Monte Carlo, Markov Chains, and Monopoly

Evan Henke

December 16, 2016

Abstract

Monopoly is a classic board game that relies on randomness, in the form of rolling dice, to move players around a board. Players will buy properties and eventually one player will win because they have gained the most money. Using Monte Carlo simulations and Markov Chains, this paper will show that there exists a function that converges to an optimal probability that is dependent on the other players' probabilities to buy on a given side of the board. But although this probability exists, it is not calculable in a quick sitting, or player turn, and due to the nature of *Monopoly's* gameplay, it is safer to go by a simple rule like that of buying 9 times out of 10.

1 Introduction

Whether you enjoy the game or not, everyone in their lifetime has played *Monopoly*. It has been around for over 70 years and in that time has become one of the worlds most played and most well known board games. The game pits a few friends against each other to see who can make the most money. Every player has a pawn that they will move around the board and purchase the spaces or properties that they land on, the distance each pawn moves depends on the outcome of a roll of two dice. Naturally, this causes the game to rely heavily on randomness because a player can choose if they want to purchase a property, but they cannot directly choose which property they are going to buy. Therefore, much of the game can be boiled down to one, true question each player needs to answer: "Should I buy the property I just landed on?" And the goal of this paper is to answer just that. More mathematically speaking, at what probability should the player purchase a property, knowing which side of the board they are on.

1.1 History and Gameplay

People have been playing *Monopoly*, or games similar to it, for close to a century. But most people are unaware that *Monopoly* is a derivative of a different board game called *The Landlords Game*. Originally created by a woman named Elizabeth Magie, *The Landlords Game* was made as an educational tool. Magie was

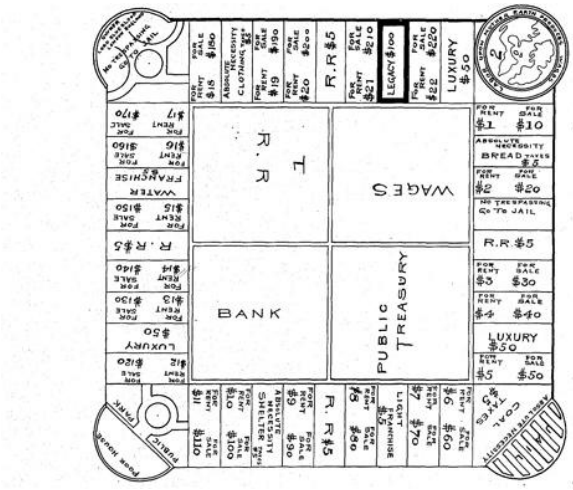

Figure 1: Early rendition of *The Landlord's Game*

a Georgist, and the staple belief of the Georgist ideology is that economic value coming from land should be divided equally among every member of society. Compounding on this, she held the belief that rent served only one purpose: to enrich the landlord at the expense of the tenant. She wanted to teach young children this ideology so that they would grow up to be "level-headed adults" but she thought that the concept was too tough to grasp without a good example, therefore she decided to create a board game to demonstrate her ideals.

Magie had her game patented in 1904 and she approached Parker Brothers to produce it. Parker Brothers refused due to the games complexity but she was determined and she was able to have her game produced by smaller publishing companies and over time the game spread. But, with popularity comes imitation, and soon enough many others would try to replicate The Landlords Game and its success. The most notably being *Monopoly*, which ironically, was produced by Parker Brothers and claimed to have been created solely by a man named Charles Darrow.

Parker Brothers began production of *Monopoly* in 1935, this is when it began to take the form that most people recognize, with the iconic mascot Rich Uncle Pennybags following a year later. The board has nine spaces on each side with one space on each corner: Go, Jail, Go to Jail, and Free Parking, for a total of 40 spaces. Each space on the board is different, there are five properties available for player purchase on the first and four side and six on the second and third side. There are two sets of color coded properties on each side of the board and if one player owns both or all three properties in the set, then they are said to have a monopoly at that location on the board. There is also one utility property on each side of the board as well as three Chance spaces, three Community Chest spaces, and two Tax spaces scattered around the board. Also in the box are 32 house pieces and 12 hotel pieces.

Every player will begin the game with 1200 dollars and their pawn will start on Go and, on their turn, the player will roll two dice and move the number of spaces indicated by the sum of their roll. Depending on where the player lands due to their dice roll, they would have to take a certain action. Chance and Community Chest spaces make the player draw a card and take an action depending on the card text, whereas the Tax spaces requires the player to simply lose the amount of money indicated on the space on the board. If the player lands on a property then one of two things will happen depending on whether or not the property is owned. If no one owns the property, whatever player landed on it has the opportunity to purchase it. If they decide they want the property, they pay the indicated price on board and add that property to their collection, but if they decide not to buy it then the property is put up for auction and any player that would want to purchase that property can put a bid down for it. Whoever is willing to pay the most will purchase the property. If the player lands on a property that is owned by another player, they must pay a sum of money to the owner of the property that depends on whether or not the owner has a monopoly at that location. In the event that the owned location is part of a monopoly, the amount of money owed will still vary depending on whether the monopoly has houses or a hotel.

