Coevolution of Rock-Paper-Scissor Players

Tawfiq Jawhar
Concordia University
Montreal, Canada
tawjaw@gmail.com

***Abstract*— In the game Rock Paper Scissor, two players compete against each other. Each player chooses one of the three options. Rock beats scissor, paper beats rock and scissor beats paper. If both players choose the same move a draw occurs. As simple as this game sounds, however it requires adaptation to be able to counter the moves of the other player. In this paper we discuss an evolutionary algorithm able to play rock paper scissor. The purpose is to study the coevolution of two evolutionary algorithms playing against each other.**

Keywords—coevolution; RPS; evolutionary algorithms.

I. INTRODUCTION

Rock Paper Scissor is a zero-sum game. A zero-sum game is a situation in game theory which one person's gain equals another's loss, resulting in a net benefit of zero [1]. In theory, if two people are playing RPS for x rounds, each player will have a winning score around x/2. This situation would be the optimal RPS game where every player is adapting to the other player's strategy of playing. For example, if the first player is always playing Rock then the second player will adapt and play Paper in order to win.

There are two analysis approaches in game theory towards humans' decisions in RPS. Classic game theory suggest that every player will randomize the choice to avoid being exploited. Evolutionary game theory of bounded rationality "predicts persistent cyclic motions" [2].

The design of our evolutionary system and representation of each individual (player) is based on the bounded rationality theory in which every player will be following a cyclic motion close to a pattern. However, this motion is not static. The pattern is relative to the history of the rounds played, taking into consideration what this player played versus the opponent's choice.

II. METHODOLOGY

The idea of evolutionary algorithms is inspired from nature and the emergence of life on Earth. There are many different variations and designs of EAs, but the concept stays the same. In any population of individuals within some environment, competition for survival occurs (survivor of the fittest). The higher the fitness of an individual the higher the chance of its selection to survive by nature. The process in which an evolutionary algorithm works is as follows: given a population of individuals, 1. evaluate the fitness of all individuals in the population, 2. select parents from the population to 3. generate the offspring of individuals using mutation and crossover, and finally 4. select from the new generated offspring the next population based on their fitness (survivor of the fittest) [3].

A. Individual

Every individual contains a set of RPS rules. These rules allow the individual to decide what move to play at a certain round depending on the history of the game. A rule is defined as follows: ***<conditions>:<action>***. For an individual to play the move in the **<action>**, the conditions of this rule must be satisfied.

A condition is any two possible moves played by both players; **RP, RR, RS, PR, PP, PS, SR, SP, SS**. The first move in the condition (left) is always in the perspective of player one and the second move (right) is in the perspective of player two. However, one condition can also include multiple conditions. For example, a condition can be **RP or PP or SR**.

An example of a rule: **RS (RR or SS): R**

If in the previous round first player played **R** and second player played **R** or first player played **S** and second player played **S**, and in the round before that the first player played **R** and the second player played **S** then the rule is satisfied and the individual can play **R**.

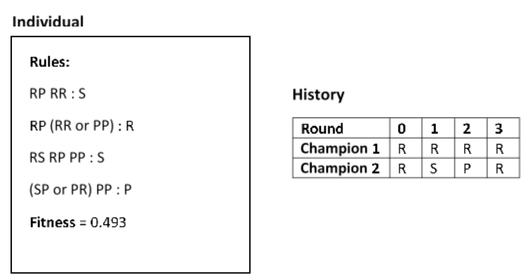

Fig. 1. For the given individual the first and second rule are satisfied by the history. All other rules will be discarded. The score of each rule is as follows:
Rule 1: 9 (RP) + 9 (RR) = 18
Rule 2: 9 (RP) + 8 (RR or PP) = 17
Rule 1 has a higher score; therefore, Scissors will be played from this individual.

Because an individual contains a set of rules then multiple rules can be satisfied at once. To choose which rule's action to be played, a score scale is introduced to give a score to a rule based on its accuracy. The more ORs in a condition there is the less the score is. However, if none of the rules of an individual is satisfied, then the individual plays the last move it played.

For example, the condition **(RR or SS or RS or PS)** has a score of 5, but a condition of **RR** has a score of 9. The score of a rule is the sum of all the scores of the conditions in a rule. If multiple rules are satisfied the rule with the highest score will determine the individual's action (move to be played by this individual). Figure 1 gives an example of an individual with a set of rules, and the chosen move to be played from the rules.

