Parrondo's Games

Konstantine Mushegian

December 14, 2016

Abstract

The Parrondo's Paradox is a seemingly absurd phenomenon where two losing strategies can be combined to produce a winning outcome. In this paper we consider two games, A and B, both of which are mediated by a coin toss. Both games are rigged against the player by a biasing parameter , and as a result both games have negative expected returns on the starting capital m. However, when games A and B are played in any kind of alternating sequence they, paradoxically, produce positive expected returns on the starting capital. Although this peculiar result is quite interesting in itself, we will focus on the underlying linear algebra concepts that result in this paradox, and discuss this phenomenon in the context of Markov chains. We will also discuss the possible applications of this phenomenon in real life.

1 Introduction

People have been studying probability for a long time; interest in this area has often been fueled by gambling, probabilistic games which people have tried to outsmart seeking financial gains. The paradox discussed in this paper that two probabilistic games A and B, both with negative expectations, can produce a positive expectation when played in an alternating sequence was introduced by Parrondo to illustrate Brownian ratchets[1]. While Brownian ratchets constitute an interesting area of research, they're outside the scope of this paper. Instead, we will focus on a discretized version of the flashing ratchet, which can be represented as a pair of gambling games that form the basis of the Parrondo's paradox.

We will first introduce the aforementioned pair of gambling games and analyze them using basic probability and linear algebra. We will then simulate the games to confirm our assumptions about the two games. Next we will introduce the third game, which just plays the first two games in some kind of alternating sequence. We will repeat the same analysis for the third game in order to figure out why the Parrondo's paradox occurs and discuss the mathematical explanations for the paradox in the context of the materials covered in the Intermediate Linear Algebra course, taught by Professor Thomas Pietraho during the Fall 2016 semester.

We are going to refer to the first, second, and third game as Game A, Game B, and Game C respectively.

2 Games

In this section we define Games A, B, and C and analyze them. Before we dive into analyzing the games, I would like to note that all statements regarding probabilities and expected values are made from the player's perspective, i.e. the person who is playing against us. For example, if the probability of winning Game X is 0.67, it means that the 'player', i.e. our opponent is winning with that probability. Similarly, if the expected value of Game X is negative it means that the 'player' is losing money, i.e. we are wining and vice versa. Now that we have cleared the possible confusions introduced by the perspective of computations we can safely proceed with our analysis.

2.1 Game A

Game A is a fairly simple game that is mediated by a biased coin toss. The bias is defined by the a biasing parameter ϵ, which is just an arbitrary small number. While the exact outcome of the game depends on the value of ϵ, it is not required to know the exact value of it find the expected value of the game using basic probability.

$$\text{Game A: } \begin{cases} P[Win] = \frac{1}{2} + \epsilon \\ P[Loss] = \frac{1}{2} - \epsilon \end{cases}$$

Winner of Game A gets \$1 from the loser.

We decided it would be worthwhile to compute the expected value of Game A to develop a general understanding of what was going to happen in the long run. The expected value can be computed using the following equation

$$EV = \sum_{i=1}^{n} x_i p_i$$

where x_i are the possible values or outcomes of the game and p_i are their associated probabilities. The possible values of our game are 1 or -1, i.e. the player either wins, earning himself \$1, or loses and gives up his \$1 to the opponent.

$$EV[\text{Game A}] = 1 \cdot \left(\frac{1}{2} + \epsilon\right) + (-1) \cdot \left(\frac{1}{2} - \epsilon\right) = 2\epsilon$$

We have been using the biasing factor $\epsilon = 0.005$ in our simulations. Plugging in this value into the result above we get the expected value of Game A equal to ≈ 0.01, i.e. playing this game would result in an approximately 1 cent profit per game for the player, draining our starting capital over time.

We also simulated Game A in MATLAB by running 10,000 iterations of 100 flips and computing the average expectation with the biasing factor $\epsilon = 0.005$. The result of this simulation can be seen in Figure 1. The graph produced by the simulation once again verifies that Game A has negative expected value for us. Thus, if someone proposes to play Game A with you, you should reject this lucrative opportunity to get rich.

Additionally, we thought it would be worthwhile to explore the effect of ϵ on the average returns of Game A, since probabilities of winning and losing are functions of ϵ. In order to figure this out, we ran 10,000 iterations of Game A, each with 100 flips over the values of ϵ ranging from 0.001 to 0.01. As we can see from Figure 2 as ϵ becomes smaller, so do the average returns from playing Game A.

