# Creating a RPSLV Bot using Non-Equilibrium Strategies

**Ilina Navani** and **Katie Kowalyshyn** and **Oma Hameed**
{ilnavani,kakowalyshyn,omhameed}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

**Abstract**

In this paper, we designed and implemented a bot that competes in a multi-round version of the game Rock-Paper-Scissors-Lizard-Spock (RPSLV). Our bot takes a strategic approach to the game and is inspired by the winning bot of the 1999 International RoShamBo Programming Competition. It aims to exploit common opponent strategies using nine different strategies of its own. By developing an algorithm to dynamically decide which strategy is most effective in defeating a given opponent, we were able to create a bot that performed well against dummy bots in internal testing as well as against other intelligent bots in a tournament.

## 1 Introduction

The RoshamBo Programming Competition was a worldwide competition in 1999 wherein participants submitted their own bots to compete in a tournament of Rock-Paper-Scissors-Lizard-Spock (RPSLV). Originally designed by Sam Kass and Karen Bryla, RPSLV is a version of the standard Rock-Paper-Scissors game which was made more complex to ensure a lower probability of tied outcomes (Krass 2012). Rather than having three possible actions, RPSLV has five, with each action beating two other actions and losing to two other actions. This game became so famous that it was also mentioned on the *Big Bang Theory* TV show. Below is an image that explains how actions defeat each other.

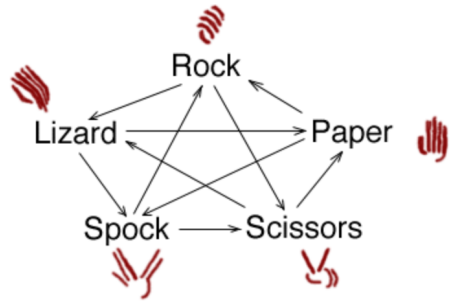


Figure 1: RPSLV Explanation
(Krass 2012)

Our goal was to implement a bot that is unique, yet utilizes aspects and strategies of other bots which have been proven to be successful at RPSLV. In order to do so, our research began by investigating articles from the 1999 competition. After reading several papers and analyzing results from the competition, we ultimately decided to base our bot on Iocaine Powder written by Dan Egnor, the winner of the competition (Billings 2000a).

We implemented multiple strategies that make predictions for the next best move based on patterns in our opponents' moves as well as our moves thus far. In this way, we were able to train our bot to dynamically respond to its opponent depending on the moves it is presented with. To further measure performance, our bot was placed in a tournament alongside other intelligent bots as competitors, in addition to a few dummy bots.

There were three papers that we consulted in order to develop and test our bot. The first, "The Effectiveness of Using a Modified 'Beat Frequent Pick' Algorithm in the First International RoShamBo Tournament," was written by Sony E. Valdez, Generino P. Siddayao, and Proceso L. Fernandez, and was published in the International Journal of Information and Education Technology in 2015 (Valdez and Fernandez 2015). This paper examines the logic behind Iocaine Powder, and specifically the impact of a frequency strategy on the overall performance of a bot. As discussed later in the Background section, a Frequency Analysis strategy is one of the primary strategies used by our bot. Hence, one of the main takeaways from this paper was to inform our strategy implementations.

The other two papers we consulted were written by Darse Billings, a game scientist from the University of Alberta. The first of his papers informed us of a few of the best bots from the competition, includingan in-depth analysis of Iocaine Powder and its strategies (Billings 2000a). This paper also gave numerical results in a table of the wins and losses in every match of the competition (Billings 2000a). The second of his papers includes further observations about the competition in 1999 (Billings 2000b). In particular, we used this paper to create dummy bots to internally test our bot before participating in a larger tournament (Billings

2000b).

Having introduced the RoshamBo Competition and its rules, the remainder of this paper will explain the logic behind our RPSLV bot and the decisions made during its implementation, as well as provide an analysis of its performance against an array of other bots.

