AA 228 Final Report: Rock, Paper, Scissors, Go! Learning Probabilistic Models with Game Theory & Bayesian Model Estimation

Nadia Galindo Méndez[1] and Allan Shtofenmakher[2]

Stanford University, Palo Alto, CA, USA.

In spite of its outwardly simple appearance, rock, paper, scissors (RPS) is a game of decision making under uncertainty that forms an excellent testbed for experimentation with various learning algorithms as well as generation of artificial intelligences (AIs) that can outperform their uninformed opponents. It is hypothesized that, with sufficient exposure to rounds of RPS, an AI can learn to predict its opponents' respective action probabilities and use this information to increase their likelihood of attaining victory. A tournament simulation against static AIs is developed to test this hypothesis and in an effort to improve upon the results of previous work, which, thus far, has developed AIs that boast only limited improvement over random action generators. Two solution methods—a modified multi-armed bandit (MMAB) approach and a game theory approach—are developed and simulated, and the corresponding results are presented and compared. The game theory approach is subsequently determined to be unambiguously superior due to the MMAB approach's failure to balance exploration and exploitation. Further experimentation with and analysis of the game theory formulation reveals an exceptional ability to predict opponent action probabilities—exceeding 95% accuracy after fewer than 100 RPS rounds—and exploit this information to yield remarkably improved performance over previous work, winning no less than 4% more rounds than the current state of the art under similar conditions. Future work may involve exploring additional approaches, such as formulating the problem as a Markov decision process and attempting to estimate the state transition probabilities of the opponents in an effort to further improve AI performance.

I. Introduction and Problem Description

On the surface, a typical game of rock, paper, scissors (RPS) may appear to be a child's game of chance, featuring only randomness and luck, but no skill. A closer inspection, however, reveals a game of decision-making under uncertainty and multi-agent planning. Even though each player can take only one of three actions—rock, paper, or scissors—the uncertainty of this game is simple and elegant—no player knows a priori what their opponent's action will be. However, it is hypothesized that, using probability, statistics, and prior knowledge, a player—or, more specifically, an artificial intelligence (AI)—can learn to predict how likely it is for their opponent to take each of those three actions, and then use these predictions to their advantage in order to increase their own chances of winning.

A simulation program has been developed to test this hypothesis. In each simulation, an AI that has never previously played a game of rock, paper, scissors will enter a competition with m opponents (nominally 1000) and face off against each one n times (nominally 10 times). Each of the n "face-offs" is referred to as a *round*, while a set of n rounds against one opponent is referred to as a *battle*. In other words, in each tournament, the AI participates in m battles and a total of $m*n$ rounds.

The AI initially has an uninformed prior regarding the population statistics concerning RPS—a Gaussian distribution over how often people tend to choose rock, paper, or scissors, as described in [1]. In this particular exercise, the population statistics are arbitrarily defined according to Table 1. These values are used to generate the remaining results presented in this paper.

Table 1 Population Statistics for each action.

	Mean	Standard Deviation
Rock	0.50	0.10
Paper	0.20	0.05
Scissors	0.30	0.08

Over the course of the competition, the AI will attempt to learn the population statistics and use this information to choose its actions in an effort to win the tournament. Winning the tournament, in this scenario, is defined in two ways, both of which are pursued by the AI:

1) The first definition involves the accumulation of more *points* than any other player. Points are distributed to each player at the end of each round as follows: +1 for winning, 0 for tying, and -1 for losing. According to this definition, the AI maximizes its likelihood of winning the tournament by simply maximizing the number of points it earns, or, equivalently, maximizing its final *score* (sum of all rewards earned).
2) The second definition involves defeating more opponents than any other player. In this case, the victor in any battle between two competitors is the individual who earned more points over the course of the n rounds between those two individuals. Each opponent that the AI faces will have their own preferences for choosing rock, paper, and scissors, which are sampled from the population-level Gaussian distributions. To win the tournament in this way, the AI must use its growing knowledge of the population statistics to defeat as many individual opponents as possible in "best of n" battles.

Several studies, as described in [1], [2], [3], and others, on this subject—that is, designing AIs that seek to maximize performance in RPS games—have already been conducted. However, the results of state-of-the-art approaches demonstrate less-than-impressive performance metrics, indicating an opportunity to improve upon previous work and generate an AI with a greater likelihood of winning a given tournament.

