Modelling Player Strategy in Monopoly

Tiarnán Curran-Feeney

April 23, 2021

Abstract

A modelling of player strategy in the board game Monopoly and using it to approximate an optimum level of riskiness to give you the best chance of winning depending on your opponents' characteristics and play style.

Contents

1 Motivation

Monopoly is a strategy game requiring people to take risks in the form of investments for the chance to earn money later in the game. A full description of how the game works is available in section 2.

This raises the question is there an optimum level of riskiness for success? Secondly, does the optimum level of riskiness depend on the riskiness of your opponents?

We will consider different factors that could impact the answer to these questions such as the number of opponents, the length of the game, and the riskiness of your opponents. For the purposes of this project, we will define a player's riskiness as their probability of buying a site or developing a site.

We might expect to find that competing against risky players requires you to be more strategic than risky as there is a higher chance of you going bankrupt if your opponent dominates the board. By contrast, if your opponent takes very few risks then maybe it pays off to exhibit risky behaviour, as there will be fewer sites that your opponent owns.

Additionally, if a game is particularly long and your opponents are risky then perhaps it pays off to be risky at the beginning but then take less risks later in the game. Since investments allow you to earn money later in the game, a short game might mean it is better to take less risks as the game is too short for the risks to pay off.

In this paper, we will attempt to model player strategy with the goal of determining the optimum risk level, if it exists, for your best chance of winning in Monopoly.

2 Monopoly: Rules and Gameplay

2.1 Gamplay

Monopoly is a game playable with 2-6 players, with the goal of earning the most money by the end of the game. Each player starts at Go and receives £1,500.

The game begins by each player rolling the pair of dice in turn and the person who rolls the highest number gets to go first. They roll again and move forward by the sum of the two dice. Players then continue to take turns going clockwise around the group.

A double is when both dice display the same number. In this scenario the player gets to roll again.

Finally, when a player becomes bankrupt they must cease playing. The game ends when only one player remains or everyone has rolled a set number of times.

2.2 The Board

The board is made up of 40 squares. They are made up of sites, community chest and chance card squares, jail, go, free parking, go-to-jail, and tax squares.

Figure 1: A bird's eye view of the monopoly board.

2.2.1 Sites and Housing

Figure 2: Some site cards.

When a player lands on a site they may purchase it for the cost stated. From this point on, if a different player lands on that site they will be required to pay the owner the rent stated on the site's card.

Sites are grouped into sets by their colour. If a player owns an entire set then they may develop sites by purchasing up to 4 houses or 1 hotel, for the indicated price on the site's card, which increases the rent for that site. Additionally, the rent is now doubled for undeveloped sites.

There are two special sets which are not possible to develop. Those are the train stations and the utilities. The rent paid on a train station is $25 * 2^i$, where i is one less than the number of train stations that the player owns. The rent paid on a utility is four times the dice roll if only one utility is owned and 10 times the dice roll if both utilities are owned.

The remaining positions on the board are not possible to purchase.

If a player chooses to do so they may mortgage any sites they own for the indicated mortgage value on the card. While a site is mortgaged, other players are not required to pay rent when they land on that site. A site can be un-mortgaged for 1.1 times the mortgage value. In this model we will not be considering mortgaging. Once a player goes bankrupt all their sites are returned to the bank and they are out of the game.

2.2.2 Community Chests and Chance Cards

When a player lands on a community chest or a chance square then they must pick up the community chest card or chance card respectively from the top of the deck. They will then be required to follow the instructions on the card. These can be rewards or losses for the player.

2.2.3 Jail

There are two possible states when on the jail square. Those being "in-jail" or "just visiting". If a player lands on jail then they are deemed to be "just visiting" and as such they are unaffected. Meanwhile, a person "in-jail" must satisfy certain conditions before they are able to leave the square.

When a player is "in-jail" they must remain there until either they pay £50 to be released early, they roll a double, they have a get out of jail free card, or they fail to roll a double 3 times in a row so must pay £50 to be released. With the exception of rolling a double, all methods of getting out of jail end your turn.

There are three ways in which a player can end up "in-jail".

- Landing on the go-to-jail square
- Rolling three doubles in a row
- Receiving a go-to-jail card when picking up a community chest or a chance.

2.2.4 Other Squares

- Go is the initial starting square for all players and every time you pass it you collect £200 from the bank.
- If you land on a tax square then you must pay the amount stated.

- If you land on Free Parking then you receive all the money that people paid when landing on tax squares or from payments made due to community chest or chance cards.

2.3 Notable Statistics

Due to the go-to-jail square, community chest cards, and chance cards the expected frequency of landing on any given square is not uniformly distributed. The exact frequencies would be too difficult to calculate. However, we can approximate the frequencies by having a player roll the dice 10,000 times and recording what squares they land on. We receive the following distribution shown in 3.

As we would expect, the frequency of being on jail's square is double other square's frequencies as square 30 sends us directly to jail. Adding to this, squares that are between jail's square and square 30 generally have a higher frequency of being landed on due to people going round the board and also people coming from jail.

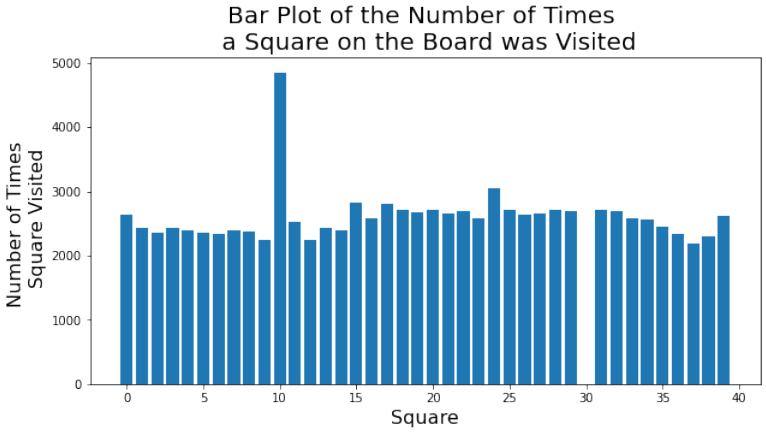


Figure 3: Frequency of Landing on each square

Another interesting statistic is that the probability of a game of monopoly never ending (i.e: at least two players never go bankrupt) with a 95% confidence interval is 0.12 ± 0.01; this was calculated in [1].