2.2 Game B

Game B is a little more complicated and its formulation is as follows: Let M to be the amount of money in your pocket and assume this is an integer. We will toss two coins: If M is divisible by 3, we will flip Coin 1, otherwise we will flip a coin. Coin 1: You win with probability $\frac{9}{10} + \epsilon$. Coin 2: You win with probability $\frac{1}{4} + \epsilon$.

$$\text{Game B: } \begin{cases} M\%3 = 0 & \begin{cases} P[Win] = \frac{9}{10} + \epsilon \\ P[Loss] = \frac{1}{10} - \epsilon] \end{cases} \\ M\%3 \neq 0 & \begin{cases} P[Win] = \frac{1}{4} + \epsilon \\ P[Loss] = \frac{3}{4} - \epsilon] \end{cases} \end{cases}$$

Again, at each turn the loser pays the winner \$1.

In order to verify our assumption that Game B has a negative expectation, we also computed its expected value.

$$EV[\text{Game B}] = 1 \cdot \left(\frac{1}{3} \left(\frac{9}{10} + \epsilon \right) + \frac{2}{3} \left(\frac{1}{4} + \epsilon \right) \right) + (-1) \cdot \left(\frac{1}{3} \left(\frac{1}{10} - \epsilon \right) + \frac{2}{3} \left(\frac{3}{4} - \epsilon \right) \right)$$

$$= \left(\frac{3}{10} + \frac{1}{3}\epsilon + \frac{1}{6} + \frac{2}{3}\epsilon \right) - \left(\frac{1}{30} - \frac{1}{3}\epsilon + \frac{1}{2} - \frac{2}{3}\epsilon \right) = 2\epsilon - \frac{1}{15}$$

We have been using the biasing factor $\epsilon = 0.005$ in our simulations. Plugging in this value into the result above we get the expected value of Game B equal to ≈ -0.056, i.e. playing this game would result in an approximately 5.6 cent loss every time we played against the player, draining our starting capital over time.

We also simulated Game B in MATLAB by running 10,000 iterations of 100 flips and computing the average expected returns at each iteration, using the

same biasing factor $\epsilon = 0.005$. The result of this simulation can be see in Figure 3. The graph produced by the simulation once again verifies that Game B has negative expected returns. Thus, similarly to Game A, we should reject the proposal to play Game B if someone kindly invites us to do so.

Similarly to Game A, the probabilities of winning and losing Game B are also functions of ϵ and thus we wanted to see the effect of ϵ on the average returns of Game B. Similar to the experiment for Game A, we ran 10,000 iterations of Game A, each with 100 flips for each of the values of ϵ ranging from 0.001 to 0.01. As we can see from Figure 4 the smaller the value of ϵ the smaller our average returns are from playing Game B.

Before we can proceed with analysis of Game B, we need to introduce several concepts that make this analysis possible.

2.2.1 Discrete Dynamical Systems

Dynamical systems are used to describe how a given system transitions from one state to another over time. The state transitions can occur smoothly over time or in discrete time steps. We will only touch upon discrete time or simply discrete dynamical systems since our games do not occur continuously over time, but rather in discrete time steps, where a time step is defined as a single iteration of the game. Discrete dynamical systems are often used to model specie populations, migrations, decease spreading, and any other system in which state changes occur over discrete time steps.

Discrete dynamical systems can be represented as a matrix that encompasses the state transitions and the associated probabilities. The state of the system at time step t_{i+1} can be computed by applying the state transition matrix to the state of the system at time t_i. State of the system at a given time is represented by a one-dimensional vector that quantifies said state.

Once we have obtained a good model, i.e. a good state transition matrix, for a given discrete dynamical system, we can apply it to some starting state infinitely and in this way predict the future. However, instead of applying the state transition matrix over and over, we would like to know whether a system ever achieves a steady state, i.e. a state where there the system remains unchanged as time keeps going, i.e. as we keep applying the state transition matrix. A steady state vector is a probability vector v, such that $Mv = v$ where M is the state transition matrix. We should know from Linear Algebra I that such v is an eigenvector of M associated with the eigenvalue 1.

we will talk about steady states more formally a little further on, but a takeaway for now is that steady states can be used to analyze the behavior of a given discrete dynamical system in the long run.

2.2.2 Markov Chains

A Markov chain is a discrete dynamical system where the state transition matrix A is a Markov matrix. A Markov matrix is a matrix where each entry is a non-negative real number that represents a probability, i.e. ranging between 0 and 1, both included. Markov matrices are also known as stochastic matrices.

Stochastic matrices come in two flavors, left stochastic matrices and right Stochastic matrices. A right stochastic matrix is a real square matrix with each row summing up to 1, while a left stochastic matrix is a real square matrix with each column summing up to 1. We are going to use the latter definition in this paper because of the way we construct the state transition matrices for our discrete dynamical systems.

2.2.3 Perron-Frobenius Theorem

Below we are going to state the Perron-Frobenius theorem; we will use it later in our analysis of Game B and Game C.