For example, [1] models the population statistics—that is, the preferences of the population for selecting rock vs. paper vs. scissors—using Gaussian distributions and implements various state-of-the-art machine learning algorithms in an effort to maximize the number of games won against 650 players. However, this AI did not perform much better than a random action generator, winning only between 32% and 36% of games in real-world simulations [1]. From this information, it can be concluded that the problem of generating an AI that can reliably outperform a random action generator in RPS competitions has not been solved—until now. In the following sections, this paper will describe a solution to this problem that, assuming the same population statistics as in [1], can reliably win approximately 40.12% of games.

Similarly, [2] uses Markov chain and maximum likelihood methods to create an AI that can learn a single opponent's behaviors, predict their next action based only on their previous 2–6 actions, and use this information to increase its likelihood of winning the next round. In this case, after approximately 13,000 "training" battles, the AI could expect to win approximately 60% of all battles against that same opponent [2]. Although this win rate is respectable, the number of battles required for training is far from optimal. The present paper presents an alternative approach that can yield a similar win rate with three orders of magnitude fewer training battles. This alternative approach is described in the upcoming sections.

II. Algorithms and Methodology

The solution to this problem was initially formulated using a modified multi-arm bandit (MMAB) approach, featuring three arms (rock, paper, and scissors), three outcomes (win, tie, lose), and 9 Dirichlet parameters, which accounted for all possible combinations of rock-paper-scissors and win-tie-lose [4]. A Julia program, referenced in the appendix, was created to solve the problem according to this approach, but, as is standard with MMAB formulations, a problem of exploration vs. exploitation arose [4]. That is, without an exploration strategy, the AI

always chose to exploit. As a result, the AI would often choose a suboptimal policy that corresponded to a local optimum that was not necessarily a global one. However, with an exploration strategy such as epsilon-greedy, although the AI had a higher likelihood of ultimately identifying the optimal policy, it would necessarily "lose" some points during the exploration phase and, consequently, end up with a lower final score than the exploit-only strategy could achieve if it converged to the global optimum [4].

For simulations with 1000 battles and 10 rounds per battle, histograms of the final scores for the MMAB approach with several different epsilon values are shown in Fig. 1 in the appendix, which contains all figures. Lower epsilon values clearly depict multimodal distributions, suggesting that the AI sometimes fails to exploit the optimal policy; however, higher epsilon values depict unimodal distributions, but centered at lower final scores, suggesting that the AI successfully discovers the optimal policy only after a significant cost to cumulative utility. Other exploration strategies, such as softmax and interval exploration, were considered, but they suffered the same fate—any investment in exploration resulted in at least a partial loss to exploitation [4].

With this in consideration, an alternative approach was considered and developed—one which implemented game theory instead of a modified multi-armed bandit [4]. The utility table of two agents playing RPS is shown in Table 2. Note that this table, despite being the same utility table presented in [3], was derived independently of previous work.

Table 2 Utility table for each possible combination of actions in a round of RPS.

		Opponent		
		Rock	Paper	Scissors
AI	Rock	0,0	-1,1	1,-1
	Paper	1,-1	0,0	-1,1
	Scissors	-1,1	1,-1	0,0

With this approach, in the jargon of game theory, the AI will have a model of the utilities of its opponents—nominally, +1 for winning, 0 for tying, and -1 for losing—but will not have a probabilistic model of the behavior of the other players or the population [4]. However, by observing and keeping count of the actions of its opponents, it can use Bayesian model estimation with just 3 Dirichlet parameters to learn/estimate both the population statistics and the individual preferences for each of the 3 actions and exploit this knowledge to win as many rounds and battles as possible [4]. In spite of being much simpler than the MMAB formulation, this approach appears to converge to the optimal policy in every simulation. This result is proven in the next section.

A unimodal, Gaussian histogram of final scores for the game theory approach for a simulation with 1000 battles and 10 rounds per battle is shown in Fig. 2. This same histogram is compared to the corresponding histograms for the MMAB approach in Fig. 3, which clearly shows the superiority of the game theory approach.

The high-level pseudocode algorithms for both approaches are presented below. More accurately, only one algorithm is presented because, remarkably, at this high level, the algorithms are the same. The only differences, fundamentally, are the Dirichlet parameters, how they are updated, and how they are used to determine the AI's action in each round. Otherwise, the algorithms are identical.

