USING SIMULATION TO ASSESS STRATEGY EFFECTIVENESS IN UNO*

ROSE PORTA†, AYUSHI GARG‡, HIRAL GUPTA§, AND CARRIE FENG¶

Abstract. UNO is a popular card game that has entertained players of all ages for decades. It is known for its simple rules, fast-paced gameplay, and strategic decision-making. This project aims to examine the impact of four key strategies (number preference, color preference, use specials, and save specials) in a 2-player and 3-player UNO game with and without special cards. We conducted a large-scale Monte Carlo simulation to replicate the UNO game. We conducted Z-tests to determine the effectiveness of each strategy as compared to a random strategy. We found that several of the strategies were effective at increasing winning probability, although the most effective only increased chances of winning by at most a couple of percentage points. The strategies found to be most effective differed based on the number of players and whether or not special cards were present. Overall, this research contributes to understanding the effectiveness of different strategies in the context of UNO gameplay.

Key words. UNO, simulation, game, strategy

AMS subject classifications. 91-10

1. Introduction.

In this project, our group aims to explore the effectiveness of different strategies in the popular card game UNO. While UNO is a game of chance, we want to investigate whether players can increase their chances of winning by focusing on attributes, such as color, number, or special cards.

1.1. Background Research.

Many works on finding an optimal game strategy using reinforcement learning have been done in this area. A recently published

*Submitted to the editors 05/10/2023.

†Department of Statistical and Data Sciences, Smith College, Northampton, MA (rporta@umass.edu, http://roseporta.com).

‡Department of Mathematics, University of Massachusetts at Amherst, Amherst, MA (ayushi-garg@umass.edu).

§Department of Mathematics, University of Massachusetts at Amherst, Amherst, MA (hiral-gupta@umass.edu).

¶Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA (xfeng@umass.edu).

This manuscript is for review purposes only.

paper provides insights into how established reinforcement learning models behave in a real-world competitive scenario.[3] Another medium blog conducted a reinforcement learning on UNO and states that while the models are able to outperform random players, their performance against human players remains unclear due to two reasons[8]: limited strategic evaluation since players typically only have one playable card in their hand. And the large amount of luck involved in the game because UNO brings in a large amount of stochasticity.[5], which decreases the potential for strategic supremacy.[9]The blog suggests that while UNO may not provide as much potential for strategic supremacy as other board games, there is still potential for intelligent play in certain situations. In our project, we decided to use a new approach rather than reinforcement learning considering how many paper has already been published in this field. We further explored the existing literature on the theoretical foundations of game theory and algorithms. One paper presents decision algorithms for determining the outcome and identifying winning strategies in various classes of multiplayer games with incomplete information. [7] Another relevant paper in the realm of card game variations contributes to a broader understanding of card game variations and probability-based decision-making.[10] The insights gained from studying decision algorithms and probility can inform the development of intelligent gameplay agents and enhance strategic thinking in multiplayer card games.

1.2. Research Question. Our research question is what strategy can we implement to increase the probability of playing the game UNO? We will test the claim that focusing on a certain attribute like color, number, or special cards when choosing which card to play increases the player's probability of winning compared to a randomized approach.

2. Methods. In order to test our claim, we developed an algorithm to simulate a basic UNO game, then recorded the outcomes given different variations of the basic algorithm including different player strategies, different numbers of players, and special cards included or not. For each of 27 different variations displayed in Table 2 (appendix), we simulated 10,000 games and recorded the number of games that player

0 won. We then used Z-tests to compare the proportion of games won by player 0 given different strategies as compared to random chance probability. For the full details of our code, visit our git repository [1].

2.1. Baseline Algorithm. We started with a very basic, simplified model of the UNO game in order to establish a baseline before adding more complicated elements. Our basic version includes only the number cards (no special cards), two players, and both players using a random strategy.

We constructed this basic algorithm according to the Official UNO Rules[1], and our assumptions are outlined in Appendix B. For full details on the cards included in the deck, see Appendix C.

Our central function, *play_game*, which initiates a game and alternates between the players' turns until one player runs out of cards. The function returns a value of 1 if player 0 wins and a value of 0 otherwise. This allows us to easily count the number of games won by player 0. We chose to keep track of wins for only one player, which is the player whose strategy changes while all other players maintain random strategies. Player 0 is not necessarily the player who goes first, as the order of players is randomized at the beginning of each game. In order to compute the number of games player 0 wins out of a large number of games, we developed a function *play_n_games* which runs n simulations of the game and returns the number of games where player 0 won.