Theorem 1 (Perron-Fronebius theorem) *Suppose A is a Markov matrix with positive entries. Then,*

1. *$\lambda = 1$ is an eigenvalue of A.*
2. *if $\mu \neq \lambda$, then $|\mu| < 1$.*
3. *Eigenvector corresponding to λ will have all positive entries.*

We should also make note of the following fact:

> Suppose A is a Markov matrix and is diagonalizable. Then the corresponding discrete dynamical system always approaches a steady state, i.e.
>
> $\lim_{n\to\infty} v_n = v_\infty$ where v_i represents the state of the system at time i

The Perron-Frobenius theorem leads to one very nice upshot:

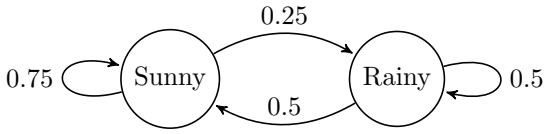
> if A is a D.D.S. given by $\vec{v}_n = A\vec{v}_{n-1}$, then $\exists$ vector $\vec{v}_\infty : \lim \vec{v} = \vec{v}_\infty$
> Note: $A\vec{v}_\infty = \vec{v}_\infty$ i.e. $\vec{v}_\infty$ will be the eigenvector corresponding to eigenvalue 1. Thus, $\vec{v}_\infty$ will tell us what the state of our system will be in the long run, i.e. it will give us the probabilities of being in each state as time approaches ∞.

We are going to use the results of this section extensively in further analysis of our games.

2.2.4 Discrete Dynamical System Example

In order to fully develop the understanding of discrete dynamical systems and steady states we are going to go over a short example. This example was adapted from the one provided in [2].

Assume we're given some probabilities of weather conditions, modeled as either sunny or rainy, given the weather on the preceding day. The system is represented by the following diagram



From this diagram we can build the state transition matrix, M

$$M = \begin{bmatrix} 0.75 & 0.5 \\ 0.25 & 0.5 \end{bmatrix}$$

Matrix M represents a weather forecasting model where a sunny day is 75% likely to be followed by another sunny day and a rainy day is 50% likely to be followed by another rainy day.

Note that all entries of matrix M are between 0 and 1, and all column sums are equal to 1. Thus, matrix M is a left stochastic, i.e. a Markov matrix.

Now that we've got a state transition matrix M that models our system, we can go ahead and start predicting the weather. Let's imagine that the weather on day 1 is sunny, and we are fully (100%) confident in our knowledge of the weather on day 1. Thus, the state of our system on day 1 can be represented as

$$d_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

We can then forecast the weather on day 2 by applying the matrix M to the vector d_1:

$$d_2 = Md_1 = M = \begin{bmatrix} 0.75 & 0.5 \\ 0.25 & 0.5 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}$$

Thus, there is a 75% chance that the weather on day 2 will also be sunny and a 25% chance that the weather on day 2 will be rainy.

Finally we would like to know the weather on all days regardless of the weather on the first day, i.e. we would like to know the steady state of this matrix. From above, we know that a discrete dynamical system achieves a steady state if its state transition matrix is a Markov matrix and is diagonalizable. We have

already established that M is a Markov matrix. We also know from Linear Algebra I that an nxn matrix is diagonalizable if and only if it has n distinct eigenvalues. Let's compute the eigenvalues of matrix M.

Note: We performed all the computations using Mathematica, however it would be beneficial for you to verify the results by performing the calculations by hand.

Eigenvalues of matrix M are $\lambda_1 = 1$ and $\lambda_2 = 0.25$. Thus, M is diagonalizable. Now we can find the steady state vector of our system by looking at the eigenvector corresponding to λ_1. The eigenvectors of M are

$$v_1 = \begin{pmatrix} 0.894427 \\ 0.447214 \end{pmatrix} \quad v_2 = \begin{pmatrix} -0.707107 \\ -0.707107 \end{pmatrix}$$

Thus, v_1 is the steady vector of our system. Last thing we need to do is normalize v_1 in order to obtain the probabilities for each state. In order to do this we can divide each entry of v_1 by the sum of all entries and get

$$v_{\text{1-normalized}} \approx \begin{pmatrix} 0.66 \\ 0.33 \end{pmatrix}$$

Thus, in the long run, about 66% of the days are sunny.

Now that we have the three definitions above, we can proceed with our analysis of Game B. It turns out that Game B can be modeled by a Markov chain. Since Game B depends on the amount of money in our pocket, M, the states of this Markov chain are going to be $M\%3 = 0$, $M\%3 = 1$, and $M\%3 = 2$. We know what state winning or losing at a given state is going to transition into; for example, if we win at state $M\%3 = 0$, M now equals $M + 1$ and thus the next state will be $M\%3 = 1$. The full Markov transition diagram can be seen in Figure 5.