Algorithm RockPaperScissorsGo
Global: *Method* ∈ [MMAB,GameTheory] . *NumberOfOpponents*. *NumberOfRounds*. *PopulationStatistics*.
ε∈ [0,1]. *WinCount* ← 0. *TieCount* ← 0. *LossCount* ← 0. *ActionSpace* ← Rock, Paper, Scissors.
OutcomeSpace ← Win, Tie, Loss. *RewardSpace* ← 1,0,-1. *PopulationAlphaParameters* ← ones.
begin
for m ∈ [1,*NumberOfOpponents*]
Rock(*μ*, *σ*), *Paper*(*μ*, *σ*), *Scissors*(*μ*, *σ*) ~ *PopulationStatistics*
OpponentAlphaParameters ← *ExpectedConformity* * norm(*PopulationAlphaParameters*)

```
        for n ∈ [1,NumberOfRounds]
            ActionAI = AIACTION(ε, OpponentAlphaParameters)
            ActionOpponent= OPPONENTACTION( Rock(μ, σ), Paper(μ, σ), Scissors(μ, σ) )

            Outcome ← ROUND( ActionAI, ActionOpponent )

            if Outcome=Win
                WinCount ← WinCount + 1
            elseif Outcome=Tie
                TieCount ← TieCount + 1
            elseif Outcome=Loss
                LossCount ← LossCount + 1

            if Method=MMAB
                Update OpponentAlphaParameters and PopulationAlphaParameters based on Outcome
                and ActionAI
            elseif Method=GameTheory
                Update OpponentAlphaParameters and PopulationAlphaParameters based on
                ActionOpponent
end

function AIACTION (ε, OpponentAlphaParameters)
        r ~ [0,1]
        if 0 ≤ r < ε/3
            return Rock
        elseif ε/3 ≤ r < 2ε/3
            return Paper
        elseif 2ε/3 ≤ r ≤ ε
            return Scissors
        else
            Convert OpponentAlphaParameters to expected probabilities using Bayesian estimation methods
            Determine expected value for each action
            return action that maximizes the expected value
end

function OPPONENTACTION( Rock(μ, σ), Paper(μ, σ), Scissors(μ, σ) )
        Rock(μ), Paper(μ), Scissors(μ) ← norm(Rock(μ), Paper(μ), Scissors(μ))
        Resolve [0,1] into three sections separated by Rock(μ) and Rock(μ)+Paper(μ)
        r ~ [0,1]
        Determine opponent action by comparing r to the three sections
        return opponent action
end

function ROUND(ActionAI, ActionOpponent)
        Compare ActionAI & ActionOpponent to determine outcome according to the general rules of RPS
        return outcome
end
```

III. Results of Simulation

The following results of simulation are presented for the game theory approach to solving this problem. In particular, the following questions are answered: Using game theory and Bayesian estimation methods, can the AI

learn to predict its opponent's action probabilities? Can the AI reliably use this information to win the tournament in both ways and outperform both a random action generator and the state of the art?

As mentioned earlier, to test the hypothesis presented in the introduction, it was first necessary to determine whether the AI could, in fact, predict the probabilities with which an opponent would select each action. In the game theory formulation, this is nearly equivalent to determining whether the AI could predict the respective population mean values. Fig. 4 suggests both that the AI can, in fact, perform this feat, as well as how quickly it can do so. In particular, the AI's estimates of the population means are fairly accurate (to within 10%) after 10 rounds, moderately accurate (to with 5%) after 100 rounds, and very accurate (to within 2%) after 1000 rounds. By 10000 rounds, the AI's estimates of the population means appear to have converged to their respective actual values.

The next task was to demonstrate the AI's ability to win rounds (i.e., victory in the first definition) more reliably than a random action generator and, ideally, the state of the art presented in [1]. Fig. 5 depicts a representative win-tie-loss distribution for a tournament with 10000 rounds—approximately 50% wins, 20% ties, and 30% losses, which is nearly exactly consistent with the population statistics presented in Table 1. In this case, the AI wins approximately 16.7% more rounds than a random number generator and 14% more rounds than the state of the art in [1]. Moreover, Fig. 3 from before suggests a *unimodal* Gaussian distribution centered around a final score of approximately +2000 out of 10000 rounds. The unimodality of this distribution suggests that the AI tends to exploit a single policy, and the fact that this final score ratio is almost exactly consistent with the expected utility of the optimal policy (highlighted in green) in Table 3 suggests that this single exploited policy coincides with the optimal policy. All together, these facts combined demonstrate that, in every simulation involving the game theory approach, the AI always identifies the optimal policy and exploits it to yield substantially greater performance than both a random number generator and the current state of the art.