3 Constructing a Model

With the above rules in mind, we can now begin constructing our model.

3.1 Assumptions

In order to simplify the process, we shall assume that players will buy at most one house per turn. This reduces the decision of how many houses should the player buy to a binary choice.

In this model, we will also make the assumption that players are not able to trade sites or interact with each other outside of the base rules previously defined. Furthermore, players will be able to check the board state and can, therefore, check what sites each player owns. However, a player is unable to see how much money another player has.

It was noted in [2] that people who suffer from anxiety are more risk-averse than people who do not suffer from anxiety. While we are not modelling players with anxiety, we can build upon this concept for our model. More specifically, we can make the assumption that players will be less anxious, or more risky, when they have a lot of money than when they have less.

We also observe in [3] that fearful people are more risk-averse than angry people. We can incorporate this into our model by considering character types of players. With players who are naturally more angry having more risky characteristics and players who are naturally more fearful having less risky characteristics.

We cannot ignore that players' personas will have an impact on each other. We can model this by assuming that two risky players will influence an anxious player to be more risky. Similarly, two anxious players will influence a risky player to be more anxious.

3.2 Machine Learning vs Rule Based System

Now we must decide the type of model we will construct. The first decision being whether to use machine learning or a rule based system.

The key advantage of machine learning is that we can construct a model far more complex than what we could derive ourselves and without the need to make any assumptions. However, with a rule based system we know the decision process that is being made at each step and can closer examine it.

Furthermore, our goal is to model player strategy in monopoly and machine learning will perform the optimum strategy each turn. This assumes that our players are perfect strategists which is not always a logical assumption as stated in [2].

So, while rule based systems always require some simplification of the real system, they are the best choice in this scenario. However, a player who

learns to play through machine learning would make an interesting addition to this project.

3.3 Our Model

3.3.1 Initial Concept

While rule-based systems are typically time based, Monopoly is a turn based game and, therefore, system changes are independent of time. This naturally leads us to a discrete approach for our model. A player's rules shall update at the start of every turn to adapt to any changes in the state of the board.

When updating the rules there are certain factors about the board and player that we will want to consider. A representation of those factors is shown in figure 4. We will want to have rules that change the type of strategy our player has with rates that are dependent on the current rent prices on the board, the sites that are owned, and the type of strategy that our player currently has. Though not shown in figure4, we will also want to consider the amount of money the player has when computing the reaction rate.

To start with lets consider the following reactions for an individual player, where α and β depend on the factors we mentioned previously. We now have a basic model for a player that is able to change strategy depending on the current system state.

R: Risky

A: Anxious

$$A \underset{\beta}{\overset{\alpha}{\rightleftarrows}} R$$

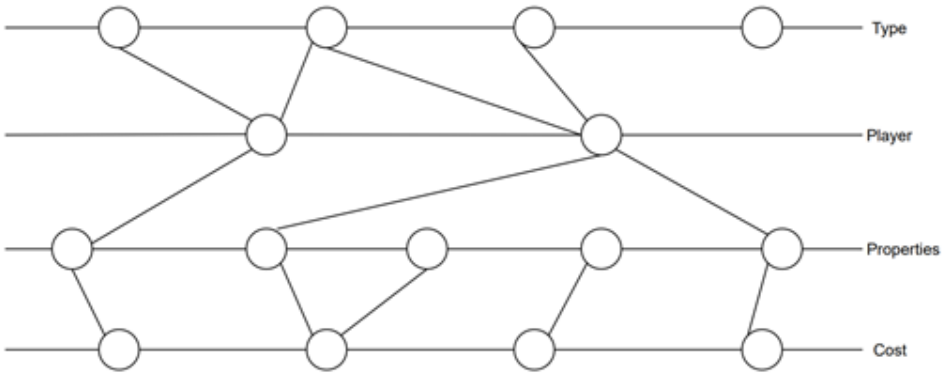


Figure 4: Simplified hierarchical classification of what we have to consider when applying rules in our model. Cost is the amount of rent paid when landing on a site and type is the type of player.

3.3.2 Refinement

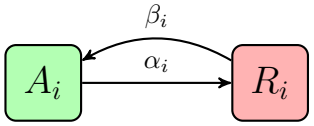
While this model is a nice start there is a lot we can do to improve upon it. Currently we only have two states players can be in, Anxious and Risky. To improve upon this we can think of a continuous scale from Anxious to Risky rather than a boolean case of Anxious or Risky.

This will require us to alter how we define the system. Instead of thinking about players changing between Anxious and Risky, we shall now consider every player having a Risky attribute R_i for each player i. More specifically, we will consider R_i to be lying in the interval zero to one, where zero is 100% anxious where you never take risks and one is 100% risky where you always take the risk.

Now let $A_i = 1 - R_i$. This allows us to build upon our previous equations to give.

$$R_i : \text{Riskiness of player } i$$
$$A_i : \text{Anxiousness of player } i$$

$$A_i \underset{\beta_i}{\overset{\alpha_i}{\rightleftarrows}} R_i, i \in [4](\text{ where } [n] = \{1, 2, 3, 4..., n-1, n\}\ \forall n \in \mathbb{N})$$

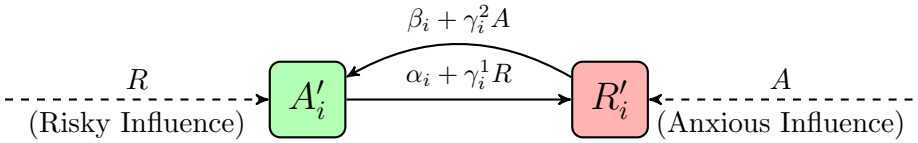


We now have quite nicely derived a model that can account for our assumption about riskiness and anxiousness. We can further extend this to account for our assumption of how characteristics (i.e: fear and anger) impact strategy. To do this we can consider a lower and upper bound on our riskiness, R_i^L and R_i^U, that is dependent on whether our player is fearful or angry. We shall do this by rescaling R_i and A_i, and performing the reactions on the rescaled $R_i' = R_i - R_i^L$ and $A_i' = A_i - (1 - R_i^U)$ instead.