From the state transition diagram we can construct a Markov matrix that represents Game B. The Markov matrix for Game B is as follows

$$M_B = \begin{bmatrix} 0 & \frac{3}{4} - \epsilon & \frac{1}{4} + \epsilon \\ \frac{9}{10} + \epsilon & 0 & \frac{3}{4} - \epsilon \\ \frac{1}{10} - \epsilon & \frac{1}{4} + \epsilon & 0 \end{bmatrix}$$

At this point, let's try diagonalizing the matrix M_B in order to see whether the discrete dynamical system it represents ever achieves a steady state. We know from 2.2.3 that a given system achieves a steady state if one of the eigenvalues of the Markov matrix associated with it is 1.

During our simulations, we have been using 0.005 as a value of ϵ and thus we used this value again when computing the eigenvalues for the matrix M_B. We obtained the eigenvalues using Mathematica and were quite delighted with the results. The eigenvalues of M_B are $\lambda_1 = 1$, $\lambda_2 = -0.869358$, and $\lambda_3 = -0.130642$.

Since M_B is a 3x3 matrix and has 3 distinct eigenvalues it is diagonalizable. Thus by 2.2.3, the discrete dynamical system that is Game B achieves a steady state. It is worthwhile to note that if we left ϵ as a variable, we still get one of the eigenvalues equaling 1, however everything else is quite messy.

The next step in our analysis is to find the said steady state of Game B. In order to do this, we will first compute the eigenvectors of M_B

$$v_1 = \begin{pmatrix} -0.57735 \\ -0.634656 \\ -0.297218 \end{pmatrix} v_2 = \begin{pmatrix} -0.57735 \\ 0.663044 \\ -0.0683448 \end{pmatrix} v_3 = \begin{pmatrix} -0.57735 \\ -0.396969 \\ 0.95236 \end{pmatrix}$$

Since v_1 is the eigenvector corresponding to eigenvalue $\lambda_1 = 1$ it is the steady vector of our system Lastly, we need to normalize v_1 in order to obtain the probabilities for each state. We normalize v_1 the same way as in 2.2.4, dividing each entry by the sum of all entries and we get

$$v_{\text{1-normalized}} = \begin{pmatrix} 0.382548 \\ 0.420518 \\ 0.196934 \end{pmatrix}$$

This vector gives us the probabilities of being in each state $M\%3 = 0$, $M\%3 = 1$, and $M\%3 = 2$, however as we can see these probabilities are different from the ones we used to compute the initial expected value. This is because we made an assumption that the state $M\%3 = 0$ will occur with probability $\frac{1}{3}$, i.e. we would flip the first coin once every three times. However, as shown by the steady state vector, this assumption is wrong. According to the steady state vector, the state $M\%3 = 0$ occurs 38.25% of the time. Having obtained the real probability distributions for the states of Game B we can now recompute the expected value and see what actually happens when we play the game.

$$\begin{aligned} EV[\text{Game B}]_{modified} &= 0.3825 \cdot \left((1) \left(\frac{9}{10} + \epsilon \right) + (-1) \left(\frac{1}{10} - \epsilon \right) \right) \\ &+ 0.6175 \cdot \left((1) \left(\frac{1}{4} + \epsilon \right) + (-1) \left(\frac{3}{4} - \epsilon \right) \right) \\ &= 0.3825 \cdot \left(\frac{8}{10} + 2\epsilon \right) + 0.6175 \left(-\frac{1}{2} + 2\epsilon \right) \end{aligned}$$

At this point let's plug in the biasing factor $\epsilon = 0.005$ we have been using all along to figure the true expected value of Game B. Therefore

$$EV[\text{Game B}]_{modified} = 0.009$$

Thus, playing Game B will result in approximately in a 0.9 cent profit for the player. Nothing has changed - if someone's offering to play this game against you for fun, you should start questioning their motives, since they will drain your capital over time.

2.3 Game C

Game C is where things start to get interesting. The formulation of Game C is quite simple:

> Randomly alternate playing Game A and Game B.

However, despite the simplicity of its formulation, the state transition diagram for Game C is far more complicated than it was for Game B. This complication is introduced by the fact that we're randomly playing Game A or Game B, but Game B has three possible states itself. We thought it would be best to first represent Game C as a decision tree in order to fully understand the process. The decision tree can be seen in Figure 6. Now that we know how Game C can be lost or won, we can proceed to analyzing it.

Since Game C is far more complicated than Game B, we decided to first compute it's steady states in order to obtain the probabilities for each of the three states, and then compute its expected value.