Table 3 Population-level expected utilities for each dominant policy.

Dominant Policy	Pop. Win Rate	Pop. Tie Rate	Pop. Loss Rate	Exp. Utility
Rock	0.30	0.50	0.20	+0.10
Paper	0.50	0.20	0.30	+0.20
Scissors	0.20	0.30	0.50	-0.30

At this stage, it is worth noting that the results of the game theory approach are highly dependent on the population statistics. In other words, in the worst case, if the population indicates no preference for any particular action, this method is no worse than a random action generator. However, in reality, as [1] discovered, the population tends to choose scissors approximately 40.12% of the time, so, in practice, this AI will still win 6.79% more rounds than a random action generator and 4.12% more rounds than the current state of the art.

The final task was to demonstrate the AI's ability to win battles (i.e., victory in the second definition) more reliably than the state of the art presented in [2]. In essence, this task requires the AI to learn each individual opponent's action preferences as it battles that opponent in order to increase its likelihood of defeating them. To this end, it is desirable to establish a means by which to convert the relatively large population Dirichlet parameters to more tractable opponent-specific Dirichlet parameters that can be noticeably influenced by the actions of the opponent over the course of the battle. In an effort to accomplish this, an *expected conformity* factor was introduced to represent the AI's belief that an individual will conform to the population statistics. As depicted in the algorithms, this expected conformity factor is multiplied by the normalized population Dirichlet parameters to determine the current opponent's individual Dirichlet parameters. A low expected conformity factor indicates a high belief in individuality; a high expected conformity factor indicates, somewhat obviously, a high expectation of conformity.

A sweep of expected conformity factors from 0 to 20 in increments of 0.01 was conducted for simulations with 1000 battles and 10 rounds per battle in an effort to find the value of expected conformity that maximizes the average percent of battles won per tournament. The results, shown in Fig. 6, suggest, in general, a quasi-logarithmic relationship between average percent of battles won and expected conformity factor and, in particular, that a high belief in conformity tends to yield better performance. In other words, the AI is more likely to defeat an opponent if it believes that opponent is unlikely to deviate from the population statistics. In any case, regardless of the values of the expected conformity factor, Fig. 6 suggests that 10 rounds per battle appears to be sufficient to produce a win rate in excess of 60%—which is already competitive with the results in [2].

At this point, it was suspected that the time scale over which the AI was able to learn from its opponent—that is, the number of rounds in which the AI faces the same opponent before continuing to the next challenger—may have significantly influenced the previous result. In order to decouple the two effects, a separate sweep of the number of rounds per opponent was conducted for a fixed expected conformity factor. The results, shown in Fig. 7, indicate that the AI is significantly more likely to defeat an opponent if it has more opportunities to learn from them. In particular, in comparison to [2], which required 13000 "training" battles to achieve a 60% win rate, the game theory formulation could achieve a similar win rate in less than 10 training battles, which indicates a substantial improvement over the state of the art, simply due to a change in solution formulation.

IV. Conclusion

In summary, the results discussed in this paper confirmed the hypothesis that an AI with sufficient exposure to RPS games could, in fact, learn the population action probabilities, estimate individual opponent action probabilities, and exploit this information to determine an optimal policy to maximize both the number of rounds won and the number of opponents defeated. In particular, the game theory formulation for simulating the behavior of an artificial intelligence in a rock, paper, scissors tournament significantly improved the performance of the AI relative to both the modified multi-arm bandit formulation and the previous state of the art. This discovery suggests the possibility of even greater performance enhancement in the form of alternative solution methods. In particular, reframing the problem as a Markov decision process, recording the past history of state/action transitions, and attempting to estimate the state/action transition probabilities of the opponents appears to be a promising extension of the presented work, and future work in general is expected to involve exploring this option.