When a player owns a monopoly, he or she has the option to purchase houses on those properties. Each individual property will specify how much a house will cost and the player must be able to afford to put a house on every property in that monopoly if they opt to purchase houses. For example, if a player owns a monopoly with three properties: property A requires 100 dollars for a house, property B requires 110 dollars, and property C requires 120 dollars. The player must be able to pay the total of 330 dollars to be able to place a house on each property in that monopoly. The amount of money the owner gains from an opposing player landing on the space increases for every house placed on that property up to a maximum of four houses. If a player has four houses on each property in their monopoly, he or she can choose to purchase a hotel to be placed on each property. Hotels function very similarly to houses in that the player must be able to afford to buy enough hotels as to have one on each property and they will increase the amount the owner gains if an opposing player lands there. But when a hotel is purchased, the four houses are removed from the property and will go back into the house pool.[1]

There a few other important spaces on the board. On each side there will be one of four utility tiles, these spaces function almost identically to that of normal monopoly properties in that a player earns more if they own more of those spaces, with the exception that these spaces cannot have houses or hotels. Go is where players start and whenever a player lands on Go or passes Go they

[1]This differentiation is important because a popular strategy is to upgrade a player's monopoly to have four houses and but never upgrade to hotels. The number of houses is finite, so if a player can build four houses on one monopoly they have the potential of holding twelve of the 32 houses on only one set of properties, effectively monopolizing the available houses as well as the properties they are on, but the action of purchasing a hotel will return those houses back to the available pool.

are given 200 dollars, unless stated otherwise by a card action. Free parking is a free space and nothing happens when a player ends their turn at that location. The Go to Jail sends the player to Jail if they end their turn there. Jail is the final special space that needs mentioning because it is fairly important. Jail is the second corner that a player will pass when rounding the board and it is two spaces in one: In Jail or Just Visiting. If the player is moving by normal means onto the jail space they are said to be just visiting and nothing special happens, it is similar to free parking in that way. But if the player lands on the Go to Jail space or are sent to jail by a card action, then their pawn is moved to the Jail space and they are said to be in jail. If a player is in jail they can do one of two things, roll the dice or pay 50 dollars. Paying 50 dollars will place them into the Just Visiting part of the space and they can continue to play on their next turn. Or the player can roll the dice and if they roll doubles then they leave jail as if they had paid the 50 dollars. If they fail the roll then they stay in jail and can retry the roll on their next turn or pay the 50 dollars. A player can leave jail after their third turn of being in jail.

If a player lands and an owned property and does not have the funds to pay the owner, they can mortgage their properties. This means they gain a small sum of money determined by the property that was mortgaged. A mortgaged property is still owned by the same player, but the owner cannot gain income from that property; in other words, if a player lands on a mortgaged property, they are not required to pay the owner.

A large portion of the game that is not governed by the rule book is called player trading. On a players turn, they can propose a trade with another player for properties or cash. There are no true stipulations to how this trade is conducted and no player is required to accept an offer. This gives each game of Monopoly more variety but is extremely difficult to model due to the individuality of players.

Monopoly is not advertised as a sinister game, it is portrayed as vibrantly, fun, family game where to goal just happens to be to have the most money at the end of the game. But it quickly becomes apperent that the game is more devious than the players are lead to believe. Many people do not enjoy *Monopoly* due to how the game's randomness as well as how the competition works. Unlike many other competitve board games, *Monopoly* relies heavily on some-what of a truce system in trading mentioned above. Players will make these trades and deals with other players, but there can be only one winner so at some point truces are broken and this can disuade people from playing this game often, or even more than once. So, despite the game's advertisements, many of Magie's beliefs still lives on in many peoples' distaste for the game's ruthlessness.

2 The Simulation

There are a number of things that complicate simulating *Monopoly* that does not directly affect the games main function, which is to roll dice to move around

the board and to purchase properties. Therefore houses, hotels, Community Chest cards, Chance cards, property auctions, mortgaging, and trading have been removed to give a more bare-bones approach.

2.1 Monte Carlo Methods

One will notice that there are many conditionals within the rules, which can cause difficulties with the modeling approach, and this is why Monte Carlo simulations will be beneficial. Our goal is to find some optimal probability to purchase a property on the board, but this is problematic because it depends on a very large number of factors, including each individual players probability to buy as well as which states are available, how much money the current player has, etc. In *Monopoly*, these previously mentioned variables rely, directly or indirectly, upon the probability of being at a specific spot on the board at a certain time. A textbook optimization problem, in most instances, could be solved using some arithmetical approximation akin to Newton's Method. But in this case a distinct function, f is not apparent, let alone finding it's derivative.

Monte Carlo simulations stress that a mathematical model can be solved using random numbers, which lends itself perfectly to this predicament. Implementing the idea that the use of random numbers can lead to a non-random solution, we can begin to simulate games of Monopoly with random purchasing probabilities in the hopes to find optimal purchase strategies.

2.2 Markov Chains, Expected Values, and Finite-State Machines

So far this is an extremely general starting point, it has been established that there are going to be a huge number of games played, but now what? Following the rules in playing order says that the first thing a player is going to do is to roll two dice to move their pawn around the board, so there must be a way to emulate the movements of each pawn.