Adding the following two equations will implement our assumption that players' personas will impact each other.

$$A_i' + R_j + R_k \overset{\gamma_i^1}{\to} R_i' + R_j + R_k, j \neq k \in [4] \backslash \{i\}$$

$$R_i' + A_j + A_k \overset{\gamma_i^2}{\to} A_i' + A_j + A_k, j \neq k \in [4] \backslash \{i\}$$



To finalise our model, we must now define our α,β,γ_1, and γ_2. First we shall define the following.

C_i : The sum of all rent on the board that player i would have to pay
M_i : The amount of money player i has
W_i : A constant representing how much player i values money
R_i^L : Lower bound on R_i based on characteristics of player i
R_i^U : Upper bound on R_i based on characteristics of player i

We need to create an α_i that will be greater the larger M_i is and lower the larger C_i is. This leads us to the following definition. Since M_i and C_i should do the opposite to β_i, we can take β_i as defined below. Since γ_i^1 and γ_i^2 are independent of C_i and M_i, they can be a constant.

$$\alpha_i = Max(\frac{M_i - W_i C_i}{M_i + W_i C_i}, 0)$$
$$\beta_i = 1 - \alpha_i$$
$$\gamma_i^j : \text{ arbitrary constant, } j \in [2]$$

This was not the original equation for α. If you would like to see the different equations that were considered, see Appendix A.

3.3.3 Final Model

$$R_i, A_i, R_i^L, R_i^U, \alpha_i, \beta_i, \gamma_i^1, \gamma_i^2 : \text{ As previously defined}$$

$$\begin{aligned}
[R_i', A_i'] =& [R_i - R_i^L, A_i - (1 - R_i^U)], && i \in [4] \\
A_i' \underset{\beta_i}{\overset{\alpha_i}{\rightleftarrows}} & R_i', && i \in [4] \\
A_i' + R_j + R_k \xrightarrow{\gamma_i^1} & R_i' + R_j + R_k, && j \neq k \in [4] \backslash \{i\} \\
R_i' + A_j + A_k \xrightarrow{\gamma_i^2} & A_i' + A_j + A_k, && j \neq k \in [4] \backslash \{i\} \\
[R_i, A_i] =& [R_i' + R_i^L, A_i' + (1 - R_i^U)], && i \in [4]
\end{aligned}$$

3.4 Coding

Now that we have derived our model we can now begin to create it in Python. In this section we will only be discussing the modelling decisions present in the code. If you are interested in looking deeper into the structure of the program, all code will be available in the Jupyter Notebook.

We begin by producing the classes board and player. These classes are used to implement the base gameplay and rules of the game. We will never use the player class directly, but all other player classes will inherit from it.

We still need to create a decision process so that players can choose whether to buy houses or sites. This is where the randPlayer class comes in to play. We can interpret a player's riskiness, R_i, as the probability of doing an action.

It is possible that a player's likelihood of taking a risk on buying a house will be different to taking a risk on buying a site, as site purchases are an early game decision while purchasing houses is a decision made later in the game. We shall account for this by redefining $R_i = [p_i, q_i]$, where there are now two factors of riskiness. The probability the player buys a site they land on, p_i, and the probability a player buys a house at the start of a turn, q_i. We finish the decision to purchase a house by shuffling the sites that they can buy houses for and in the event they decide to buy a house the first site in the list is chosen. Hence, the more risky a player the more likely they are to buy sites or houses.

For the randPlayer class, a player's risk is constant. This gives us a good building ground for our model. We shall do this by implementing the stratPlayer class. This class builds upon the randPlayer by implementing the rules of our model in the updateStrats() function.

In our stratPlayer class, [pUpper,qUpper] and [pLower,qLower] represent our upper and lower bounds,R_i^U and R_i^L, on R_i respectively.

Putting all this together we now have the means to run a simulation of a game of monopoly with players performing strategically based on the rules we defined in our model.

3.5 Initial Tests

We shall now test out our code to ensure it runs as we would expect. Shown in figure 5 is a game of length 100 with arbitrarily chosen p and q values. In this particular example player 0 almost goes bankrupt but recovers to win the game at the end.

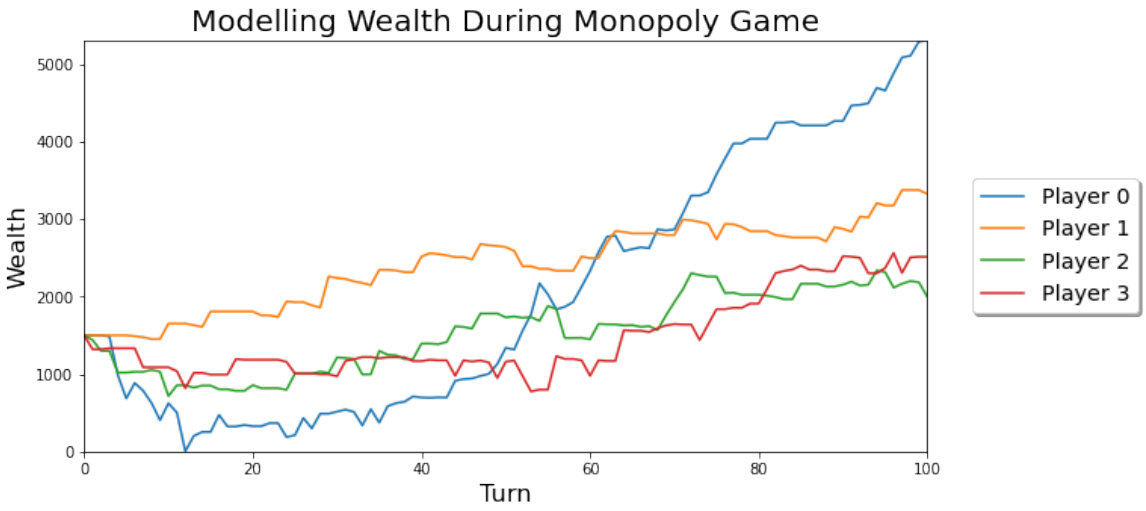


Figure 5: Graph of player wealth over a game of 100 turns with fixed p and q values.