2.2. Adding Special Cards. Once we had solidified the basic version of the algorithm, we added in the special cards (24 total), which complicate the game by making it possible for players' turns to be skipped, order of players to be reversed, target color to be changed, or players forced to pick up two or four extra cards from the deck. We included all special cards present in the original version of the official UNO game (*Draw 2*, *Skip*, *Reverse*, *Wild Color*, and *Wild Draw 4*).

We modified the *play_game* function such that it executes the intended action for each special card. We added a parameter to the function such that *special_cards* can

[1] https://github.com/rporta23/draw4

be set to *True* or *False*, so we can run a version with or without the special cards.

2.3. Adding Multiple Players. In order to expand the game beyond two players, we added a *numPlayers* parameter to the *play_game* function such that we can set the number of players and the players list will be automatically generated within the function. For the purpose of clarity within this analysis, we only simulated games with two or three players, but the number of players could be set as any (positive whole) number. An outline of the final *play_game* algorithm is outlined in Algorithm D.1.

2.4. Strategies. Much of the game UNO is based on random chance, but the key area where players have choice is in which card they choose to play if they have multiple cards matching the target card. For example, if the target card is a *Blue 9*and the player has a *Blue 5*, a *Blue 3*, and a *Red 9* in their hand (all matches), which do they choose to play? We developed four strategies, all based on this choice of which card to play given multiple matches, in order to test whether any particular strategy is significantly correlated with a higher probability of winning. The strategies are outlined in Table 1.

Table 1
Strategies

Strategy	Description
Random	Given a list of matches, choose one to play at random.
Color Preference	Give preference to matches where the color of the card matches the target card color.
Number Preference	Give preference to matches where the number of the card matches the target card number.
Use Special	Give preference to special card matches.
Save Special	Give preference to non special card matches.

Implementation of strategies works such that if there exist matches which fit the strategy preference, the player chooses one of those to play(at random). Otherwise, the player chooses a random card from all matches. The general algorithm used for implementing each strategy is outlined in Algorithm D.2. The condition for appending to *preference_matches* list would change depending on the specific strategy.

2.5. Simulations. Once we had built out the full version of the game, we chose to run simulations under six separate conditions:

1. Two players, no special cards, only player 0's strategy changes. (3 cases)
2. Two players, special cards, only player 0's strategy changes. (5 cases)

3. Three players, no special cards, only player 0's strategy changes. (3 cases)
4. Three players, special cards, only player 0's strategy changes. (5 cases)
5. Three players, special cards, multiple players have different strategies. (4 cases)
6. Two players, special cards, multiple players have different strategies. (7 cases)

Within each condition, we simulated different combinations of the strategies described in the previous section as compared to a baseline of all players having random strategies. We primarily chose to focus on how a player's probability of winning changes when only one player adopts a strategy and all other players play randomly. We focused on changing only one player's strategy because this was the clearest way to determine whether a particular strategy is advantageous as compared to a random strategy. Conditions 1-4 represent this primary approach.

After we ran some initial tests and determined which strategies were most effective as compared to a random strategy, we added some cases (included in conditions 5 and 6) in which we tested the most effective strategies against other strategies besides random. We chose to narrow our combinations based on which strategies showed up as effective in our initial testing since if we were to test all possible combinations, there are too many to test, and it becomes more complicated to compare the outcomes. We also chose to limit the number of players to two or three for the same reason. For full details of which strategy combinations we tested under each condition, refer to Table 2.

2.6. Random Seed. In order to ensure that we were comparing strategies directly and that random chance was not affecting our comparisons, we implemented a random seed such that when we run n games via the *play_n_games* function, we set a random seed for each game which corresponds to the number of the current iteration. For example, if we are on the 100th game, the seed will be set to 100. This implementation of a random seed essentially ensures that if we run n simulations, we will generate a different shuffling of the deck for each of those games, however that same set of n ordered decks will be used for all strategy combinations.

The implementation of the random seed within the *play_n_games* function is represented visually in Algorithm D.3.

2.7. Statistical Analysis. The outcome of each game is represented by a Bernoulli(p) random variable defined as follows:

$$X_i = \begin{cases} 1 & \text{if player 0 wins} \\ 0 & \text{if player 0 does not win} \end{cases}$$

where $i = 1, \ldots, n$ within each case of interest and $P(X_i = 1) = p$.

