct 11 14:05:08 UTC 2024
```

Listing 6: Simulations with SEED_ROUNDS 3

References

[1] Bridgehands, "Probability hand distribution," 2011, accessed: 2024-10-21. [Online]. Available: https://www.bridgehands.com/P/Probability_Hand_Distribution.htm