Similarly, we can test our strategic players with variable p and q values, as shown in figure 6. We can see how the riskiness of the player evolves as their wealth and the total cost to them on the board changes with time. Now with everything working we can attempt to find an optimum strategy.

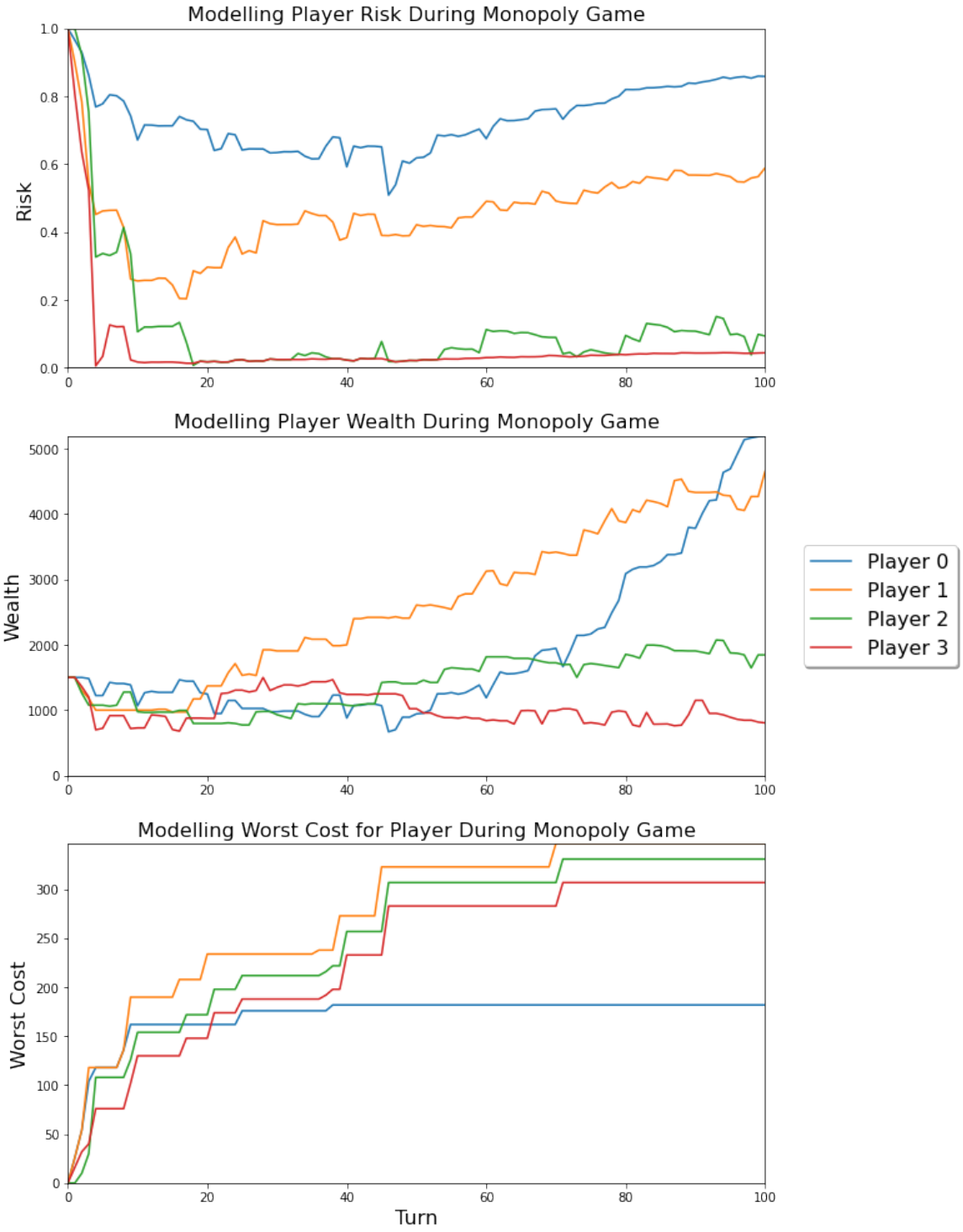


Figure 6: Graph of player riskiness, p, (top), wealth (middle), and worst cost (bottom) over a game of 100 turns for strategic players.

4 Finding an Optimum Strategy

4.1 Playing Against a Known Opponent

Now that we have our model functioning, we can attempt to determine the optimum riskiness given different types of opponents. To simplify our problem we will consider three opponent types, Risky, Anxious, and Neutral.

As the name suggest, a player who is Risky will be very likely to purchase sites and houses. By contrast, a player who is Anxious will be unlikely to invest in sites or houses. A player who is Neutral will lie somewhere in between the two.

We will assume that a player's riskiness is not constant. The player type will be defined by the bounds on their riskiness, R_i^L and R_i^U from our model. For the purpose of our test, we will use the bounds from table 1.

Table 1: The Bounds on p and q for Different Player Types

Player Type	Bounds on p	Bounds on q
Risky	[0.8,1]	[0.9,1]
Neutral	[0.3,0.5]	[0.4,0.6]
Anxious	[0,0.2]	[0,0.2]

4.1.1 Method of Approximation

Let's begin by constructing a method for the case where $R_i \in \mathbb{R}$, which we can then generalise to a method for $R_i \in \mathbb{R}^2$. The idea shall be to start with particular bounds for the riskiness and consider two other bounds above and below it. We play 1000 games with each of the bounds, then we take the bounds with the highest number of wins and repeat.