We can assume that within each case, the outcome of each game is independent of the outcomes of all other games since the deck is shuffled randomly each time, and therefore the X_is are i.i.d.

Our goal in running our simulations is to find an estimate for the parameter p representing the probability that player 0 wins given the condition and strategy combination. We generate this estimate, $\hat{p}$ by running $n = 10,000$ game simulations per case and recording the number of games where player 0 wins.

We generate the estimate for p by computing

$$\hat{p} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

where $n = 10,000$ and $x_1, x_2, ..., x_n$ represent the observed outcomes for each game. Essentially, we divide the total number of games won by player 0 by the total number of games, which is 10,000.

Since each estimate $\hat{p}$ represents the sample mean of a sequence of i.i.d. Bernoulli random variables, we can use the Central Limit Theorem (Theorem D.1) to conclude that with a large n, the distribution of p will be approximately normal. Since $n = 10,000$ for each case, n is sufficiently large to assume the CLT. This assumption of a normal distribution allows us to use Z-tests for comparison of proportions to determine whether or not p is significantly different from our baseline proportion of 0.5 for two players, or 0.33 for three players (assuming all players have an equal probability of

This manuscript is for review purposes only.

winning). If we let $numPlayers$ represent the number, of players, then the baseline probability is $\frac{1}{numPlayers}$.

For each case, we used a Z-test to test the hypothesis:

$$H_0: p = \frac{1}{numPlayers}$$

$$H_A: p \neq \frac{1}{numPlayers}$$

We used an initial significance threshold of $\alpha = 0.05$. We also implemented a Bonferroni multiple test correction within each condition in order to control the overall false positive rate within each group at 0.05. The Bonferroni correction is a process of adjusting the α level for a family of statistical tests in order to control the false positive rate accross all of the tests at the original α level, and it is defined mathematically as follows: [2]

$$\alpha_{corrected} = \alpha_{original}/m$$

where m is the number of tests performed.

3. Results. Our initial question was whether we can use the z-hypothesis test for our data. We found that not only were the wins approximately normally distributed, but also that two-player games with a random strategy were centered at a median of 0.50, while our three-player games were centered at 0.33. This matches our hypothesis that if p_0 and other players have random strategies, p_0 will win 50% of the time in two-player games and 33% of the time in three-player games.(refer to Figure 5 and Figure 6)

Moving on to two player game without special cards, color preference strategy was the most significant in increasing p0 chances of winning with a p-value of .0007, while number strategy was significant in decreasing p0 chances of winning with a p-value of .02. These results are displayed in Figure 1.

Next for a two player with special cards, number preference, save specials and use specials were all significant in increasing p0's chances of winning, with a p-values of

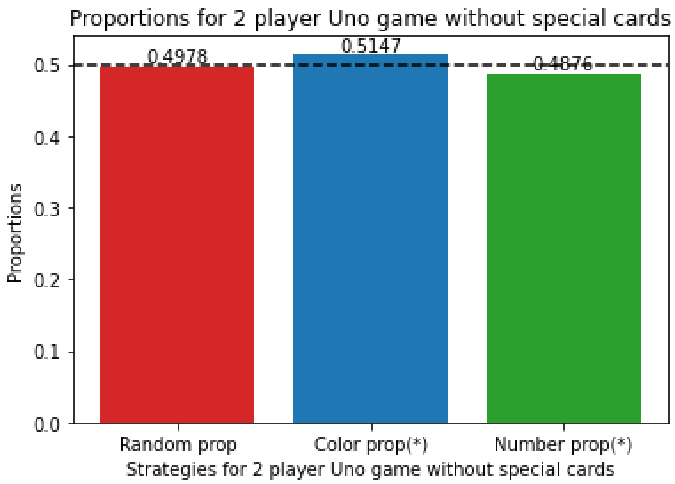

Fig. 1.

.0117, 0.002358, and 0.0001977 respectively. These results are displayed in Figure 2.

