Strategy in the Dice Game "Pig"

Dylan Gonzalez
University of Central Florida
COT3100: Intro to Discrete Structures
Prof. Arup Guha
Dec. 2, 2024

Abstract

This paper explores strategies for the dice game Pig, focusing on probabilistic analysis to determine the most effective "hold at k" thresholds for optimal play. Using Python simulations, this study evaluates the expected number of turns required to win under varying strategies. By leveraging insights from discrete mathematics and probability, the research identifies holding at or near 20 as the most effective strategy for minimizing turns, aligning with existing findings. However, this approach is shown to have limitations in scenarios where the opponent's score or game state demands more aggressive play. These results highlight the intersection of mathematical modeling and real-world decision-making in games of chance.

Introduction

The game of Pig is a dice game that involves two or more players competing to reach 100 points. The game is made up of multiple rounds where each player takes turns rolling a single die, usually a standard six-sided die. One round of pig is structured like this:

- Player one's turn:
    1. Rolls the die, keeping note of the value
    2. If the player rolled a 1, ignore all points gained this turn and pass the die to the next player
    3. If the player rolls any other number, write it down and add it to the player's score for the turn.
    4. Allow the player to choose to roll the die again (assuming they did not roll a 1) and repeat steps 1-4, or hold their points and add them to their total score for the

game, passing the die to the next player. It is this total score that determines the winner.

- Player two and more:
    - Follow the same procedure as player 1, rolling the die until they hold or roll a 1. Then passing it back to player 1, starting the next round.

The first player to reach a total score of 100 at any point is the winner (rules can be modified to add the turn scores to the total at the end of each round, allowing multiple players to reach 100 in a single round and therefore creating a tie; in the case of this study, the first to 100 wins).

Strategy for rolling in this game could include holding at a certain turn score, ensuring you get the highest value per turn without risking rolling too much, which could incur losing all your turn's value. Since this is the most modelable strategy, this is the one I will explore in this study.

Methods and Findings

To explore these probabilities, one could look at the game from a single-player standpoint, and instead of analyzing who wins, analyze how many turns it would take to win. This way, we can see which strategy has the highest chance of winning against an opponent by seeing which strategy wins in the least amount of turns on average.

In any given turn, the player can roll the die no matter

E = the expected value rolling a single die (besides 1)

t = the current turn score of the player

T = the expected turn score after rolling from the current turn score t. Numberphile (2021) demonstrated in his video “The Math of Being a Greedy Pig” that this could be modeled with the equation:

$$\frac{1}{6} \times 0 \ + \ \frac{5}{6} \times (E \ + \ t) \ = \ T$$

The expected value of a random number between 2 and 6 is 4, so E = 4,

$$\frac{5}{6} \times (4 \ + \ t) \ = \ T$$

And since we want to roll while gaining points on average, we should roll as long as our new score (on average) is expected to be greater than our current score, or:

$$\frac{5}{6} \times (4 \ + \ t) \ < \ t$$

Solving for t, we want to roll whenever our current turn score is

$$t \ < \ 20$$

This follows Numberphile’s findings in his video, so in order to test it, I programmed a single-player version of the game in Python, `playpig.py`. I also created a simulator to run many of these games while holding at various thresholds *k*. `pigsim.py` runs the simulator for one value of k while `ultrapigsim.py` runs the simulator for every integer in a range given, inclusive.