A Markov Chain is a stochastic process that is composed by a set of states that follow the *Markov Property*, which can also be described as being *memoryless*. In terms of a Markov Chain, this means that the future state depends only on the current state and not the *history*, or previous set of states.

A Markov Chain can be envisioned as a group of points on a graph, called *states* each state has *transition probabilities* associated with it that will dictate the probability of transitioning from the current state to the specific state. For example, $p_{i,j}$ would be the probability to go from state i to state j. All of these probabilities are part of a *Transition Matrix*, P where:

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & p_{0,j} \\ p_{1,0} & p_{1,1} & p_{1,2} & \cdots & p_{1,j} \\ p_{2,0} & p_{2,1} & p_{2,2} & \cdots & p_{2,j} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{i,0} & p_{i,1} & p_{i,2} & \cdots & p_{i,j} \end{bmatrix}$$

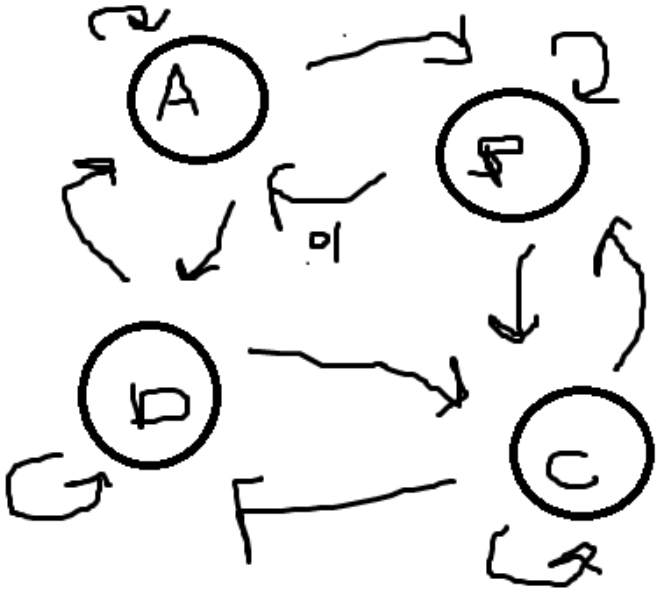

Figure 2: Example of a Markov Chain with four states and associated probabilities to change state

At a high-level, a *finite-state machine* is a model used for software design that, as the name implies, assumes a that the program will have a finite amount of states and that it will only be in one state at a time. For example, if someone needed to model the motion of a door then that person could define the entire system of the door as a finite-state machine with only two states: the door is open or the door is closed. And then there is some sort of function or method that will cause the finite-state machine to switch states. In this example, when the door is in the closed state it would most likely have a unction that opens the door and the opened state would have a corresponding function to close the door.

We can think of the *Monopoly* board as a finite-state machine. The first step is to define each property on the board to be a state, giving a finite total of forty states. A player moves between two to twelve spaces away from where they are currently positioned on the board. This fits with the definition of *memorylessness* because the state the player is going to move to depends only on their current position. From there, we can extend this to a finite-state machine because at no time can a player be on more than one state.

Finally, the transition matrix, P, must be defined. P will be in the form:

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & p_{0,39} \\ p_{1,0} & p_{1,1} & p_{1,2} & \cdots & p_{1,39} \\ p_{2,0} & p_{2,1} & p_{2,2} & \cdots & p_{2,39} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{39,0} & p_{39,1} & p_{39,2} & \cdots & p_{39,39} \end{bmatrix}$$

To give values for each $p_{i,j}$ we have to consider the expected outcomes when two dice are rolled. The probabilities of getting a specific outcome on the dice roll can be shown in an array with the probability, p_i, of rolling the value of i, where i = 1,2,...,12.

$$p = \begin{bmatrix} 0 & \frac{1}{36} & \frac{1}{18} & \frac{1}{12} & \frac{1}{9} & \frac{5}{36} & \frac{1}{6} & \frac{5}{36} & \frac{1}{9} & \frac{1}{12} & \frac{1}{18} & \frac{1}{36} \end{bmatrix}$$

Using these probabilities we can create a 40x40 transition matrix for our board. For example, if a player is on the jail space, i = 10, then P_i being the i^{th} row of transition matrix P:

$$P_i = \begin{bmatrix} p_0 = 0.0 & ... & p_{11} = 0.0 & p_{12} = \frac{1}{36} & ... & p_{22} = \frac{1}{36} & p_{23} = 0.0 & ... & p39 = 0.0 \end{bmatrix}$$

One glaring difference is on index $i = 30$, this index indicates the Go to Jail space and will always have a probability of zero. To offset this, the Jail space will always gain that corresponding transition probability to simulate the action of moving the pawn to the jail space due to landing on Go to Jail.

Now that movement has been covered, we can build the rest of the simulation.

2.3 Bringing It All Together

The simulation is broken into three parts: single games, rounds, and overall averaging. A game, as the name implies, is only one game with a single winner. Whereas a round constitutes playing one thousand games. Every player will have a counter that will increase with how many games they have won per round.