Contributions

The authors contributed equally to the research and development of this project and the associated paper. Allan Shtofenmakher was responsible for writing the majority of the text for the paper, designing the algorithms, and programming the primary Julia codes and supplementary functions. Nadia Galindo Méndez generated all tables and figures and modified the primary Julia codes and supplementary functions and generated additional MATLAB programs to accomplish this task. She also converted the algorithms to pseudocode and formatted the paper to ensure compliance with AIAA standards.

Appendix

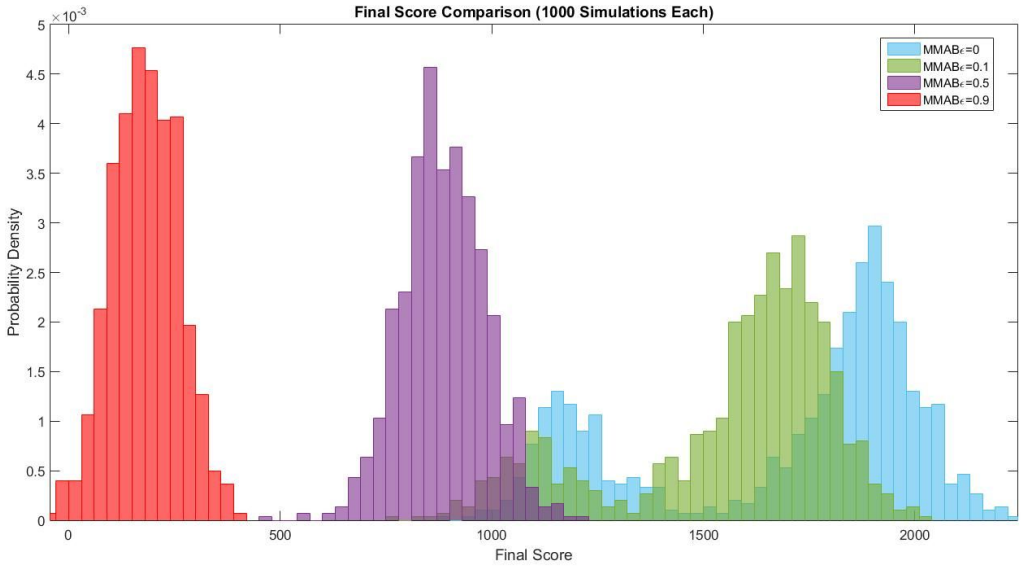

Fig. 1 Histograms of the final scores for the MMAB approach for different epsilon values.

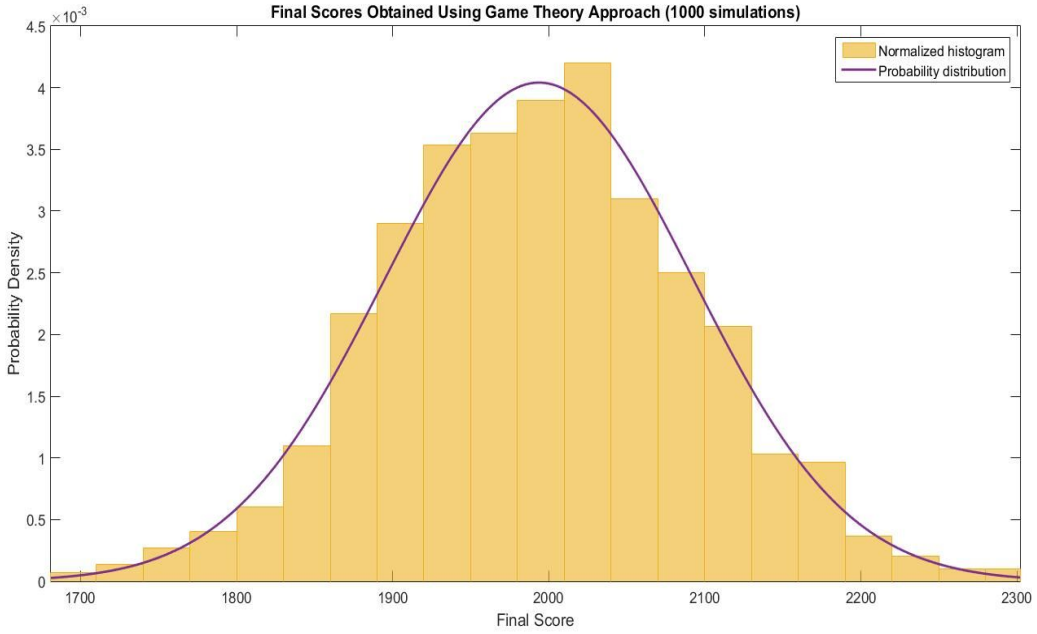

Fig. 2 Gaussian final-score histogram and resulting probability distribution for the game theory approach.