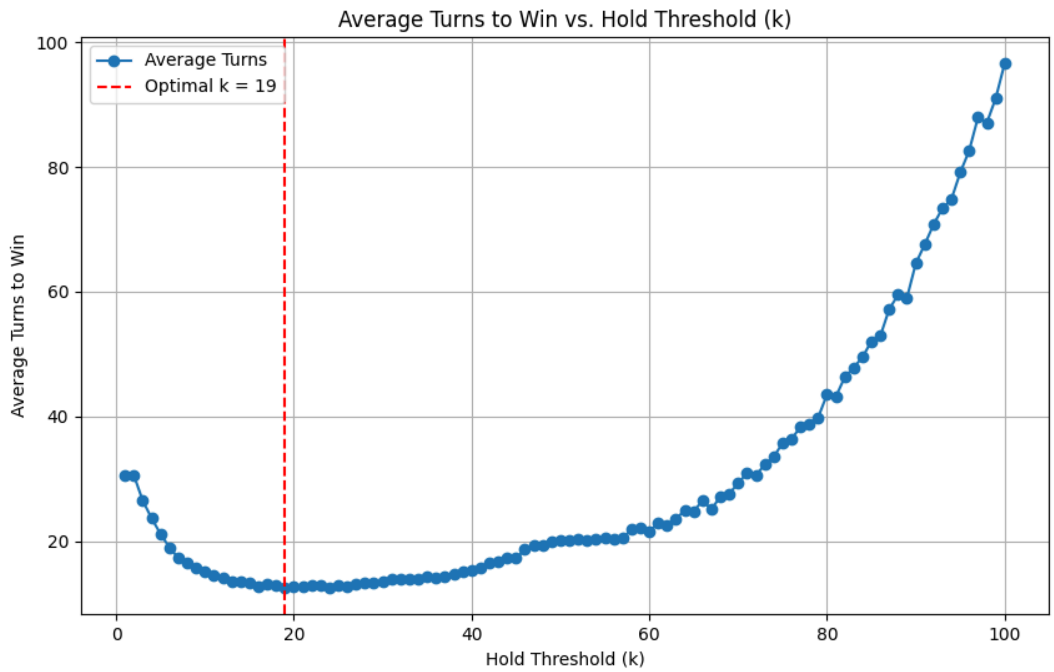

Figure 1. `ultrapigsim.py` ran with a range of 1 to 100, simulating 1,000 games each

In this graph, we can see that holding anywhere below around 6 and above around 50 is out of the question, as we'd expect holding after every roll to have too little profit each turn and almost never holding holds way too much risk in rolling a 1 and losing it all. So, we should look closer between 10 and 30 since contains the lowest point in the graph. This also allows us to run more simulations per hold threshold, as it will take much less time (the values closer to 100 take very long to simulate since some games last a very long time since it is very rare to never roll a one).

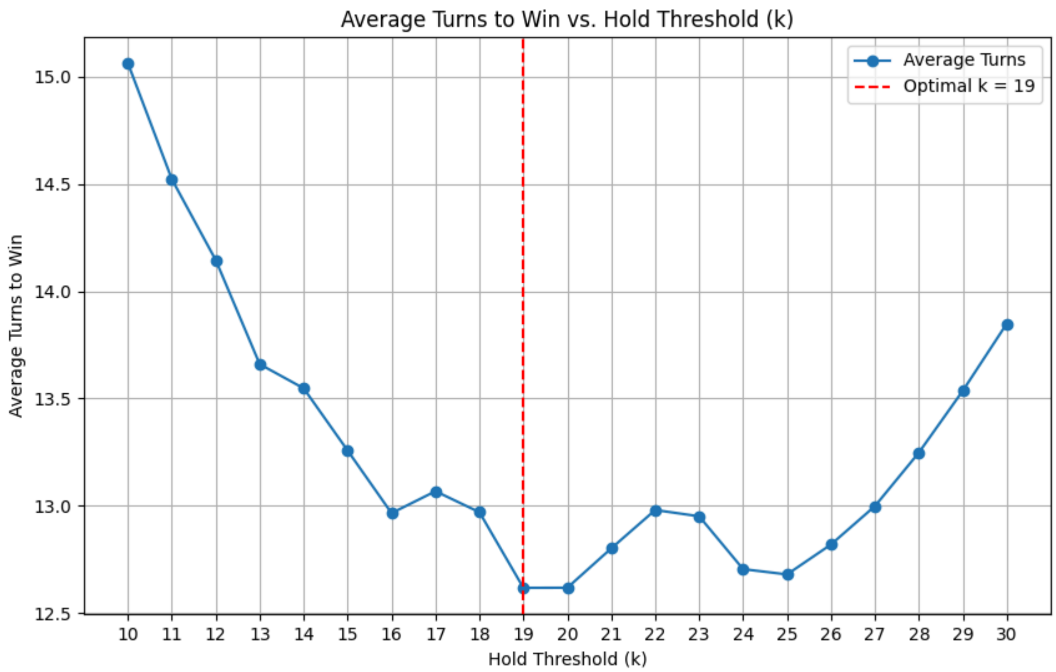

Figure 2. `ultrapigsim.py` ran with a range of 10 to 30, simulating 200,000 games each

Note that 19 and 20 seem to both be viable, and when this code is run multiple times, they sometimes have the same average and sometimes beat or lose to each other by around 0.02 turns. Just in case, I ran it with a range of 18 to 21, simulating 1,000,000 games each. This was the output:

Running...