More specifically, we take a width as an input. Then we take the initial guess for the bounds to be $R_i^L = 0.5 - \frac{width}{2}$ and $R_i^U = 0.5 + \frac{width}{2}$. For stage i, the other two guesses for the bounds will be calculated by adding and subtracting $(0.5 - \frac{width}{2}) * 2^{-i}$ to our current guess. So in theory, if we iterated the process forever we would get the exact optimum range. For the sake of my sanity, we shall perform the procedure 5 times.

To generalise for $R_i = [p, q] \in \mathbb{R}^2$, we shall follow the same principle. We begin by picking an initial bound p, then we pick two other bounds for p above and below our current bound. Then we will fix the three bounds for p and perform the previous method on q. We pick the p that has the most wins when paired with its best q value.

Algorithm 1: Approximating Optimum Riskiness

Result: R^L, R^U
Input: Width;
Let $i = 0$, $L = 0.5 - \frac{width}{2}$, and $U = 0.5 + \frac{width}{2}$];
for $i = 0$ *to* 10 **do**
  $L_0 = L - (0.5 - \frac{width}{2}) * 2^i$, $U_0 = U - (0.5 - \frac{width}{2}) * 2^i$;
  $L_1 = L$, $U_1 = U$;
  $L_2 = L + (0.5 - \frac{width}{2}) * 2^i$, $U_2 = U + (0.5 - \frac{width}{2}) * 2^i$;
  for $j = 0$ *to* 2 **do**
    Play 1000 games of Monopoly with $R_i \in [L_j, U_j]$;
    Let W_j be the number of games won ;
  end
  Let $J = Argmax_{j \in \{0,1,2\}}\{W_j\}$;
  $L = L_J$, $U = U_J$
end
$R^L = L$, $R^U = U$

4.1.2 Assumptions

To use algorithm 1, there are certain assumptions that we must first make. We must assume that an optimum solution exists and, furthermore, that the mapping of our bounds on p and q to the total number of wins has only one maximum. The latter must be assumed as our algorithm only returns one maximum so we will only be aware of one optimum solution. This issue would arise if, for a fixed q, the total number of wins depended on p like graph C in figure 7.

Similarly, we assume that there are no local maxima, like in B from figure 7. We assume this as depending on where the global maximum is located our algorithm may return the local maximum, if it exists, instead of the global maximum. In other words, our algorithm would be returning the wrong solution.

This means that we are assuming that if we fix either p or q then the number of wins is dependent on the unfixed variable like in graph A from figure 7. We will also be assuming that enough games are played so that the proportion of games won is an accurate representation of how good the bounds are on p and q.

The bounds on q will only matter if our player owns all the sites in a colour set. We will assume that this is the case and q is implemented in at least some of the games run. However, for lower bounds on p this is unlikely to be the case. So the optimum bounds for q are more accurate when the lower bound for p is higher.

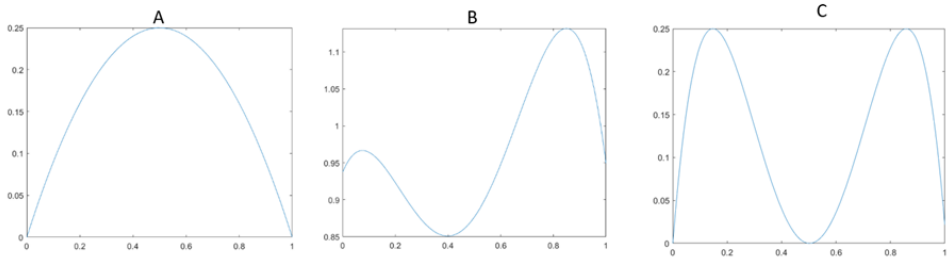


Figure 7: Examples of different types of graphs with different numbers of maxima.

4.1.3 Results

To simplify the problem, we will assume all players value money the same. So from this point on $W_i = 1$ for all players. Additionally, we will also take the difference between the upper and lower bounds of p and q to be 0.2 for both. We will consider games of length 30 through to games of length 120 incrementing the length by 5 turns per player each time.

We will start by looking into games with two opponents. We can see the results plotted in figure 8. Regardless of the opponent type the optimum lower bound on p is approximately zero until we reach games of length 80 turns each. Since it is reasonable to assume that most games will be shorter than 80 turns each, we have shown that without a doubt two player games of Monopoly are boring. This could perhaps be because an investment has a slower pay off as between two of your turns there is only one opponent who might land on your site.

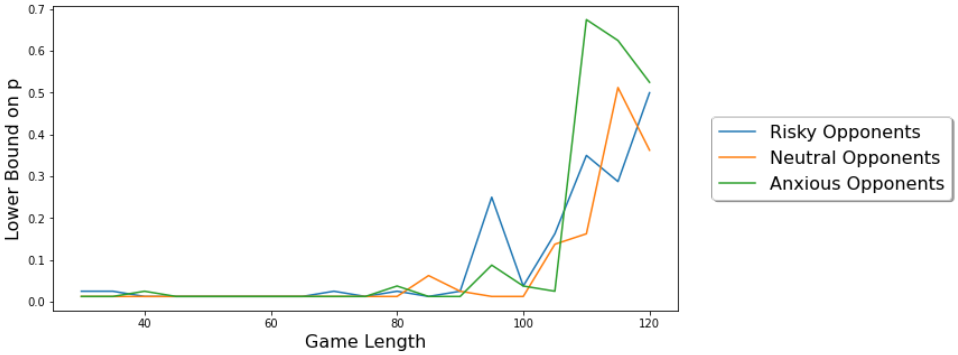


Figure 8: Plot of the optimum lower bound on the riskiness, p, for games of different lengths from 30 to 120 turns for one opponent.

Moving two-player games to the side, we are now going to look at games with three opponents. Our results are shown in figure 9, where instantly it looks much more promising. We see that there is a sudden switching point around games of length 50 turns each.