## 2 Background

Having ranked highest in the 1999 RoshamBo Competition, Iocaine Powder gained tremendous attention for its adaptable and strategic approach to RPSLV. A strategy is defined as a method to predict the opponent's next move. At its core, Iocaine Powder uses multiple strategies, along with a predictive algorithm to select a move (Valdez and Fernandez 2015). When playing a match, each of its strategies makes a prediction regarding the next best move to play. It keeps score of which prediction wins a round, and continues to use that strategy to determine its next move. In addition, it also makes assumptions about the opponent's next move in two primary ways. First, it assumes that the opponent can predict that it will use a given strategy and therefore counter it, and second, it assumes that the opponent uses a given strategy against it. Hence, it can accordingly adjust its prediction to counter its prediction of the opponent's next move (Billings 2000a). For a more in depth description of Iocaine Powder's strategies, please refer to Valdez's paper (Valdez and Fernandez 2015).

Inspired by Iocaine Powder, our bot employs several statistical strategies in a game of RPSLV. It makes its next move using the prediction made by the strategy with the highest score thus far. The scoring mechanism we implemented is further described in the latter part of this section.

We have a total of five primary strategies, three of which are borrowed from Iocaine Powder (Random Guessing, Frequency Analysis and History Matching), and two of which target strategies that are commonly played in RPSLV (Rotation and Reverse Rotation). Our bot also accounts for the fact that a straightforward prediction could often fail as the opponent may anticipate our predictive logic and play accordingly. Hence, we include four additional strategies, called meta-strategies, that counter the predictions of our primary strategies, excluding the random guessing strategy, and enable us to effectively compete against other intelligent bots. Each of our strategies is explained in detail below.

### Strategies

**Random Guessing Strategy**
This strategy simply chooses one of the five possible actions at random. Since there is no way to guess our next move if we play at random, this algorithm is included as a hedge and prevents us from suffering extremely heavy losses if we start losing after a point and resort to choosing moves randomly.

**Frequency Analysis Strategy**
This strategy chooses a move that defeats the opponent's most frequent move. Having analyzed the opponent's history thus far, the algorithm finds their most frequent move and predicts that they will choose it. Utilizing such a strategy allows us to compete against opponents who tend to prefer playing certain moves over others.

**History Matching Strategy**
This strategy looks at sequences of moves that may be repeated in previous rounds. The maximum length of a sequence that can be analyzed in history is of length 20. Once history has been established in a match, this strategy is capable of winning against opponents that employ patterns periodically. We implemented two versions of this strategy: Opponent History Matching and Both History Matching, which are described below.

1. **Opponent History Matching**
   In this strategy, we predict the opponent's next move by finding repeating patterns in their history. For example, if the last three moves made by the opponent are Rock, Paper, and Scissors, we search for a point in their history where these 3 moves occurred in the same order. We then predict that the opponent will play the same move that they played after those three moves in the past, and can therefore choose a move to defeat their predicted move.
2. **Both History Matching**
   In this strategy, we predict the opponent's next move by finding repeating patterns in their history and our history. For example, if the last three rounds played by the opponent against us were paper against Rock, Scissors against Scissors, and Scissors against Rock, we search for times when the same three sets of moves were played, and assume that the opponent will play the same move that it played after those three rounds in the past.

**Rotation Strategy**
This strategy is based on the assumption that the opponent will choose their next move by rotating their last move to the right along the order of RPSLV. For example, if their last move is Rock, their next move will be paper and if their last move is paper, their next move will be Scissors. We then choose a move that defeats their next move.

**Reverse Rotation Strategy**
This strategy is the opposite of the Rotation strategy and assumes that the opponent's next move will be obtained by rotating their last move to the left. Implementing rotation strategies allows us to quickly defeat common approaches to a game of RPSLV without having to wait for history to be established to perform history matching.

**Meta-Strategies**
This set of strategies makes predictions based on the assumption that the opponent uses each of our strategies against us, excluding random guessing. Essentially, for each of the four strategies, we exchange the position of the opponent and ourselves, resulting in a new set of meta-strategies that make predictions of their own. For example, if a given strategy predicts that we should play Rock next, we assume that the opponent will play Paper or Spock to defeat Rock, so its meta-strategy will predict that we should defeat Paper and Spock by playing Lizard instead.