Each game will start with four players, simply named Player One, Player Two, Player Three, and Player Four. Player One is the player we will be optimizing around and will act slightly different than the other three players. Each player will start on Go with 1200 dollars and an initial probability to buy that is randomly chosen between zero and one for each side of the board, with the exception that Player One will have a static initial probability to buy on each side of .50. The players will cycle around the board using the transition matrix P, mentioned earlier. The first and fourth sides of the board have five available properties to purchase and the second and third sides have six properties to purchase. The simulation contains two arrays, one that tells if the property is available for purchase and the other array stores the owner of the property if it is owned. When a player lands on a property the game will check both of these arrays for an owner, and if the property has no owner then the player will buy the property with that players respective probability to buy on that side of the board. If the property is owned then the player will pay the owner an amount according to the side of the board as well as how many properties the owner has in their possession. Each player will have a turn and if they land on a property that they cannot afford, they are removed from the game and their owned properties are returned to their unowned state and become available for purchase by the remaining players. This will go on until one player wins.

Once a round is complete whichever player has the most wins is deemed the round winner. At this point the winners probability to win will be saved and the probabilities of Player One will be be adjusted according to the following weighted adjustment:

$$p_{i+1,n} = \alpha_i p_{i,n} + (1 - \alpha_i) p_{winner,n} \tag{1}$$

Where p_i and p_{i+1} are probabilities to buy on side n ($n = 1, 2, 3, 4$) and α_i is a constant that will be varied for different simulations.

This encompasses a single simulation, but in the spirit of Monte Carlo methods, we cannot stop at just one simulation. The simulation is run thirty times and the final value for the buying probability for each of the four sides is saved. The average value of that final probability is used as a general average for it's corresponding side of the board.

Initially players two, three, and four have a buying probability between zero and one but later runs will use this average as a baseline and alter the probability to buy to see how the simulation reacts.

3 Results and Discussion

3.1 Expectations

To win in *Monopoly* a player must purchase properties and the more properties that the player owns, the more likely it is that an opponents land on the first player's properties. So, roughtly speaking, we would expect that the probability to buy on a side to be high. Furthermore, due to differing property values and differing number of properties on a given side, we would expect the probability to buy to vary depending on the side of the board.

When Markov Chains are taught, it is not uncommon to use *Monopoly* as an example, so there is no shortage of information on *Monopoly*. But what is important for us is the steady-state vector for the board. In terms of a game of *Monopoly* the steady-state vector is a collection of probabilities(s_i where $i = 0, 1, 2, ..., 39$) of where a player will be at a given time. From the steady-state vector the relative frequency that a player is in Jail is by-far the largest, at .11724, the next closest is over seven percent less at .02990.[2]

This is important because we know the expected outcome of rolling two dice to be 7 and furthermore the expected outcome of rolling two dice over two turns to be 14. This means that players will be moving over sides two and three more than sides one and two. Therefore, we expect those two sides to be purchased at a higher probability than that of the first and the fourth side.

3.2 Initial Results

Simulations will vary in two aspects: the size of α_i and the *probability window*, $\omega_{c,s}$, which we will define as the difference between the maximum probability and minimum probability centered about probability c on side s. For example $\omega_{0.50,2} = .20$ would be defined as the window from 0.40 to 0.60 because it is of size 0.20 centered around the value 0.50 on the second side($\omega_{c,s} = 1$ implies that the window is the entire set of possible probabilities from zero to one).

The first step was to vary our α_i while keeping keeping a constant $\omega_{c,s} = 1$. Initially, we make the assumption that $\lim_{\alpha_i \to 1} \sigma_i = 0$ due to how p_{i+1} was defined. The results are kept in Tables 1, 2, and 3.

Table 1: Average Values for $\alpha_i = 0.9$ with $\omega_{c,s} = 1$

Side	Average p_i	Standard Deviation	Variance
1	0.8203074242	0.040839359	0.0016678532
2	0.8910287708	0.0276067073	0.0007621303
3	0.905966587	0.0204380301	0.0004177131
4	0.8785991185	0.0280977615	0.0007894842

As we had assumed, as α_i converges to 1, our σ_i shrinks. But the big surprise is how small σ_i is even with a full window, $\omega_{c,s} = 1$.

[2]This information comes from a presentation at Darthmouth College

Table 2: Average Values for $\alpha_i = 0.99$ with $\omega_{c,s} = 1$

Side	Average p_i	Standard Deviation	Variance
1	0.807603148	0.019313276	0.000373003
2	0.87750807	0.013404922	0.000179692
3	0.898804895	0.013240082	0.0001753
4	0.873359701	0.014231473	0.000202535

Table 3: Average Values for $\alpha_i = 0.999$ with $\omega_{c,s} = 1$

Side	Average p_i	Standard Deviation	Variance
1	0.789819981	0.004523793	2.04647E-05
2	0.863308345	0.002313867	5.35398E-06
3	0.887882061	0.002148036	4.61406E-06
4	0.861980883	0.002834187	8.03262E-06

As α_i gets closer to 1, more rounds need to be ran to appear to approach a convergent solution. On the low end 5000 rounds were ran for $\alpha_i = 0.9$ and 200,000 rounds for $\alpha_i = 0.999$. To give a more accurate graphical representation both $\alpha_i = 0.9$ and $\alpha_i = 0.99$ were ran 200,000 times for comparison to the $\alpha_i = 0.999$ values and those are pictured in Figures 3,4,5, and 6.