We used the (non-pictured) Markov chain state transition diagram for Game C to construct the Markov matrix that represents our Markov chain. This Markov matrix is as follows

$$M_C = \begin{bmatrix} 0 & \frac{5}{8} - \epsilon & \frac{3}{8} + \epsilon \\ \frac{7}{10} + \epsilon & 0 & \frac{5}{8} - \epsilon \\ \frac{3}{10} - \epsilon & \frac{3}{8} + \epsilon & 0 \end{bmatrix}$$

We can now try diagonalizing the matrix M_C in order to see whether the discrete dynamical system it represents ever achieves a steady state. We know from 2.2.3 that a given system achieves a steady state if one of the eigenvalues of the Markov matrix associated with it is 1.

During our simulation, we have been using 0.005 as a biasing factor ϵ and thus we used this value again when computing the eigenvalues of matrix M_C. We obtained the eigenvalues using Mathematica and were once again quite delighted with the results. The eigenvalues of M_C are $\lambda_1 = 1$, $\lambda_2 = -0.6838$, and $\lambda_3 = -0.3162$. Since M_C is a 3x3 matrix that has 3 distinct eigenvalues it is diagonalizable. Thus by 2.2.3, the discrete dynamical system that is Game C achieves a steady state. Once again, it is worthwhile to note that even when we performed the above computations symbolically, one of the eigenvalues was still equal to 1.

The next step in our analysis is to find the said steady state of Game C. In order to do this, we will first compute the eigenvectors of M_C.

$$v_1 = \begin{pmatrix} -0.5897 \\ -0.6822 \\ -0.4323 \end{pmatrix} v_2 = \begin{pmatrix} -0.6325 \\ 0.7634 \\ -0.1310 \end{pmatrix} v_3 = \begin{pmatrix} -0.6325 \\ -0.1310 \\ 0.7634 \end{pmatrix}$$

Since v_1 is the eigenvector corresponding to eigenvalue $\lambda_1 = 1$ it is the steady state vector of our system. Lastly, we need to normalize v_1 in order to obtain the probabilities of being in each state. We normalize v_1 the same way as in 2.2.4, dividing each entry by the sum of all entries and we get

$$v_{\text{1-normalized}} = \begin{pmatrix} 0.3460 \\ 0.4003 \\ 0.2537 \end{pmatrix}$$

This vector gives us the probabilities of being in each state $M\%3 = 0$, $M\%3 = 1$, and $M\%3 = 2$.

Computing the expected value of Game C can be broken down into two parts, when $M\%3 = 0$ and $M\%3 \neq 0$. When $M\%3 = 0$,

$$P(Winning) = \frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2}\left(\frac{9}{10} + \epsilon\right) = q_1$$

When $M\% \neq 3$

$$P(Winning) = \frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2}\left(\frac{1}{4} + \epsilon\right) = q_2$$

And thus we can compute the expected value of Game C using the following equation

$$EV[\text{Game C}] = 0.346(q_1(1) + (1 - q_1)(-1)) + 0.654(q_2(1) + (1 - q_1)(-1))$$

We are once again going to use the biasing factor $\epsilon = 0.005$ to make it consistent with our simulations. Plugging in this value into the result above we get the expected value of Game C equal to -0.0151,i.e. playing this game would result in an approximately 1.5 cent loss for the player, thus building upon *our* capital over time!

This is the Parrondo's paradox! It is so counter-intuitive, and yet absolutely real. The combination of two losing strategies, results in a winning strategy. This is explained by the probabilities of being in each state $M\%3 = 0$, $M\%3 = 1$, and $M\%3 = 2$ which are provided to us by the steady vector. It turns out that the probabilities of being in each state are different from what we initially thought - this in turn causes our opponent to lose more. The probability of being in $M\%3 = 0$ state is 0.346, i.e. our opponent is less likely to be in a state where she is more likely to win. The paradox arises because the initial assumption was that each one of the three states was equally likely to happen.

This is a pretty amazing result! In order to confirm our results, we simulated Game C over 10,000 iterations, each with 100 flips. The results of this simulation can be seen in Figure 7. Additionally, you can see the simulations of all the games plotted on the same graph in Figure 9.

Since Game C is just a combination of Game A and Game B, we thought it would be interesting to see how values of ϵ affect the average returns of Game C and whether there exists a value of ϵ at which Game C stops having positive returns. In order to do this, we ran 10,000 iterations of Game C, each with 100 flips for each of the values of ϵ ranging from 0.001 to 0.012. It turns out that Game C stops being a winning game when $\epsilon = 0.012$. This result is obtained from the simulation, which is presented in Figure 8. In some cases we found that higher values of ϵ would make Game C a losing game, but we attributed this to the randomness that is inherent to Game C. $\epsilon = 0.012$ resulted in Game C being a losing game during every single simulation.