Done! While Holding at 18 Finished on average in 12.97 turns, with a minimum of 5 turns and maximum of 51 turns.

Done! While Holding at 19 Finished on average in 12.64 turns, with a minimum of 5 turns and maximum of 51 turns.

Done! While Holding at 20 Finished on average in 12.64 turns, with a minimum of 5 turns and maximum of 50 turns.

Done! While Holding at 21 Finished on average in 12.81 turns, with a minimum of 4 turns and maximum of 52 turns.

19 and 20 still seem to be about the same both still being the best numbers to hold on, so I increased the number of leading decimals and ran it with a range of 19-20, each 2,000,000 simulations. I ran it twice, and these were the results:

Running...

Done! While Holding at 19 Finished on average in 12.634298 turns, with a minimum of 5 turns and maximum of 52 turns.

Done! While Holding at 20 Finished on average in 12.633979 turns, with a minimum of 5 turns and maximum of 59 turns.

Running…

Done! While Holding at 19 Finished on average in 12.63766 turns, with a minimum of 5 turns and maximum of 51 turns.

Done! While Holding at 20 Finished on average in 12.6396715 turns, with a minimum of 5 turns and maximum of 53 turns.

These outputs still don't show a clear winner, despite being run with 2,000,000 simulations. This could only mean I need to run many more simulations.

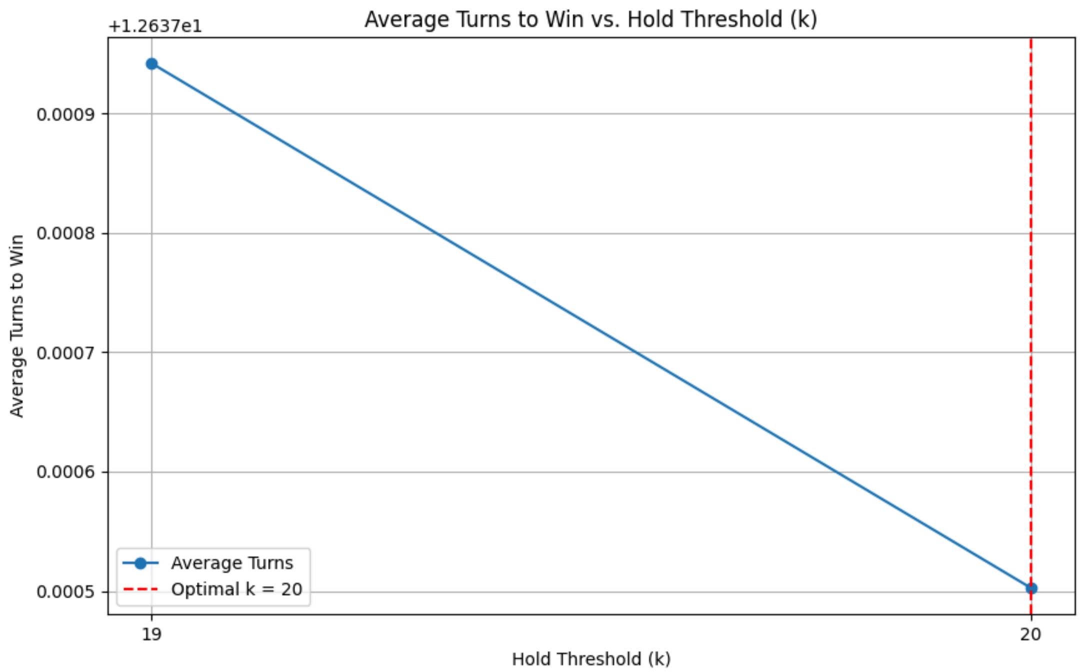

Figure3. `ultrapigsim.py` ran with range 19 to 20, simulating 10,000,000 games each