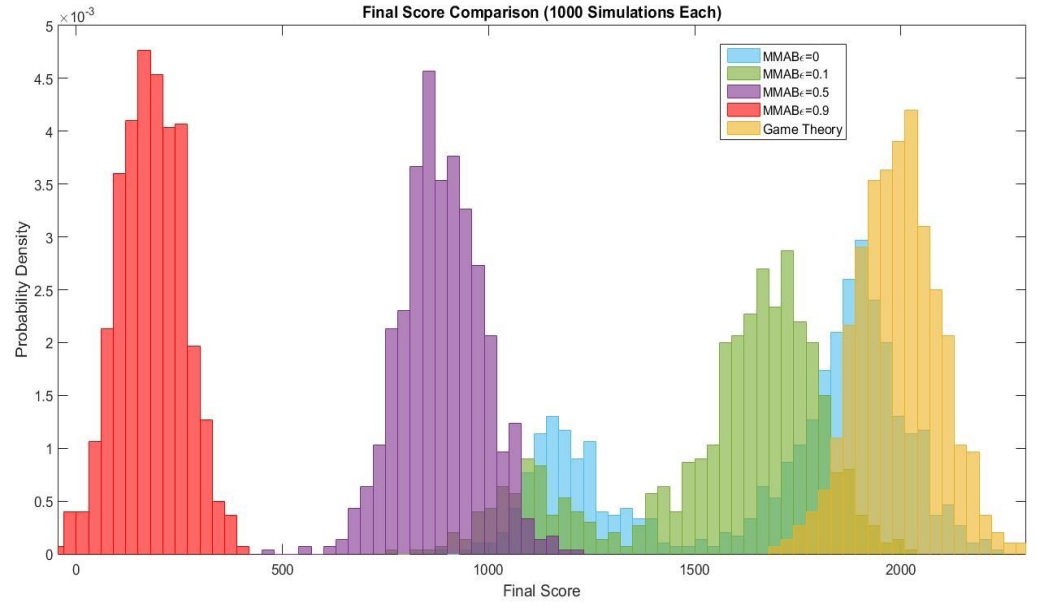

Fig. 3 Comparison of final-score histograms for the MMAB approach and game theory approach.

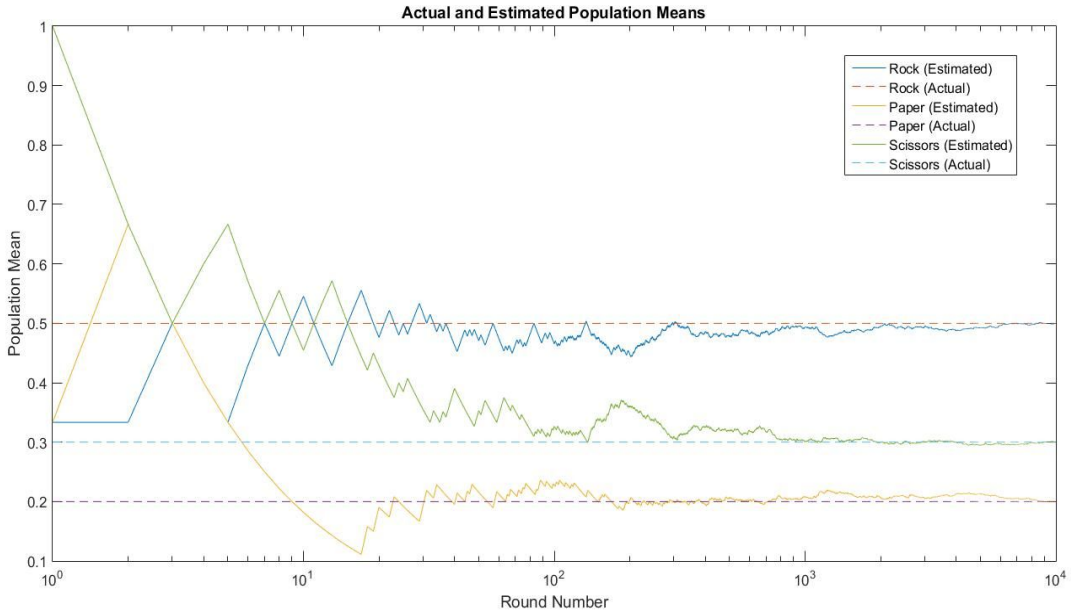

Fig. 4 Convergence of AI's expected pop. means to actual pop. means using game theory approach.