3 Possible Applications

The Parrondo's paradox has found application in engineering, physics, biology, and finance. People have also obviously tried using the paradox to trick the casinos, however I don't think that the Parrondo's paradox could be very beneficial for gambling since one of the games has to be capital-dependent by definition. On a second thought there might be some gambling games that the Parrondo's paradox could be applied to, but I am no expert, for better or worse.

I was hoping to use this section to demonstrate the paradox stripped out of all the mathematics. However, lacking enough knowledge in all of the above-mentioned areas to figure out how the paradox is applied, I decided to construct my own example.

Imagine that as a student at Bowdoin College you have two activities available to you:

1. Do homework all day.
2. Throw a frisbee on the quad all day.

If you choose to do one or the other for the whole day, every day, you're running the risk of becoming either the person who is always in the library and doesn't talk to anybody or the person who is always on the quad and has never been seen doing homework. Neither of these is a great option, since ideally you'd want to graduate and also have some fun while at it. However, if you combined these two activities, say you spent half a day doing homework and half a day throwing a frisbee on the quad or spent one full day doing work and another full day throwing frisbee on the quad you'd be able to balance academics and leisure. *Gracias, Parrondo!* You've just combined two strategies with negative expectations into a single strategy with a positive expectation.

4 Conclusions

At the beginning of this paper we introduced two gambling games, which we refer to as Game and Game B. We carefully analyzed both games using probability and linear algebra techniques to figure out whether these games would yield

positive or negative returns over time. As it turned out both of these games have negative expected returns, and would drain our starting capital over time. Then we introduced a third game, which is comprised of playing Game A and Game B randomly. We also analyzed Game C, and it turns out that the expected returns of Game C are positive, guaranteed to build up our starting capital over time. This is the Parrondo's paradox - two losing strategies could be combined to produce a winning strategy. The Parrondo's paradox has found applications in physics, engineering, and finance.

5 Figures

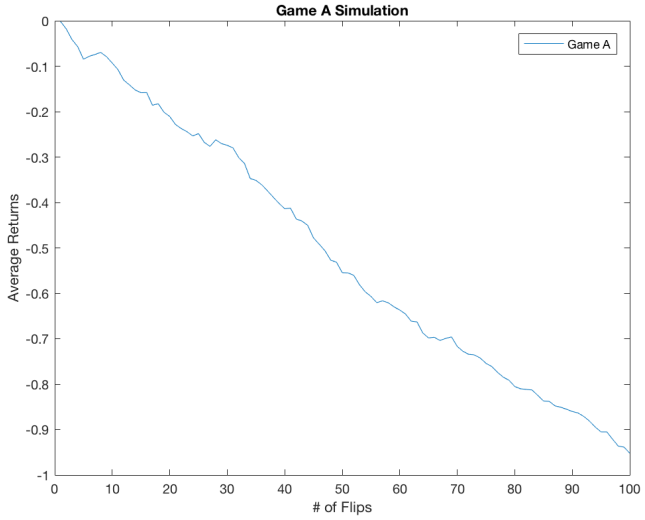


Figure 1: Average returns over 10,000 iterations of Game A, each with 100 flips.

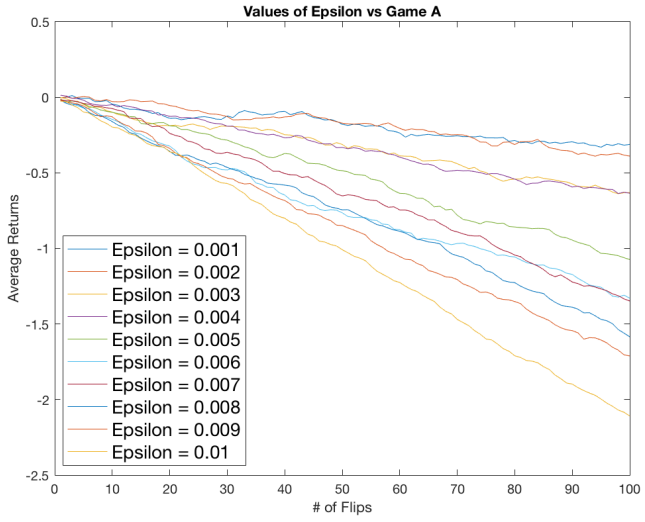


Figure 2: Average returns over 10,000 iterations of Game A, each with 100 flips over various values of ϵ.