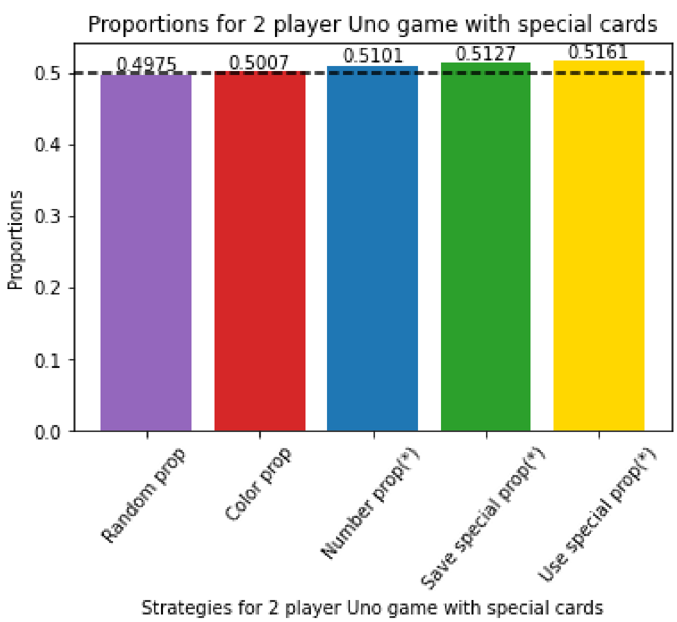

Fig. 2.

Moving on to three player with no special cards, only number preference strategy was significant with a p-value of .012 in decreasing the p0 chances of winning for 3 player without special cards. These results are displayed in Figure 3.

For three player with special cards, number preference and save specials were

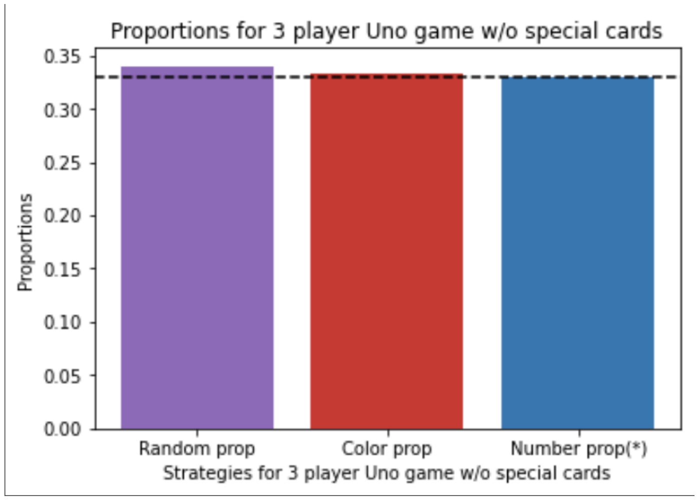

Fig. 3.

both significant in increasing p0 chances of winning with p-values of 0.005 and 0.0015 respectively. These results are displayed in Figure 4.

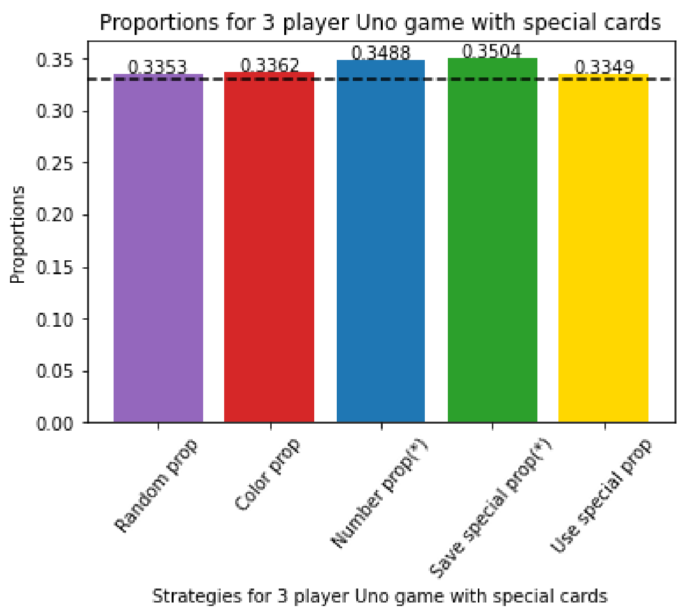

Fig. 4.

When looking at combined strategies for conditions 5 and 6, we found that the effectiveness of strategies that showed up as significant when played only against random strategies became insignificant when played against other non-random strategies.

We omit the formal results for these conditions.

4. Summary. Analysing the results, we were able to draw some conclusions. The first being that the use special strategy is the most efficient strategy when the game is played between two players but when we have 3 players in the game, the save special strategy turns out to be the most efficient one. This is interesting because both the strategies are polar opposite of each other. The second conclusion was that the number preference strategy decreased the number of wins for player 0 without any special cards but increased the number of wins for player 0 when played with special cards. The last conclusion we drew was that the difference in proportions did not appear large visually, but was statistically significant for many of the strategies.