### Scoring Mechanism

Each strategy and meta-strategy has a running score associated with it. Initially, all scores are set to zero. After a round has been played, we check every strategy's prediction for that round to determine whether it predicted a winning move in regard to the opponent's actual move. If so, its score is incremented. Otherwise, its score is reset to zero. Instantly resetting the score to zero, instead of decrementing it, allows us to produce a faster response and switch strategies when the opponent changes their strategy. On the other hand, incrementing scores upon a correct prediction allows a strategy that is performing well, that is, a strategy that is successfully deducing the opponent's strategy, to continue being selected to make moves as the rounds progress.

Having implemented a total of nine strategies and meta-strategies, we were then able to train our bot to make predictions based on past performance. When playing against an opponent, each of our strategies outputs a prediction using their respective logics described above. Since every action in RPSLV can be defeated by two other actions, we randomly chose between the two winning actions in cases where strategies aim to directly defeat the opponent's move. Each strategy was also assigned a score as per our scoring mechanism. We therefore had nine predictions for any given move, and to determine the best move for that round, we simply chose the strategy with the highest score thus far. Given the nature of our scoring mechanism, we were able to take an adaptable approach and dynamically switch between strategies upon experiencing losses.

## 3 Experiments

In order to test the performance of our bot, we conducted internal testing against dummy bots, as well as participated in a tournament against other intelligent bots.

### Internal Testing

Our internal testing procedure was to develop simple bots and examine whether our bot was able to exploit them successfully. These bots were the same as some of the same as the ones used in the 1999 competition, and a brief description of each of them is provided below.

**Pi Bot:** This bot returns the modulo 3 of each digit of pi unless the digit is 0, in which case it is skipped.

**Princess Bride Bot:** This bot is based on TextBot from the 1999 competition and uses a monologue from the film *Princess Bride* in the form of a string. It takes the modulo 3 of each character and returns the corresponding move, restarting at the beginning of the string upon reaching the end.

**Switch-a-lot Bot:** This bot implements a biased strategy depending on its last move. When choosing its next move, it has a 12% chance of repeating the previous move, and the rest of the time it plays one of the other moves.

**Foxtrot Bot:** This bot follows the pattern: [rand prev+2 rand prev+1 rand prev+0] repeating, where rand is a random move and prev is the previous move of the opponent. Hence, every even move is related to the previous move, and every odd move is random.

**Add-Drift Bot:** This bot plays a random move with 50% probability. When it doesn't do so, it bases its decision on the sum of both players' previous moves.

**Anti-Rotn Bot:** This bot has a minimal use of hisotry and counts only the rotation in the opponent's move sequence. If the opponent repeats an action, rotates to the right by one, or rotates to the right by two, it is counted as a +0, +1, or -1 rotation respectively. It treats the actual choice between RPSLV as irrelevant and considers only the difference between the pair of moves. This bot also resorts to an optimal strategy of countering the opponent's last move in the case of 40 consecutive losses.

### Tournament

Our next set of experiments involved participating in a tournament against other non-equilibrium bots, that is, bots that don't play a Nash Equilibrium strategy. There were a total of 27 bots in the tournament, and every bot entered was played in a 10,000-round match-up against every other bot. If a bot won by a margin of at least 200 rounds against its opponent, it was considered a statistically significant win for that bot and a loss for the opponent. Alternatively, if the two bots were separated by fewer than 200 rounds won/lost, then it was considered a statistical tie.

## 4 Results

Our results from internal testing are shown in Table 1 and Figure 2 below. As seen in these visualizations, our bot had a statistically significant number of wins against four of the six of the bots used in internal testing. Anti-Rotn Bot, Pi Bot and Princess Bride Bot played patterns that our bot was able to exploit quickly. The two bots we had marginal wins against were Switch-a-lot Bot and Foxtrot Bot. This was expected, as Billings writes that these were among the hardest for players to beat in the 1999 tournament (Billings 2000b).