For games of length shorter than 50 turns each it is best to be risk averse when buying sites while for longer games it pays off to be risky when buying sites. However, are algorithm is unable to find an optimum lower bound on q. This suggests that perhaps there is more than one optimum lower bound on q or no optimum range on q exists.

We have learnt that given a fixed W_i and a fixed width on the bounds of p and q, the type of opponent does not appear to significantly impact the optimum lower bound on p. We shall further explore the optimum strategy by considering an alternative algorithm that allows for more than one optimum strategy. Using an evolution algorithm we will explore the possibility of the existence of two distinct optimum strategies.

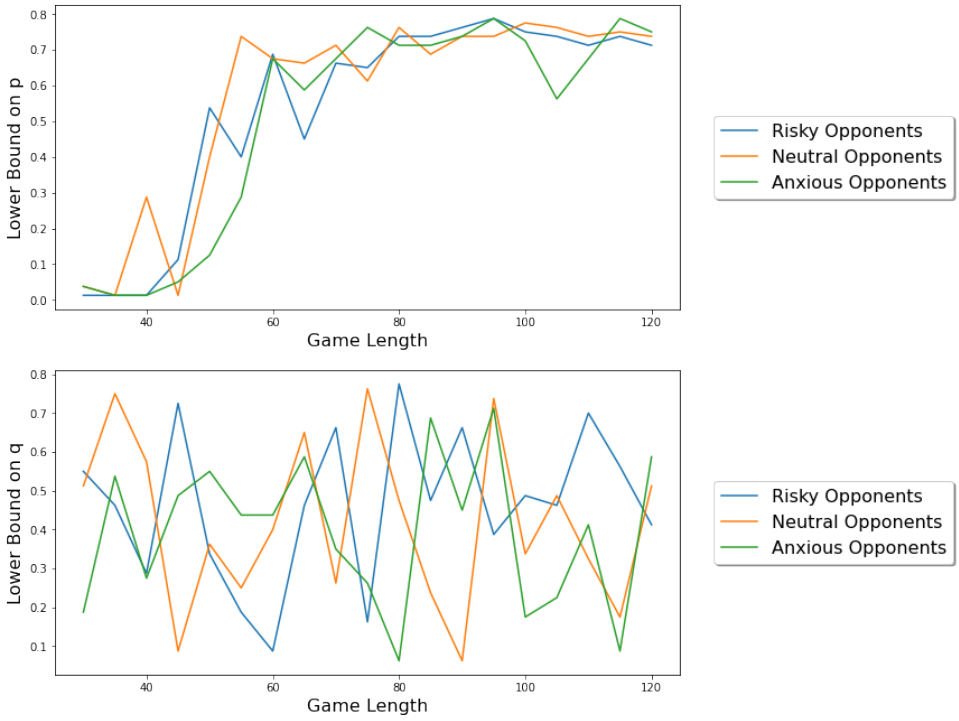


Figure 9: Plot of the optimum lower bound on the riskiness, p (top) and q (bottom), for games of different lengths from 30 to 120 turns with three opponents.

4.2 Optimisation Using an Evolution Algorithm

We will now use an evolution algorithm to find the optimum strategy if you don't know anything about your opponent. The premise of this algorithm is to have individuals with different parameters compete against each other and then the best individuals will survive to the next generation and reproduce to create similar children, as shown by Algorithm 2. The hope is that after a certain number of generations our players will converge to an optimum solution. The implementation of this algorithm was based on [4].

Algorithm 2: Evolution Algorithm

```
Result: W, [p_L, p_U], [q_L, q_U]
Input: Game Length, Number of Generations;
Create 100 random players;
for i = 1 to Number of Generations do
    Randomly group players into groups of 4;
    for Each Group do
        Play 1000 games of Monopoly;
    end
    Discard the worst 50 players;
    Randomly pair up players;
    for Each Pair do
        Create two new players with a mix of both parents
         parameters;
        Modify childrens' parameters with gaussian noise;
    end
end
```

We can see our results in figure 10, in which each point represents one of the final 40 players at the end of the 100 generations for our Winners. Looking at the top right-hand figure, we can see that a high riskiness is beneficial when buying sites, p, and our weighting on money doesn't seem to have a specific optimum value. Repeated runs of our evolution algorithm give the same conclusion that our weighting on money should be in the range 1-5. It would appear that the probability of buying a house, q, does not make any significant difference to our chances of winning. We shall call these players Winners.

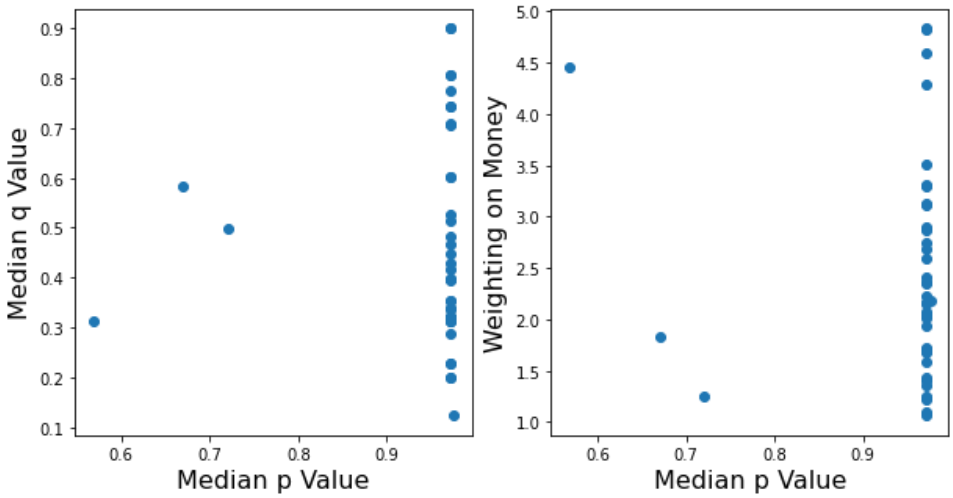


Figure 10: Plot of the optimum median q value (left) and optimum weight (right) against optimum median p value for games of length 80 with 3 opponents.