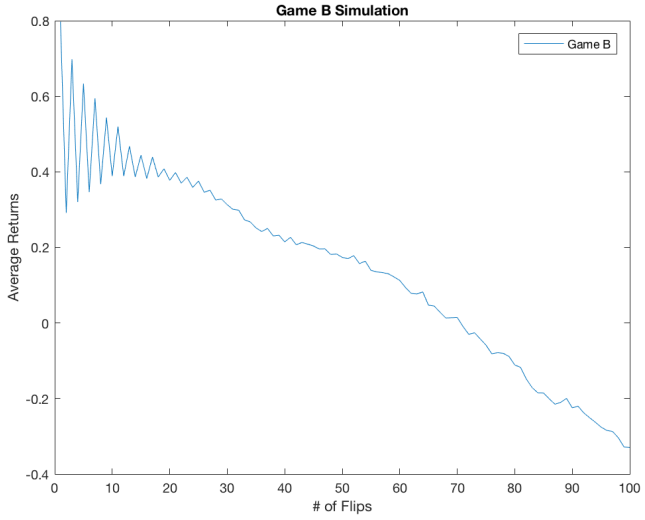


Figure 3: Average returns over 10,000 iterations of Game B, each with 100 flips.

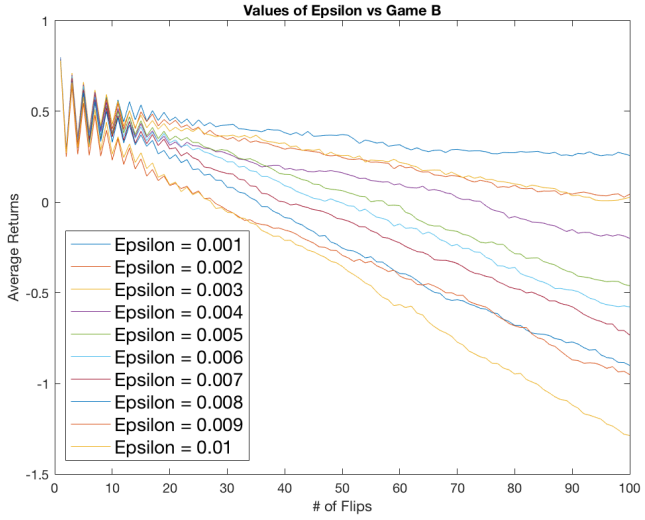


Figure 4: Average returns over 10,000 iterations of Game B, each with 100 flips over various values of ϵ.

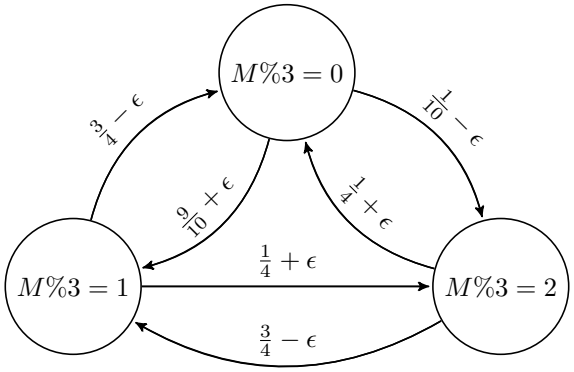


Figure 5: Game B State Transition Diagram.