3.3 Varying the Probability Window

With such a small deviation the thought is to reduce $\omega_{c,s}$ around the average of their respective α_i and side to see what happens. For this experiment the use of only $\alpha_i = 0.99$ is sufficient because the deviation is already quite small and the amount of time to run the simulation is significantly lower than that of $\alpha_i = 0.999$. Setting $\omega_{c,s} = 0.4$ and $\omega_{c,s} = 0.2$ as the first two windows to evaluate centered around the general average for $\alpha_i = 0.99$. The results are in Tables 4 and 5, respectively.

Table 4: Average Values for $\alpha_i = 0.99$ with $\omega_{c,s} = 0.4$

Side	Centered Around c	Average p_i	Standard Deviation	Variance
1	0.807603148	0.860462447	0.007753085	0.000060110334
2	0.87750807	0.898130494	0.005789447	0.000033517693
3	0.898804895	0.914810143	0.004411178	0.0000194585
4	0.873359701	0.8978265	0.004674069	0.0000218469199

The first thing to notice is that the average values are not the same across different probability windows. By using the general average as the center of the window the final result was not the same value as the general average, it was not even within a standard deviation of that value. As a continuation of this experiment we will set the value of $\omega_{c,s}$ to be a constant that is center about a much lower value than that of our general average in such a way that the

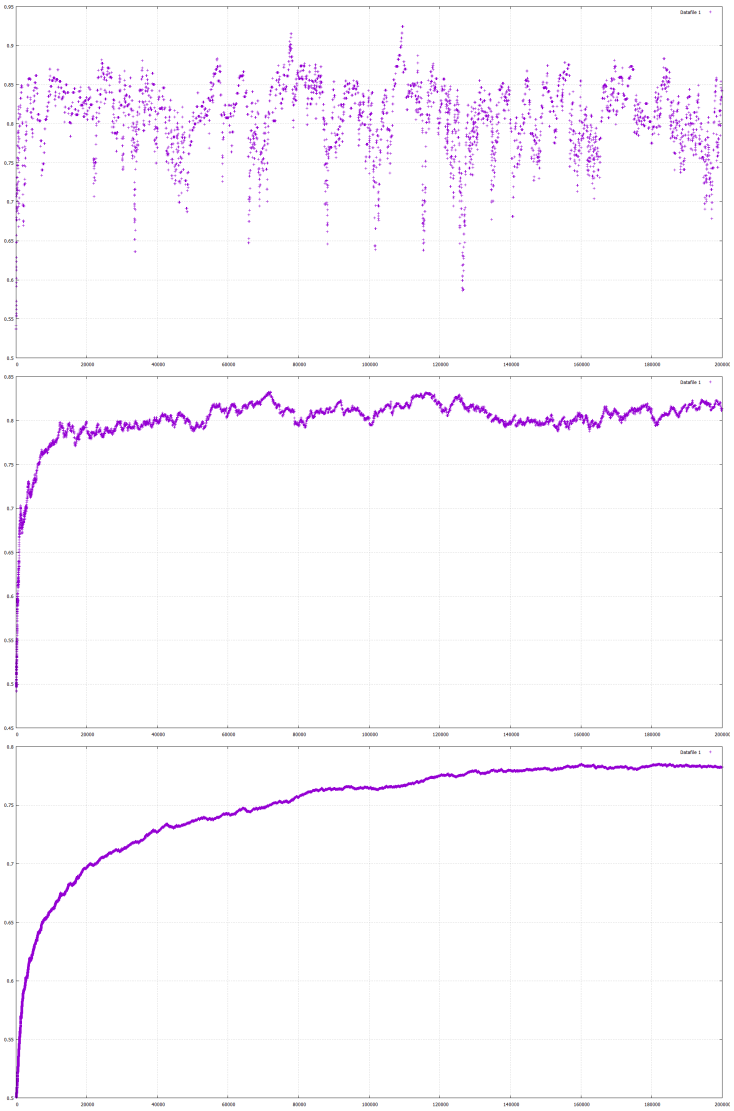

Figure 3: Graph of p_i in one simulation over 200,000 rounds. From top to bottom, $\alpha_i = 0.9, 0.99, 0.999$ on side one