B. Fitness

Every individual starts with a fitness of 0.5 + noise; where noise is between -0.05 and 0.05. the fitness of an individual determines how well this individual is playing compared to the rest of population. After every round of the game the fitness of all individuals are updated using a reward.

Fitness = fitness*(1 – alpha) + reward*alpha

Alpha is a number between 0 and 1. If the individual wins the round then the reward is +1, if it loses then the reward is -1 and if a draw occurs the reward is 0.

C. Coevolution

Coevolution is a competitive environment of two evolving populations competing against each other. Every population will try to win against the other, and every time this happens the other population will evolve to counter the winning of the other one. The fitness of every population is determined by the other population. This scenario is found in nature where species effect other species in determining the fitness of an organism [3].

Using coevolution to evolve two populations playing rock paper scissor has shown interesting results. As both populations are adapting and changing their game strategies to win against the other. Whenever a population wins the most over a certain amount of rounds it is expected to lose the most over a certain amount of rounds after that (the graphs in the RESULTS AND ANALYSIS section will show that).

D. Champion of a Population

Every population consists of a fixed number of individuals. With every round of RPS one individual from the first population competes against one individual from the second population. This chosen individual is called the champion, and usually it is the fittest individual of all its population. All the individuals in population 1 plays against the champion of population 2 and vice versa. The history of the game saves the actions of the champions only. The champion of every population effects the fitness of every individual in the other population, hence the coevolution. Figure 2 shows the connection of each individual in a population with its opponent.

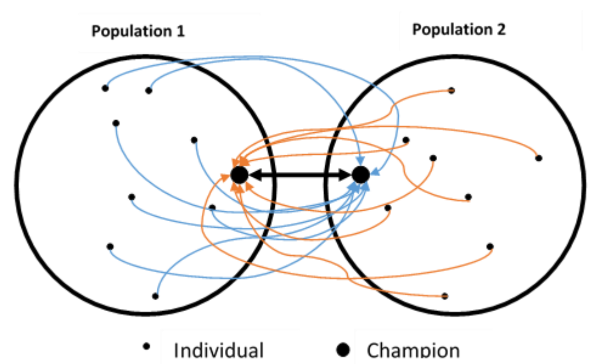

Fig. 2. The game flow of individuals inside the two population. Every individual in a population competes against the champion of the other population. However, all individuals check the satisfaction of their rules based on the history of the two champions only.

III. THE ALGORITHM

The algorithm is basically the process of two populations competing in Rock Paper Scissor. After every specific amount of rounds both populations evolve. The evolution will generate a new population that maintains diversity between the individuals, which is important in a game like RPS. Figure 3 and 4 explains the flow chart of the coevolutionary algorithm.

The two populations play against each other for few rounds before evolution happens. The number of rounds to play before evolution affects the coevolution. If evolution of each population happens after every round of the game, then the individuals will be adapting based on one round only and not a cyclic pattern of multiple rounds.

PSEUDO CODE explaining the evolution of one population: parent selection with elitism and the generation of an offspring by mutating a parent individual.

INITIALIZATION
parent size is set to 20% of the population size
parent vector is empty
offspring vector is empty
number of offspring is set to 0
END INITIALIZATION
BEGIN EVOLUTION
BEGIN PARENT SELECTION
//ELITISM. Always keep the most fitted individual
choose the fittest individual in the population and push it to the parent vector
REPEAT
    //TOURNAMENT SELECTION
select 3 individuals randomly from the population
choose the fittest individual between the 3
IF the difference of fitness between two individuals is less than 0.001 **THEN**
    choose the individual which has higher average score of rules
    //average score of rules = sum of all rules' scores in an individual / number of rules
END IF
push the selected individual out of the 3 individuals to the parent vector
UNTIL parent vector size is equal to parent size
END PARENT SELECTION

BEGIN OFFSPRING GENERATION
FOR EVERY individual in parent vector
    number of offspring is set to 0
    REPEAT
    //generate an offspring by mutating the selected individual from the parent vector
    create a new individual by copying the set of rules of the selected parent
    assign a fitness to the new individual of 0.5 + noise
    //mutate individual
    BEGIN MUTATION
    generate a random number "random" between 0 and 1
    CASEWHERE random is between:
    0-0.2: DELETE MUTATION:
        IF number of rules is more than 1
        select a random rule and delete it
    0.2-0.4: ADD MUTATION:
        generate a random rule and add it to the set of rules of the individual
    0.4-0.9 MODIFY MUTATION:
        choose a mutation type randomly
        select a random rule from the rule set of the individual
        CASEWHERE mutation type is:
            ADD:
            select a random position from the conditions of the selected rule
            add a randomly generated condition at the selected position
            DELETE:
            IF number of conditions in the selected rule is bigger than 1 **THEN**
                select a random position from the conditions of the selected rule
                delete the selected condition
            END IF
            MODIFY:
            select a random position from the conditions of the selected rule
            switch this condition with a randomly generated condition
            OR
            switch the action of the selected rule with a randomly generated action
            NO MUTATION:
            do not do anything
    0.9-1 NO MUTATION:
        do not mutate the offspring