One strength of our study is that the code has a fast run time and that allowed us to obtain a large sample size by running many simulations. However, we also recognized some limitations of our project. The main one being that we were not able to run all strategies against each other as we had five different strategies and running then across multiple players would give us too many combinations to keep track of. Future research could further assess how these strategies perform against each other in different combinations.

Appendix A. Software. We developed the algorithm using Python 3.10 through iPython notebooks within the Google Collaboratory platform. We utilized the *random* module for random shuffling and random choices of cards, the *pandas* module for creating a dataframe to organize our results, the *statsmodels* module for running z-tests, and the *seaborn* module for generating plots.

Appendix B. UNO Rules Assumptions. We used the following assumptions to construct our algorithm based on the Official Uno Rules.[1]

- Game for 2-10 players
- Each player starts with 7 cards
- Rest of cards go to draw pile
- Choose a random first player, then move clockwise
- Start by picking the top card from the draw pile and flipping it face-up

This manuscript is for review purposes only.

- Player 1 must either put down a card which matches the color, number, or action of the card, or pick a card from the draw pile
  - Player can play the card right away if they pick up a matching card from the draw pile.
  - A player can only put down at most one card per turn.
- The goal is to get rid of all cards; when a player has one card left, they must say “UNO!”.
- The first player to run out of cards wins.

Note that we did not incorporate the possibility of a player ”forgetting” to say ”UNO!” when they have one card left in our algorithm, so this rule is essentially disregarded.

Appendix C. Distribution of Deck.

The deck contains 108 total cards with special cards or 76 cards without special cards. We used this reference [6] to determine the distribution of cards within the deck. The distribution is as follows:

- 76 number cards
  - Four colors: Blue, Yellow, Red, Green
  - 19 cards per color
    - one 0
    - 2 of each other number 1-9
- 24 action cards
  - 3 types of action cards, 2 cards per color for each action
  - Actions:
    - *Draw 2*: next player must draw 2 cards and miss their turn.
    - *Reverse*: direction of game play reverses (clockwise to counterclockwise or vice-versa).
    - *Skip*: Next player skips their turn.
- 8 wild cards
  - 2 types of wild cards, 4 of each type
    - *Wild Color*: Player chooses color to continue game play.

  * *Wild Draw 4*: Player chooses color to continue game play and next player must draw 4 cards

We are using the classic version of the game for reference and are not including the newer special cards which have been added recently (*Wild Swap Hands Card*, *Wild Shuffle Hands Card*, and *Wild Customizable Card*).[1]

Appendix D. Supplementary Algorithms, Theorems, and Figures.

Algorithm D.1 Play Game

```
Require:
  numPlayers: int
  strategies: tuple of length numPlayers
  special_cards : boolean
  seed: int

  random seed ← seed
  instantiate and shuffle deck
  game_over ← False
  create list of players of length numPlayers
  deal cards to players
  randomize order of players
  target_card ← top card from deck
  while not game_over do
    for player in players do
      check for special card actions
      choose card to play
      play card or draw card
      if new card played then
        target_card ← card played
      end if
      if player has no cards left in hand then
        game_over ← True
        if current player is player 0 then
          return 1
        else
          return 0
        end if
      end if
    end for
  end while
```

Theorem D.1 (Central Limit Theorem). *[4] Let $X_1, X_2, \ldots$ be a sequence of i.i.d. random variables with $EX_i = \mu$ and $0 < VarX_i = \sigma^2 < \inf$. Define $\bar{X_n} = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i$.*

Algorithm D.2 General Strategy Algorithm

```
Require:
  target_card : Card
  player: Player
  deck: Deck
  matches : list of Cards

  preference_matches ← [ ]
  for card in matches do
    if card matches preference then
      append card to preference_matches
    end if
  end for
  if preference_matches contains at least one card then
    card_to_play = random choice from preference_matches
  else
    card_to_play = random choice from matches
  end if
  return card_to_play
```

Algorithm D.3 Play n Games

```
Require:
  n : int
  numPlayers: int
  strategies: tuple of length numPlayers
  special_cards : boolean
  seed: int

  p0_wins ← 0
  for i = 1, ..., n do
    value ← play_game(numPlayers, strategies, special_cards, seed = i)
    p0_wins ← p0_wins + value
  end for
  return p0_wins
```

Let $G_n(x)$ denote the cdf of $\sqrt{n}(\bar{X}_n - \mu)/\sigma$. Then, for any x, $-\infty < x < \infty$,

$$\lim_{n\to\infty} G_n(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy;$$

that is, $\sqrt{n}(\bar{X}_n - \mu)/\sigma$ has a limiting standard normal distribution.