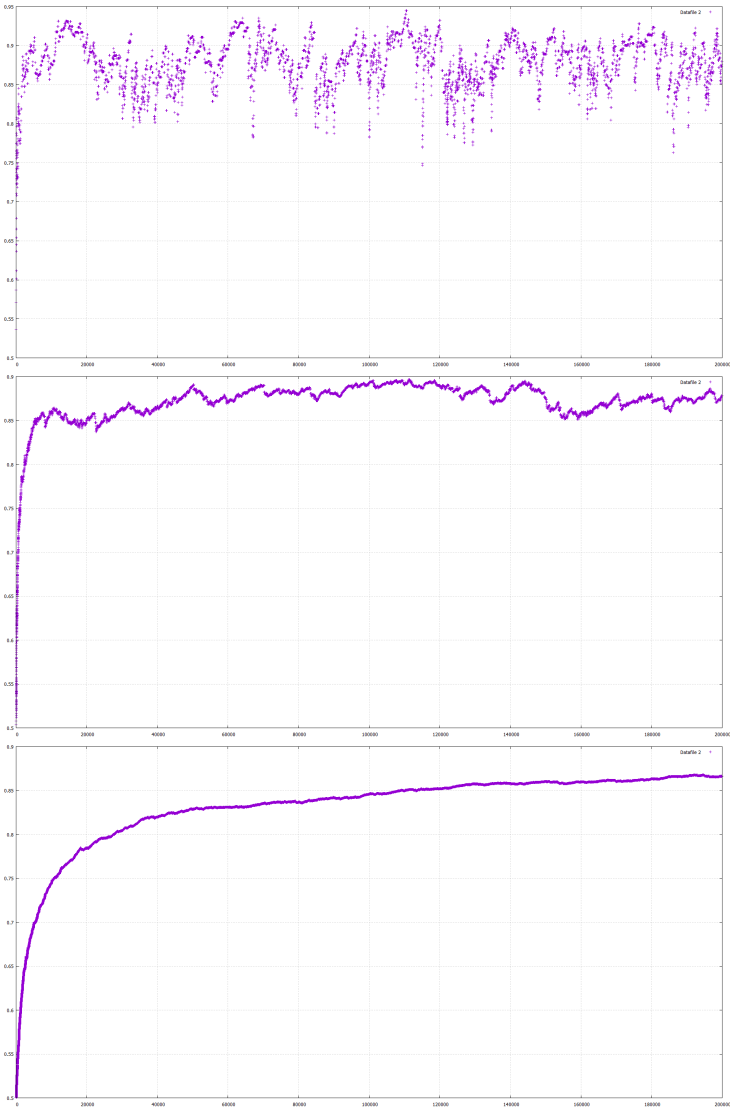

Figure 4: Graph of p_i in one simulation over 200,000 rounds. From top to bottom, $\alpha_i = 0.9, 0.99, 0.999$ on side two

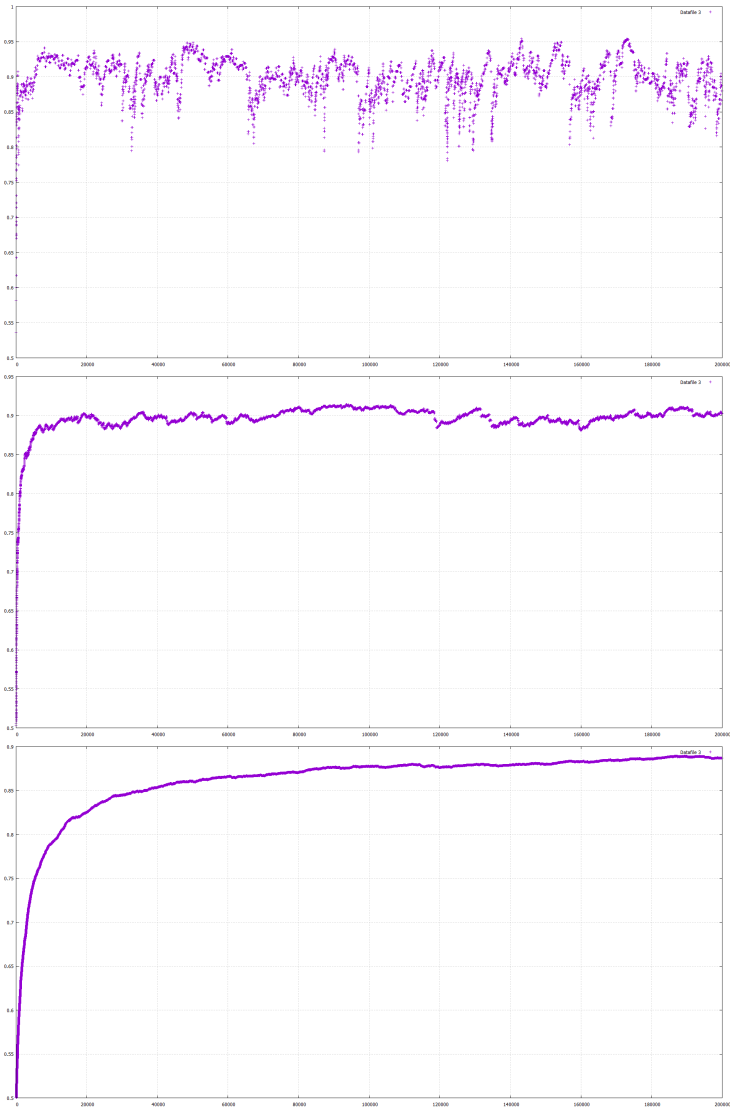

Figure 5: Graph of p_i in one simulation over 200,000 rounds. From top to bottom, $\alpha_i = 0.9, 0.99, 0.999$ on side three