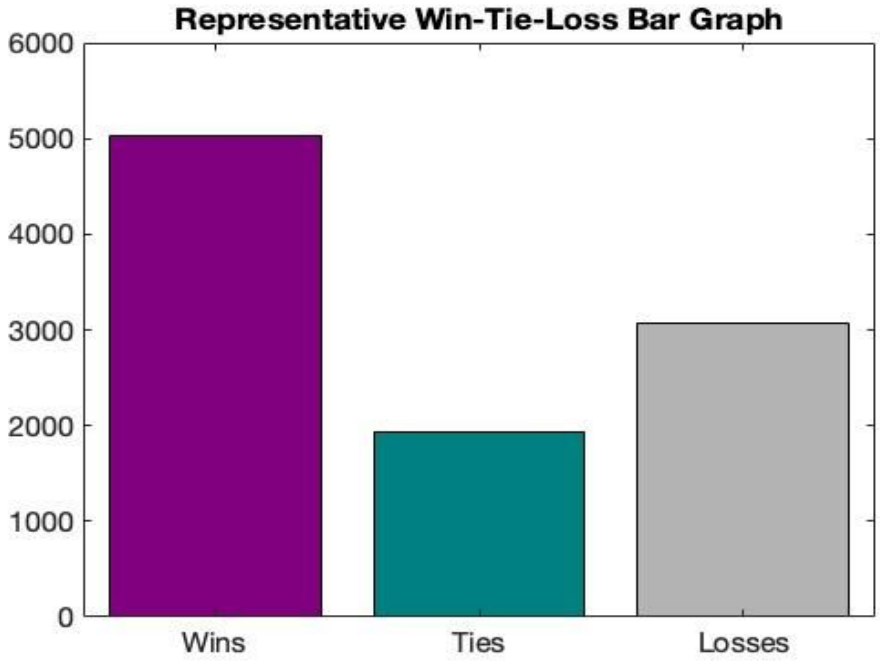

Fig. 5 Representative outcome distribution for a tournament with 10000 rounds.

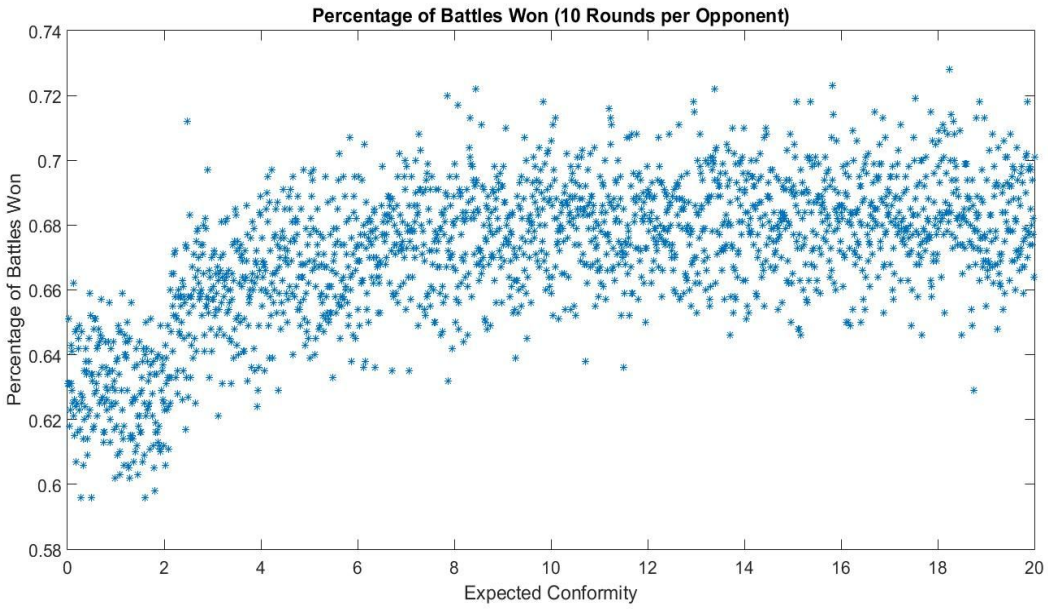

Fig. 6 Relationship between average percent of battles won and expected conformity factor.