In the case of Switch-a-lot, since its strategy is biased towards repeating its previous move only 12% of the time, it is difficult to detect a pattern due to the incorporated randomness. For Foxtrot Bot, Billings informs us that its pattern was hard to detect for many players of the 1999 tournament that prioritized direct history algorithms, such as Iocaine Powder (Billings 2000b). As Billings writes, every even move that this bot makes is deterministic, while the odd-numbered moves are random. Strong pattern detector bots therefore did far better against this bot, as it is clear its strategy is based on opponent moves alternating with random moves. Our bot, like many of the bots in 1999, beat Switch-a-lot Bot and Foxtrot Bot by small margins. We can assume that our bot tends to incorrectly interpret their randomness for a strategy in conjunction with their completely deterministic moves. Our bot is constantly searching for a dominant strategy that can repeatedly defeat its opponent, and therefore overlooks the fact that a bot like Foxtrot Bot could be playing alternate moves randomly.

| Opponent Name | Wins | Ties | Losses |
|---|---|---|---|
| Anti-Rotn Bot | **9992** | 3 | 5 |
| Princess Bride Bot | **9358** | 234 | 442 |
| Pi Bot | **9352** | 228 | 420 |
| Add-Drift Bot | **4098** | 2052 | 3850 |
| Foxtrot Bot | **4037** | 1993 | 3970 |
| Switch-a-lot Bot | **4023** | 2024 | 3953 |

Table 1: Summary of Internal Testing Results

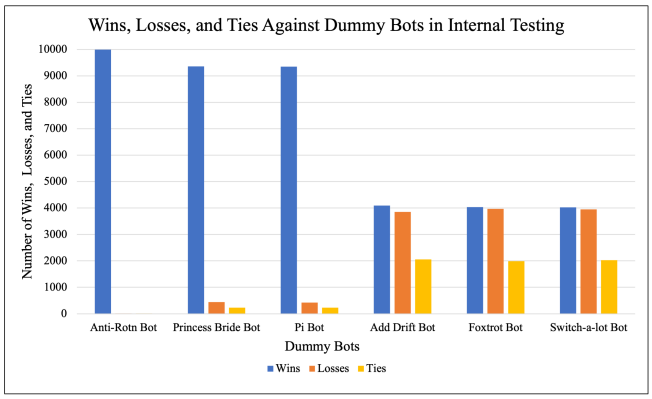


Figure 2: Internal Testing Results: Wins, Losses, and Ties

| Opponent Name | Wins | Ties | Losses |
|---|---|---|---|
| SolidAsARockBot | **10000** | 0 | 0 |
| ApeBot | **9992** | 3 | 5 |
| PrincessBrideBot | **9399** | 180 | 421 |
| QLBot | **7455** | 823 | 1722 |
| HeatBot | **7303** | 900 | 1797 |
| Spe2Bot | **5568** | 1209 | 3223 |
| StrategicBot | **5241** | 1333 | 3426 |
| Spe3Bot | **5161** | 1392 | 3447 |
| NootropicsBot | **5106** | 1274 | 3620 |
| CopiedBot | **5085** | 2070 | 2845 |
| DepressedBot | **5048** | 1320 | 3632 |
| WinLoseBot | **4966** | 1759 | 3275 |
| CoughingBot | **4925** | 1362 | 3713 |
| MixedBot | **4842** | 1077 | 4081 |
| CocainePowder | **4409** | 1583 | 4008 |
| ChasedBot | **4152** | 1902 | 3946 |
| NashBot | **4140** | 1906 | 3954 |
| CoolAsACucumberBot | **4119** | 1966 | 3915 |
| PatternBot | **4117** | 1930 | 3953 |
| CrunchyAsAChickenBot | **4050** | 2073 | 3877 |
| PrettyAsAPizzaBot | **4050** | 2006 | 3944 |
| Bot_B | **4050** | 1967 | 3983 |
| HistoricalProbBot | **4033** | 1965 | 4002 |
| FreqBot | **4027** | 2054 | 3919 |
| UltimateBot | **3993** | 2009 | 3998 |
| My2Bot | **3883** | 2071 | 4046 |