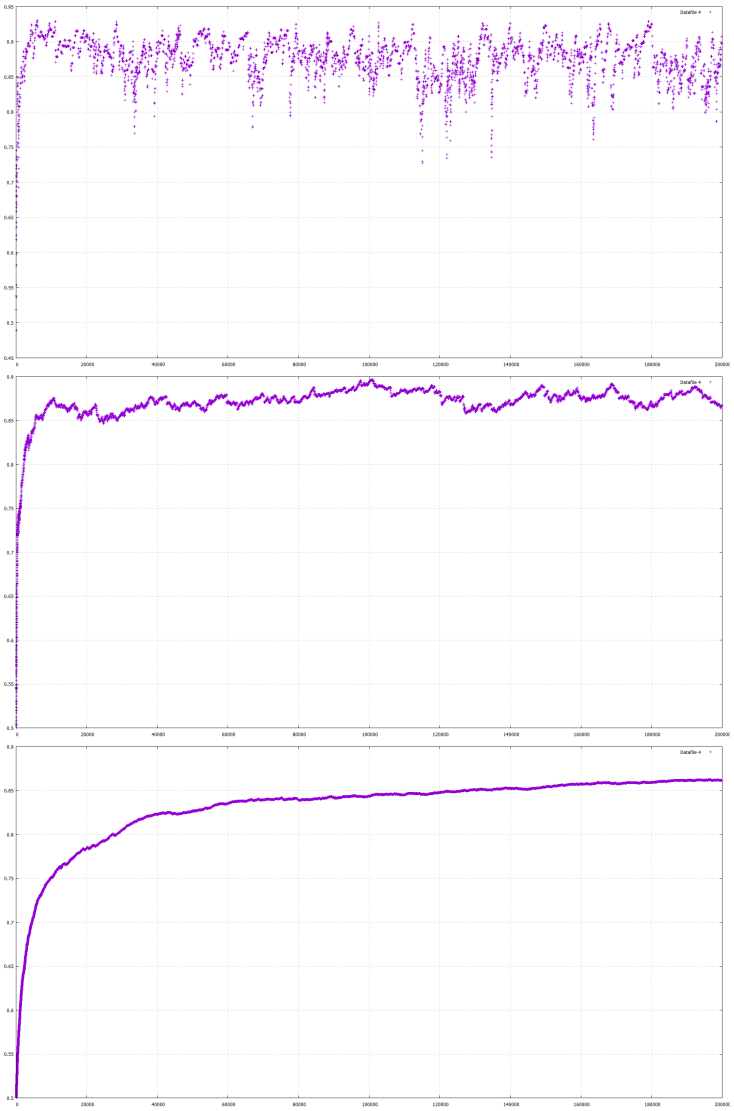

Figure 6: Graph of p_i in one simulation over 200,000 rounds. From top to bottom, $\alpha_i = 0.9, 0.99, 0.999$ on side four

Table 5: Average Values for $\alpha_i = 0.99$ with $\omega_{c,s} = 0.2$

Side	Centered Around c	Average p_i	Standard Deviation	Variance
1	0.807603148	0.823715044	0.001086344	1.18014E-06
2	0.87750807	0.898156638	0.001364272	1.86124E-06
3	0.898804895	0.924639045	0.001033281	1.06767E-06
4	0.873359701	0.898555805	0.001166616	1.36099E-06

general average is not in the probility window. If we continue to suspect that the optimal probability is in a close vicinity to the general average then the experimental average for this simulation would be the maximum value of the window.

Table 6: Average Values for $\alpha_i = 0.99$ with $\omega_{c,s} = 0.2$

Side	Centered Around c	Average p_i	Standard Deviation	Variance
1	0.40	0.427064312	0.003882672	0.00001507514
2	0.45	0.492191439	0.003539403	0.00001252737
3	0.50	0.550079608	0.002700408	7.2922E-06
4	0.45	0.495842442	0.00285502	8.15114E-06

Table 6 shows the output for the lowered center and Figure 7 graphs the results. These results imply that the optimal value is not equivalent to the general average but that it is related to the endpoints of the probability window. More specifically, that there is a convergent value for a player's probability to buy a property that depends on the probabilities of the other players. This can be portrayed as a function, f where.

$$p_{optimal,s} = f(\omega_{c,s}) \tag{2}$$

3.4 Relevance

We have a basic function that converges to an optimal probability to purchase a property on a side is dependent on the probabilities of the other players, i.e. the probability window $\omega_{c,s}$, but what does this this mean to the player? *Monopoly* is a family game and can be an interesting modeling problem, but it is not played in a competitve fashion in the same way that games like *Blackjack* or *Texas Hold'em* are. And the average player is not going to be looking for a hard and fast percentage of when they should buy because when the game is being played, it would be in a friendly manner.

That being said, if it is assumed that the opponents are competent players, meaning they know that they must buy at a high percentage to win, then the main player should follow suit and purchase at a high probability. A safe ratio of would be to purchase a property 8 or 9 times out of 10 assuming they can afford the property.