Acknowledgments. Thanks to Professor Patrick Flaherty (Associate Professor, Department of Mathematics and Statistics, University of Massachusetts at Amherst) and Thomas Cook (Teaching Assistant for STAT 535 Spring 2023, University of Mas-

TABLE 2
Variations within each Simulation Condition

condition	num_players	strategies	special_cards
1	2	('random', 'random')	False
1	2	('color', 'random')	False
1	2	('number', 'random')	False
2	2	('random', 'random')	True
2	2	('color', 'random')	True
2	2	('number', 'random')	True
2	2	('save specials', 'random')	True
2	2	('use specials', 'random')	True
3	3	('random', 'random', 'random')	False
3	3	('color', 'random', 'random')	False
3	3	('number', 'random', 'random')	False
4	3	('random', 'random', 'random')	True
4	3	('color', 'random', 'random')	True
4	3	('number', 'random', 'random')	True
4	3	('save specials', 'random', 'random')	True
4	3	('use specials', 'random', 'random')	True
5	3	('color', 'number', 'save specials')	True
5	3	('random', 'number', 'save specials')	True
5	3	('random', 'random', 'save specials')	True
5	3	('random', 'random', 'number')	True
6	2	('use specials', 'use specials')	True
6	2	('number', 'use specials')	True
6	2	('save specials', 'use specials')	True
6	2	('number', 'number')	True
6	2	('number', 'save specials')	True
6	2	('save specials', 'save specials')	True
6	2	('save specials', 'number')	True

sachusetts at Amherst) for their support throughout the process of our project.

REFERENCES

[1] *The Full Rules for Uno Card Game Plus Other Versions*, Nov. 2009, https://www.unorules.com/ (accessed 2023-05-04).

[2] Z. , *The Bonferroni Correction: Definition & Example*, Feb. 2021, https://www.statology.org/bonferroni-correction/ (accessed 2023-05-09).

[3] P. Barros, A. Tanevska, and A. Sciutti, *Learning from learners: Adapting reinforcement learning agents to be competitive in a card game*, in 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 2716–2723, https://doi.org/10.1109/ICPR48806.2021.9412807.

[4] G. Casella and R. L. Berger, *Statistical Inference*, Cengage Learning, Australia ; Pacific

This manuscript is for review purposes only.

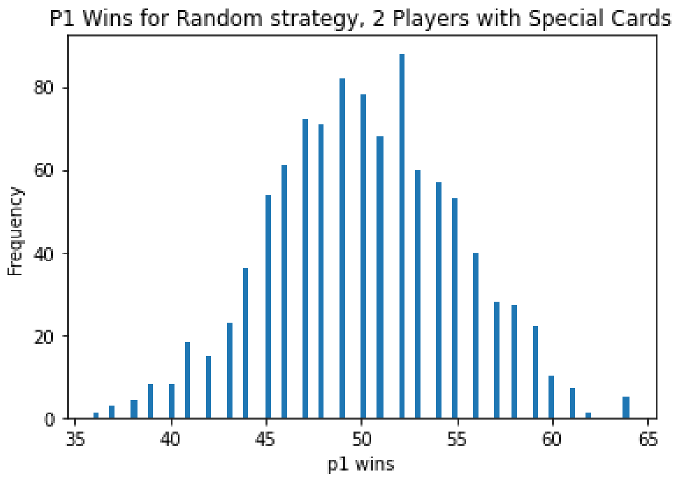

FIG. 5.