Table 2: Summary of Tournament Results

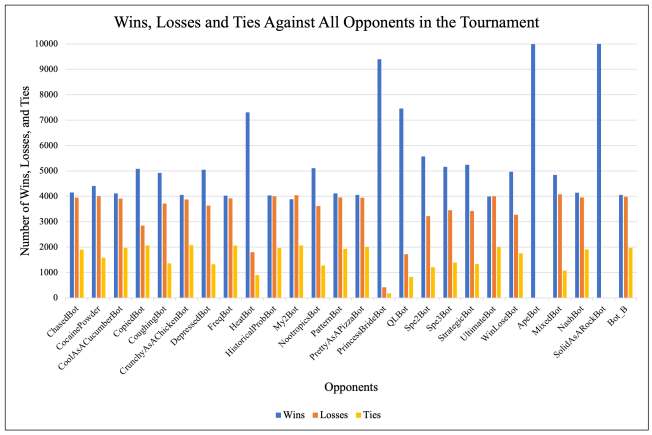


Figure 3: Tournament Results: Wins, Losses, and Ties

Our results from the tournament are shown in Table 2 and Figure 3 above. When analyzing these results, we see that by the definitions provided above regarding wins, ties, and losses, we only won or tied with all bots. However, by the pure number of games won, our bot lost marginally to two bots, My2Bot and UltimateBot. On the other hand, it won against all the remaining other bots, even if by a smaller margin than the previously defined significant win of 200 rounds.

Both My2Bot and Ultimate Bot use similar approaches of combining analytical and random strategies. Both bots keep track of the historical density of an opponent's actions by analyzing how often the opponent plays a specific action. In the case of My2Bot, it returns with that same average probability, one of the two actions that can beat the opponent's most played move. For example, if our most frequent move thus far is to play Rock 50% of the time, then on its next move, My2Bot will play Paper 25% of the time and Spock 25% of the time, in order to defeat Rock. In the case of Ultimate Bot, it will play Paper 50% of the time and Spock 50% of the time when our bot's most frequent move is Rock with a 50% probability. These strategies contain an element of randomness but are based on an analysis of the entire game. Hence, we believe that these bots performed better than ours because while we choose our next move

based on either frequency, history, or randomness, their strategies aim to combine all three.

While it is clear that our weakness is against bots that have random strategies incorporated into their analytical next moves, it is hard to entirely combat this. When two equilibrium bots play each other, it is expected that they will both win a certain percentage of the time. However, our bot is based on primarily deterministic strategies, which naturally cannot target random strategies. When bots combine unbeatable random strategies with deterministic learned strategies, our bot attempts to analyze patterns in their moves. A failure to do so results in none of our strategies dominating over the others, and therefore, we essentially play randomly as well.

## 5 Conclusions

Our goal to create a bot that utilizes non-equilibrium strategies to excel in a game of RPSLV was ultimately successful. By relying on a combination of nine strategies and a predictive algorithm to select our next best move, we were able to achieve improved performance when internally testing against 6 dummy bots. After exploiting these bots to the best of our ability, we competed in a tournament with 26 opponents and won or tied against all but two. The two intelligent bots that we lost to both implemented similar strategies that rely on the probability of the opponent playing a specific move. For future implementations of our bot, it may be beneficial to to perform additional internal testing to identify dominating strategies and weigh their usefulness against each other in terms of how effectively they defeat different types of opponents, as well as implement additional strategies and meta-strategies that enable us to win more quickly and frequently.

## 6 Contributions

We were all able to offer input on every aspect of the assignment and complete the code as well as the paper during our in-person meetings. We each proof-read the entire document as well.

## References


Billings, D. 2000a. News, Information, Tournaments, and Reports: The First International Roshambo Programming Competition. *ICGA Journal* 42–50.

Billings, D. 2000b. Thoughts on Roshambo. *ICGA Journal* 3–8.

Krass, S. 2012. ROCK PAPER SCISSORS SPOCK LIZARD .

Valdez, Sony E., S. G. P., and Fernandez, P. L. 2015. The Effectiveness of Using a Modified "Beat Frequent Pick" Algorithm in the First International RoShamBo Tournament. *International Journal of Information and Education Technology* 740–747.