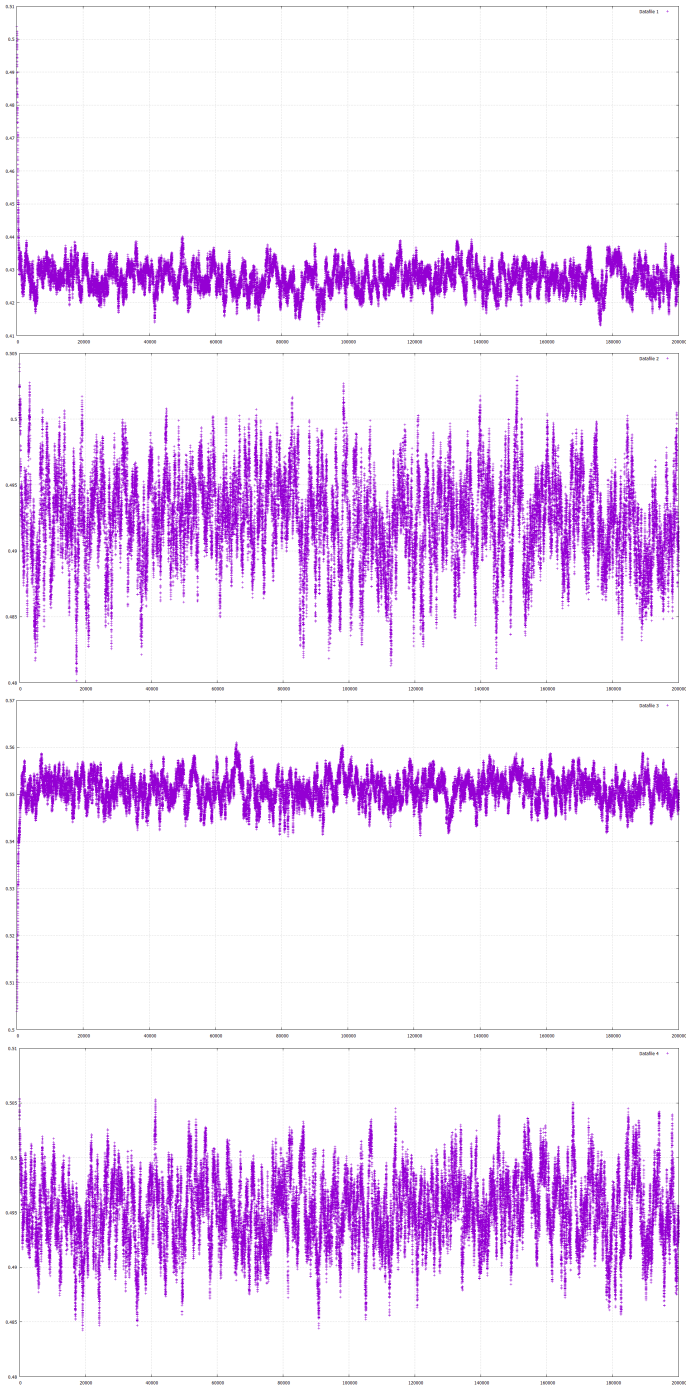

Figure 7: Graph of p_i in one simulation over 200,000 rounds with.

3.5 Other Comments

At the start of a game of *Monopoly* every player starts with 1200 dollars and because the amount of money a player has directly affects how many properties they can purchase any alteration to this create a noticable change in the probabilities to buy. Logically we would expect that if a player starts with less money they are going to favor less expensive properties due the possibility of losing quickly if they purchase an expensive property. And vice versa, we would expect players to favor more expensive properties if they start with a larger amount. Varying amounts for starting money are shown in Tables 7 and 8.

Table 7: Starting Money = 500

Side	Average p_i	Standard Deviation	Variance
1	0.813392354	0.007994897	6.39184E-05
2	0.849089753	0.01109538	0.000123107
3	0.833370982	0.008402483	7.06017E-05
4	0.659255734	0.01728581	0.000298799

Table 8: Starting Money = 1000

Side	Average p_i	Standard Deviation	Variance
1	0.788698681	0.012049833	0.000145198
2	0.859592281	0.007973815	6.35817E-05
3	0.880384151	0.008646778	7.47668E-05
4	0.840693321	0.008964628	8.03646E-05

Table 9: Starting Money = 1500

Side	Average p_i	Standard Deviation	Variance
1	0.782533218	0.01132606	0.00012828
2	0.861558466	0.00719153	0.00005171811076
3	0.8890913	0.007156836	0.000051220294517
4	0.876152387	0.006005488	0.0000360658809

The interesting, although not shocking, result here is that the player will tend to have more success with the "safer" properties and avoid the "risky" properties. Safe being the properties on the first side because they cost the player less to own and risky being the properties on the fourth side because they have a large payoff, but if the player is unlucky they can lose on one bad dice roll.

On another note it is worth mentioning that had auctioning been simulated as well that these results would most likely change drastically. If a player decides they did not want to purchase a property any other player can place a bid on it with the minimum starting at a single dollar. Therefore it is in everyone's interest to bid on this property because there is the potential to purchase

as a reduced price and the profit margin would be larger. This was removed because, similarily to player trading, it is extremely hard to accurately mimic the thoughts of individual players and the difference between players is so large that a modeling approach would not be nearly accurate enough to result in a meaningful solution.

4 Closing

In *Monopoly* players move around a board and purchase properties. It has been shown graphically and numerically that there is a a function, $f(\omega_{c,s})$ that converges to some optimal probability $p_{optimal,s}$ dependent on the other player's probabilities to buy. But due to the inate randomness of *Monopoly* and the environment it is played in, it would typically not be worth a player's effort to calculate this optimal while playing the game and a simple rule is to buy close to 9 out of 10 times regardless of side.