4.3 Minimising Bankruptcy

Our Winners have been optimised to prioritise coming first and not caring about any other outcomes. Perhaps we also wish to minimise bankruptcies while still winning some of our games. As great as winning is, if the chance of going bankrupt is too high, perhaps the risk isn't worth it. We will run another evolution algorithm with the alteration that the players wish to maximise the number of wins divided by one more than the number of bankruptcies. We shall call this new group Survivors.

We can see our final 40 survivors in figure 11. This time a low median probability of buying a site, p, of no more than 0.1 is beneficial to avoiding bankruptcy. To complement this, our players now have a high value on money with a weighting in the range 5.5-10 on their money. Once again the probability of buying a house, q, does not appear to matter for our Survivors.

We can see how our two groups of players compare in figure 12, where Randoms is a group of players created with random parameters as a reference. What we see is that the appropriately named survivors have a very low probability of going bankrupt but only win 4.25% of the games they play. Meanwhile, our Winners will win the game with probability 43.19% but this increase in winning comes with an increased probability that they will go bankrupt.

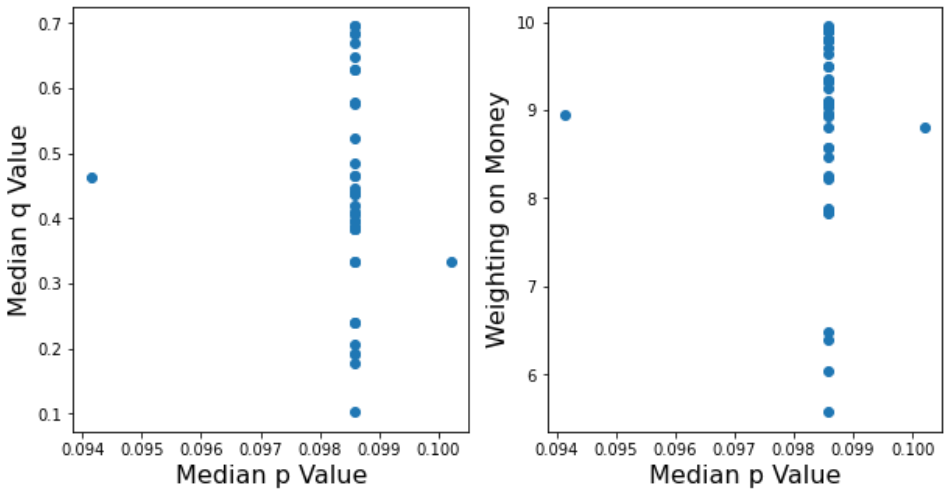


Figure 11: Plot of the optimum median q value (left) and optimum weight (right) against optimum median p value for games of length 80 with 3 opponents.

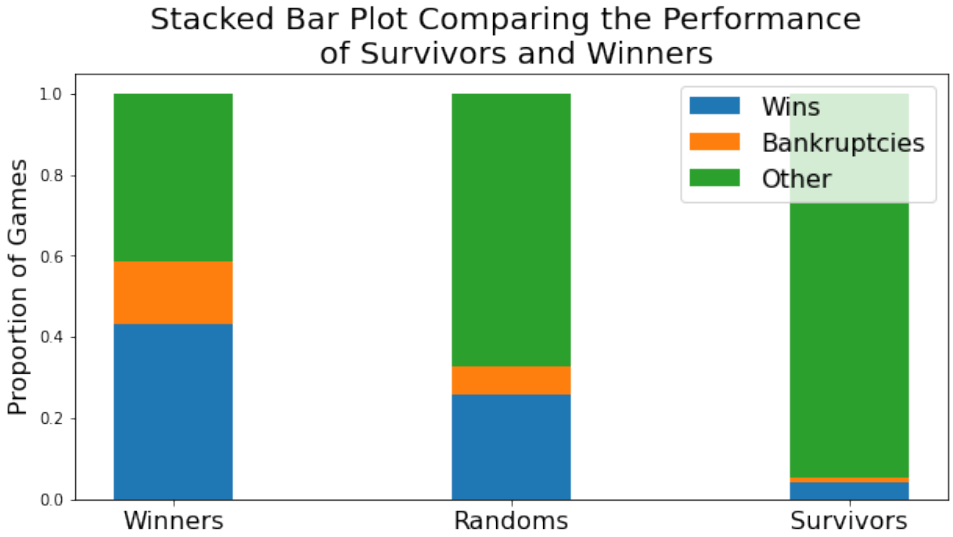


Figure 12: Stacked bar plot recording the proportion of games in which a player wins, goes bankrupt, or neither over 1000 games for 10 different players from each group of players

5 Closing Remarks

5.1 Project Extensions

There are potential extensions that could be made to this project that we shall address here. In subsection 2.3, we observed that there are certain sites that are landed on more frequently than others due to people going to jail. This could be used in the decision to purchase a square where p is dependent on how frequently the square you are currently on is landed on. Similarly, we could attempt to account for personal preference by creating character types who value certain sites over others. For example, a player type that likes to collect the train stations. A potential method for implementing this is discussed in Appendix B.

Additionally, we could have considered the ability for players to trade in the model, giving players the choices to swap sites in order to complete a set. However, this would have made the model overly complex; so for the purpose of the problem I was looking at I did not deem this necessary. A proposal for how trading could be implemented is described in Appendix C. A proposal for how to implement the ability for players to mortgage sites is discussed in Appendix D.

Finally, it could be interesting to extend this to other random risk-based board games. Using Algorithm 1 and 2 in a game like Frustration, we could consider balancing the risk of being caught by another player or not being able to free all your pieces.

5.2 Conclusions

What we have found is that the two objectives of surviving and winning oppose one another. To be in with the chance of winning you need to have a high risk strategy but with that you also burden yourself with an increased risk of bankruptcy. So in a game of monopoly you have to ask yourself how big a risk are you willing to take in order to win? At what point is the risk of bankruptcy too high to make the risk worthwhile? This is a decision only you can make. So take on board what we have found and choose how risky you wish to be in your next game of Monopoly wisely, as what we have shown is that the game of Monopoly does not rely on just chance alone.