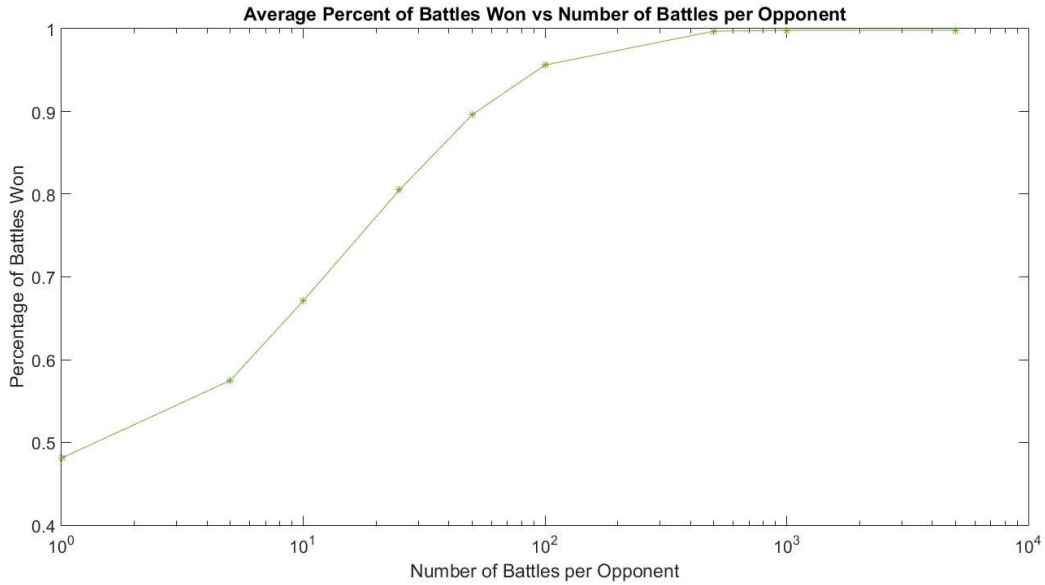

Fig. 7 Effect of number of rounds per opponent on average percent of battles won.

Program Archive

All programs used in this project can be found on GitHub: https://github.com/ashtofen/AA-228-Final-Project

Acknowledgments

The authors would like to thank Professor Mykel Kochenderfer, for his thoroughly enjoyable course and his unending support, and Nolan Johnson, Robert Moss, and Kunal Menda for their suggestions and feedback at various stages of this project. The authors would also like to thank each other for their equal contributions to the project and continued friendship. Nadia Galindo Méndez thanks CONACYT for the financial support received to pursue her degree.

References

[1] Pozzato, G., Michieletto, S., and Menegatti, E, "Towards smart robots: rock-paper-scissors gaming versus human players", *Proceedings* of the *AI*IA National Workshop: Popularize Artificial Intelligence*, AI*IA, Turin, Italy, 2013, pp. 89-95. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.6833&rep=rep1&type=pdf#page=89 [retrieved 8 October 2019].

[2] Pomerlau, N., "Rock Paper Scissors", Worcester Polytechnic Institute, Worcester, MA, 2013 (unpublished). URL: https://www.neilpomerleau.com/posts/wp-content/uploads/rps-report.pdf [retrieved 8 October 2019].

[3] Batzilis, D., Jaffe, S., Levitt, S., List, J.A., and Picel, J., "How Facebook Can Deepen our Understanding of Behavior in Strategic Settings: Evidence from a Million Rock-Paper-Scissors Games", Harvard University, Cambridge, MA, 2016 (to be published). URL: http://scholar.harvard.edu/files/jpicel/files/rps.pdf [retrieved 8 October 2019].

[4] Kochenderfer, M., *Decision Making Under Uncertainty: Theory and Application*, 10th ed., The MIT Press, Cambridge, MA, 2015, Chaps. 1-5.