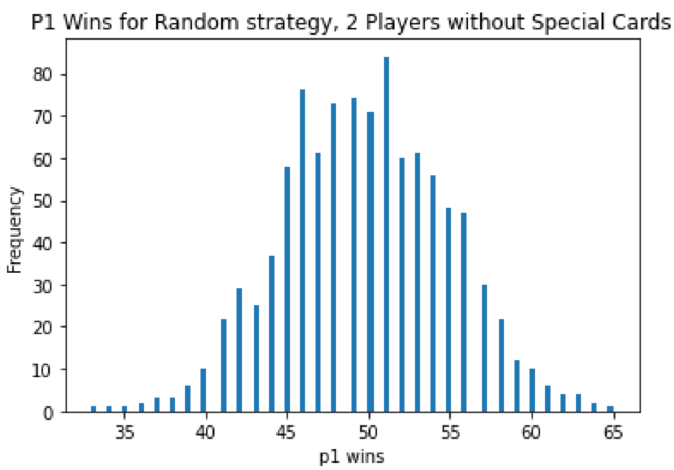

FIG. 6.

Grove, CA, 2nd edition ed., June 2001, https://mybiostats.files.wordpress.com/2015/03/casella-berger.pdf.

[5] E. D. DEMAINE, M. L. DEMAINE, N. J. HARVEY, R. UEHARA, T. UNO, AND Y. UNO, *Uno is hard, even for a single player*, Theoretical Computer Science, 521 (2014), pp. 51–61, https://doi.org/https://doi.org/10.1016/j.tcs.2013.11.023, https://www.sciencedirect.com/science/article/pii/S0304397513008670.

[6] INC, *UNO!™ - Official UNO™ mobile game*, https://www.letsplayuno.com/news/guide/20181213/30092_732567.html#:~:text=Each%20color%20contains%2019%20cards,of%20cards%20numbered%201%2D9. (accessed 2023-05-04).

[7] G. PETERSON, J. REIF, AND S. AZHAR, *Decision algorithms for multiplayer noncooper-*

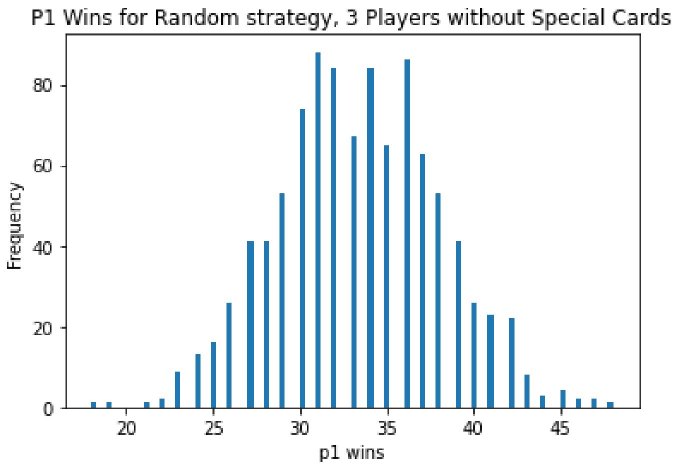

Fig. 7.

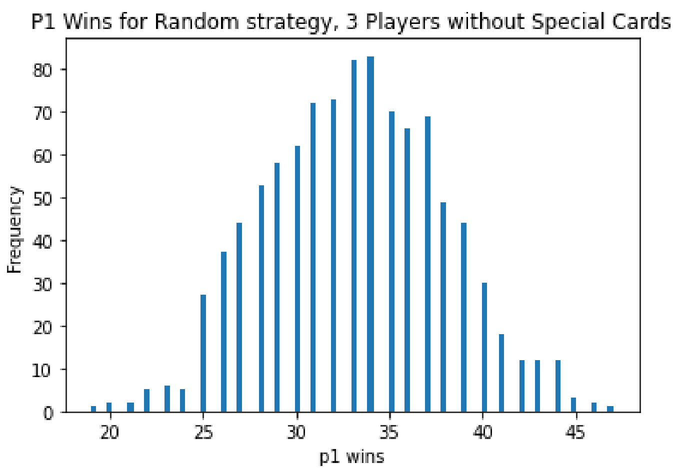

Fig. 8.

ative games of incomplete information, Computers Mathematics with Applications, 43 (2002), pp. 179–206, https://doi.org/https://doi.org/10.1016/S0898-1221(01)00282-6, https://www.sciencedirect.com/science/article/pii/S0898122101002826.

[8] B. Pfann, *Tackling uno card game with reinforcement learning*, Jan 2021, https://towardsdatascience.com/tackling-uno-card-game-with-reinforcement-learning-fad2fc19355c.

[9] A. Ramadhan, N. Maulidevi, and H. Iida, *Game refinement theory and multiplayer games: case study using uno*, 02 2015.

[10] Z. Stanić, *Probability and strategy in a variation of a blackjack game*, 2013, https://arxiv.org/abs/1304.5455.