A Variations for Reaction Rate

When deciding upon α_i there were several possible options. The first idea was

$$\alpha_i = Max(\frac{M_i - C_i}{M_i + W_i}, 0)$$

While this considered how much a player values money, the amount of money they had, and the rent they would have to pay on the bored it wasn't sensitive enough to changes in C_i. Large changes in C_i made virtually no difference once you made W_i large enough to represent how the player valued money.

Then the next idea came along.

$$\alpha_i = \frac{M_i}{M_i + C_i + W_i}$$

This had the advantage that α_i was always in the interval zero to one without the need for a max function. However it still wasn't responsive enough to changes in C_i.

Finally, the current version of α_i was chosen.

$$\alpha_i = Max(\frac{M_i - W_i C_i}{M_i + W_i C_i}, 0)$$

This had the advantage of considering both the worst cost and how much the player values money. It is sensitive to changes in both and as such it was chosen as the final equation.

B Implementing a Preference for Certain Sites

In a real game of Monopoly, it is reasonable to say that players are biased towards and against certain sites. Most people have little to no interest in Old Kent Road. We can represent this aversion or attraction to different sites with a vector of length 40.

We shall say that a player with an attraction to site i will have a value between 0 and 1 at position i of the preference vector while a player with an aversion will have a value greater than 1 at position i. The value of p when deciding to purchase site i will be raised to the power of the value at position i of the preference vector. A value of 1 translates to a player being neutral about that site.

Thus, a player will be more likely to buy a site they have an attraction to while they will be less likely to buy a site they have an aversion to. This would extend our module to allow for players to exhibit preferences towards certain sites.

C Implementing Trading into the Model

In order to implement a trading system successfully we must assume that no two players will accept a trade unless it is mutually beneficial to do so. As such we must have a method to compare a player's sacrifice with their gain. We can do this by saying every site must have a worth to a player that can be described numerically.

The potential for a trade proposal shall be implemented at the start of a player's turn. A player, who we shall call A, will look at the sites each of their opponents currently possesses. The opponent with the site that is most valuable to A, we shall call the opponent B, is the opponent who A shall attempt to initiate a trade deal with. We shall assume that A will only value sites that are part of a set A already has some of the sites of.

The trading process begins by A informing B that they are interested in site X. B shall respond by ranking A's sites in order of their value to B. B shall reply to A by requesting the first k sites of the ranking, with k the smallest value for which B deems the trade preferable; if no such k exists B shall decline the trade. A shall then make a counter offer to make the trade still beneficial to A. If B is happy with this final trade then the trade is successful. If B is not happy with the final trade then the trade is cancelled.

For this implementation we shall have to decide what factors impact the numerical worth of a trade. We will assume a site has no value to player A if player A currently owns no streets from the same set. We shall consider A's probability of buying a site, p, the default cost of rent on site i, P_i, and the proportion of the set site i is in that A already possesses, Q_i.

We begin by defining the value of a site i to A as $V_A(i) = P_i Q_i$. This conveniently gives a site in a set A currently has no streets from a value of 0. If we denote $I, J \subset [39] \cup \{0\}$ as the streets in the trade that A gains and sacrifices respectively then we can define the value of the trade as shown below, where δ is some small positive constant of size at least one to prevent a zero on the denominator. σ_1 and σ_2 are scaling factors that could be different for each player and represent how they value their gain relative to their sacrifice.

$$T_A(I, J) = \sigma_1 \frac{\sum_{i \in I} V_A(i)}{\delta - p} - \sigma_2 \frac{\sum_{j \in J} V_B(j)}{\delta + p}$$

By defining the trade value for B in a similar way, we can say that both players will accept the trade if and only if $T_A(I, J)$ and $T_B(I, J)$ are both strictly positive. The use of p in the equation for $T_A(I, J)$ means that a less risky player is less likely to accept a trade since they care more about the gain their opponent makes.

D Implementing Mortgaging of Sites into the Model

We will start by assuming that no player wants to mortgage their sites unless they have no choice but to do so. As such, mortgaging will only be used when a player cannot afford to pay the rent or tax they are being asked for.

As in Appendix C, we need to implement a method to assign a numerical cost to mortgaging a site. To do so we shall consider the cost of rent on site i, P_i, the mortgage value of site i, V_i, the expected frequency that a player lands on site i, F_i, and the amount owed, D.

We want to minimise our expected loss of rent while still mortgaging enough sites to pay our debts. We can define the numerical cost to mortgaging a site as $S_i = P_i F_i$. Since the frequency a player lands on a site is proportional to the probability a player lands on a site, we are lead to an optimisation problem. We wish to minimise the expected loss in rent, so if we define J as the properties our player owns we get the following optimisation problem.

$$\text{Minimise}_I \sum_{i \in I} S_i \quad \text{subject to } \sum_{i \in I} V_i \geq D,\ I \subset J$$

By implementing an algorithm to solve the optimisation problem above, we would now have a way to mortgage sites. Since a mortgaged site is of no value to a player, we shall assume that a player will unmortgage a site as soon as they can afford to do so.

References

[1] Eric J Friedman, Shane G Henderson, Thomas Byuen, and Germán Gutiérrez Gallardo. Estimating the probability that the game of monopoly never ends. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 380–391. IEEE, 2009.

[2] Cinzia Giorgetta, Alessandro Grecucci, Sophia Zuanon, Laura Perini, Matteo Balestrieri, Nicolao Bonini, Alan G Sanfey, and Paolo Brambilla. Reduced risk-taking behavior as a trait feature of anxiety. *Emotion*, 12 (6):1373, 2012.

[3] Jennifer S Lerner and Dacher Keltner. Fear, anger, and risk. *Journal of personality and social psychology*, 81(1):146, 2001.

[4] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2): 398–417, 2008.