Running it three times at 10,000,000 simulations, all three had a winner, k=20, but it wasn’t by much, so it very well could be a result of chance. At most it was better by 0.004 turns, and at the least it was by 0.0004 turns. However, this still aligns with our calculation earlier, so it's safe to assume holding at or above 20 is the best “hold at k” strategy, with holding at or above 19 being slightly behind or just as viable. If you would like to try your luck against it, `versuspig.py` is available in the github under references and allows you to play against this “hold at 20” strategy.

Discussion and Conclusions

With over 60,000,000 simulations of Pig ran, it can be concluded that it is optimal for a player to hold their turn’s score whenever they reach a score of 20 or more. The score ensures the most points gained per turn, minimizing risk while maximizing reward. However, this strategy fails to consider the opponent's score. For example, what if the opponent's score is 99, and you are much further behind? Or what if the opponent just has a large lead, like having 70 points, while you only have 20? In both of these, it is logical to try to roll for many more points in one turn in order to catch up before the opponent can reach 100. These types of scenarios are very difficult to model and simulate, needing to take the probability of the opponent winning in the next turn into account when finding the player's own probability of winning by rolling even more than 20. In "Optimal Play of the Dice Game Pig" Neller describes that "the 'hold at 20' policy only serves as a good approximation to optimal play when both players have low scores." Instead, he proposes an optimal player should hold at a much higher value when the opponent is ahead and hold at a low value when the opponent is behind. In "Practical Play of the Dice Game Pig" he calculates that the "hold at 20" strategy has an 8% win disadvantage against the "optimal player" (which refers to a theoretical player who always makes the best possible decisions to maximize their probability of winning). Even more bizarre in his findings is that the "hold at 25" strategy only has a 4.2% win disadvantage against optimal play, making it better than the "hold at 20" strategy. This doesn't match up with my findings, but it is possible adding the complexity of an opponent changes strategy so much that hold at 25 is more viable, or that hold at 25 specifically works against optimal play somehow. In the same paper, he finds the best strategy is actually "If either player's score is 71 or higher, roll for the goal. Otherwise, hold at 21 + round($\frac{j-i}{8}$)", with j being the opposing player's score and i being the player's. This strategy actually only holds a 0.9222% win disadvantage against optimal play, making it the best strategy known for humans to be able to use. It might be possible with an AI to find the "optimal play," and it would make an interesting project, but it most likely wouldn't be practical for a human to use in a real life game.

One of the best strategies is also going first, as it has around a 5% advantage of winning against the second player (assuming both are using the same strategy). You could, of course, use a coin flip to make this more fair, but as long as your opponent doesn't mind, any probability advantage is a great strategy.

Ultimately, this is a game of luck, so no strategy guarantees a win. In fact, it is possible to win a game in a single turn, never holding a single value. The probability of this is around 0.0105 (probability of not rolling a 1 for 25 turns, also tested with 10,000,000 simulations in `pigsimprob.py` [simulating this took over 10 minutes]), but to guarantee a win, you must not roll a 1 at most 50 times, which has a probability of 0.0001. So, if you're feeling lucky, just keep rolling!

References

Numberphile. (2021, April 28). *The Math of Being a Greedy Pig* [Video] YouTube. https://www.youtube.com/watch?v=ULhRLGzoXQ0

Neller, Todd W. and Clifton G.M. Presser. (2004) *Optimal Play of the Dice Game Pig* [PDF] Gettysburg College. https://cupola.gettysburg.edu/cgi/viewcontent.cgi?article=1003&context=csfac

Neller, Todd W. and Clifton G.M. Presser. (2010) *Practical Play of the Dice Game Pig* [PDF] https://cupola.gettysburg.edu/cgi/viewcontent.cgi?article=1006&context=csfac

Dylan Gonzalez (2024) *The-Game-of-Pig (contains programs used)* [Github Repo] https://github.com/captainmelic/The-Game-of-Pig