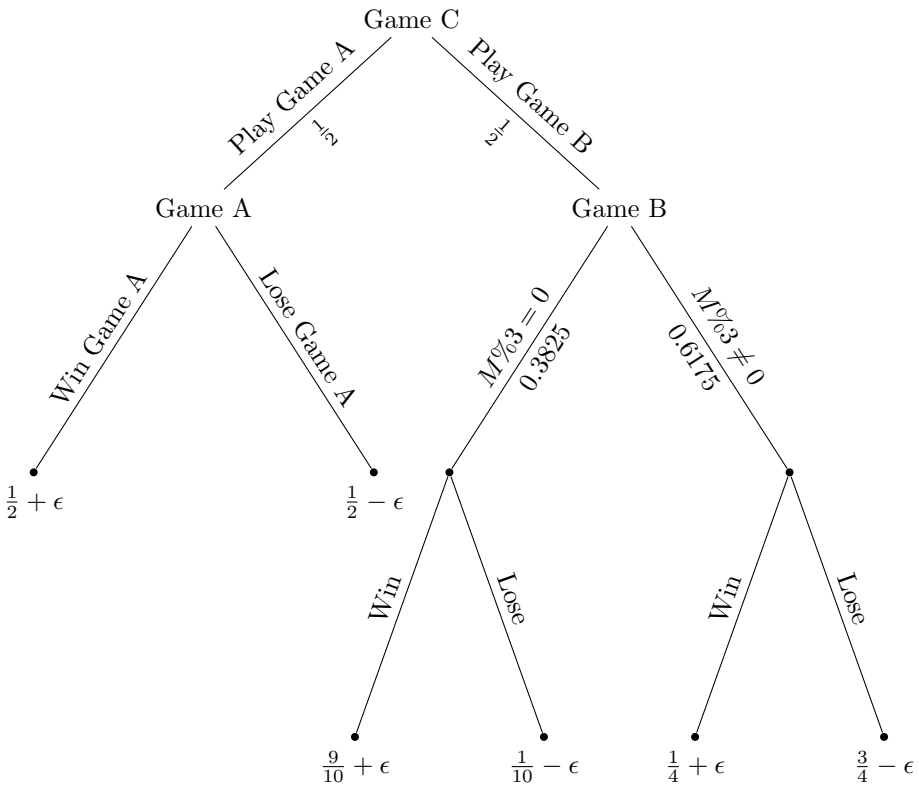


Figure 6: Game C Decision Tree

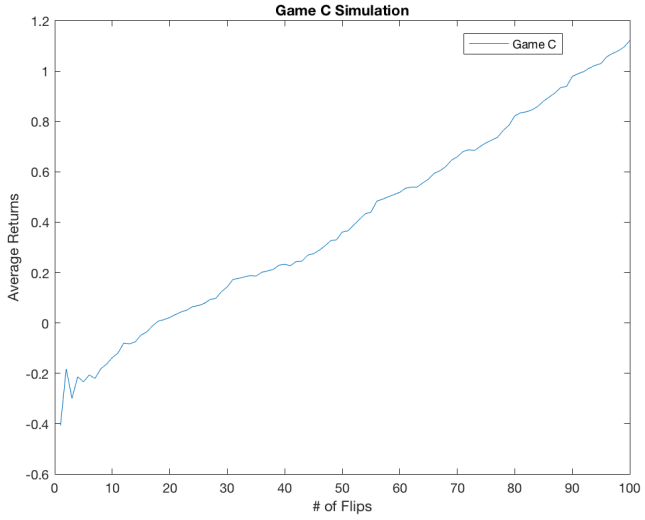


Figure 7: Average returns over 10,000 iterations of Game C, each with 100 flips.

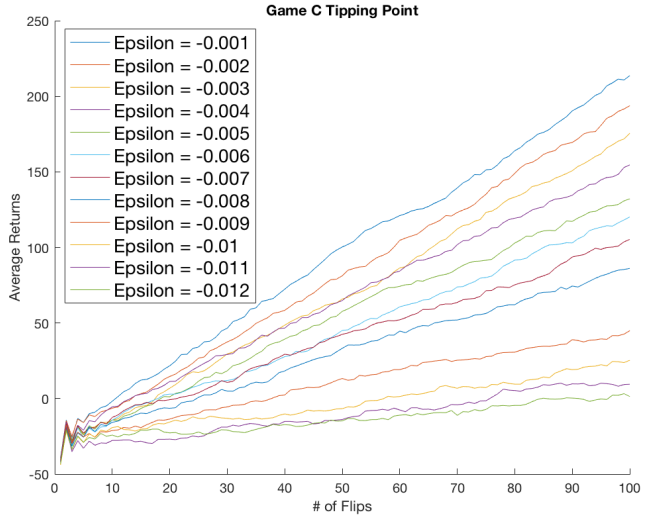


Figure 8: Game C stops being a 'winning' game after ϵ reaches a certain threshold.

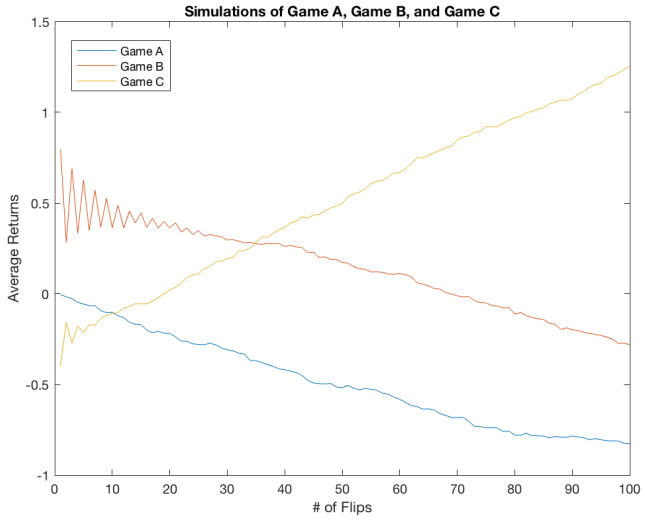


Figure 9: Average returns over 10,000 iterations of Game A, B, and C, each with 100 flips.

References

[1] J. M. R. Parrondo and L. Dinis. Brownian motion and gambling: from ratchets to paradoxical games, 2014, Contemporary Physics 45 (2), 147 (2004); arXiv:1410.0485. DOI: 10.1080/00107510310001644836.

[2] Wikipedia. Examples of Markov chains — Wikipedia, The Free Encyclopedia, 2016. Online; accessed 6, December, 2016.

[3] Intermediate Linear Algebra. Class notes. Konstantine Mushegian. Fall 2016.