    END MUTATION
    push the offspring to the offspring vector
    increment number of offspring
    UNTIL number of offspring is 4.
END OFFSPRING GENERATION
combine the individuals in the parent vector and the individuals in the offspring vector and assign them as the new generation of the population, and kill the old generation
sort the individuals of the population by their fitness
assign the fittest individual as the champion of the population
END EVOLUTION

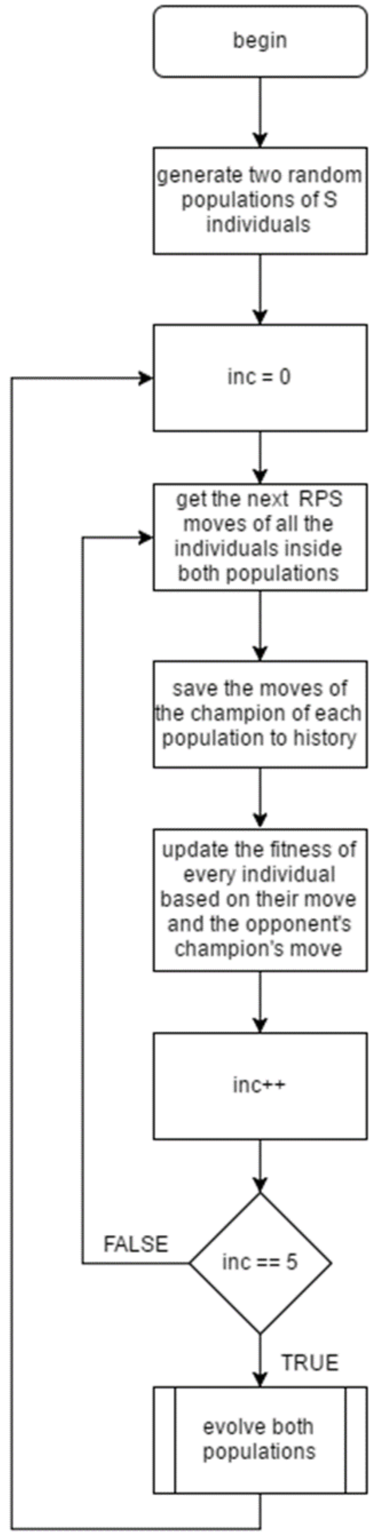

Fig. 3. Flowchart representing the algorithm of the coevolution process. Evolution is happening every 5 rounds of RPS game.

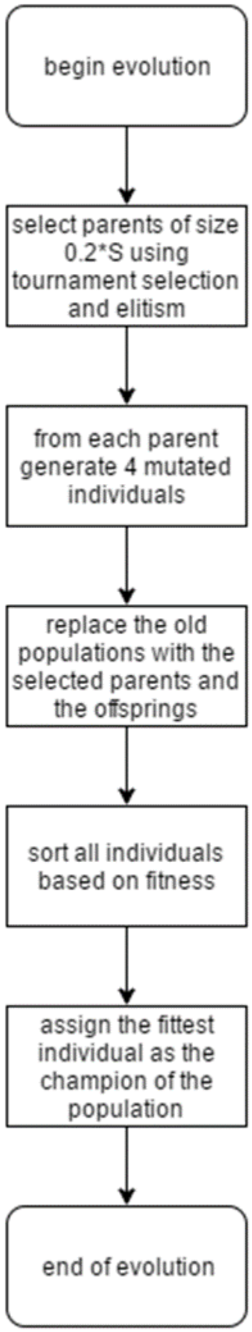

Fig. 4. Flowchart of algorithm representing the evolution of one population.

IV. Results and Analysis

After running the algorithm many times with different parameters, the results show the best coevolution environment under the following parameters:

- Mutation parameters:

  ADD: 0.2

  DELETE: 0.2

  MODIFY: 0.5

  NO MUTATION: 0.1

- Alpha: 0.4

  Alpha and evolution have a strong connection. Alpha effects what percentage the results of every round affects the fitness. After multiple tests, having alpha of 0.5 and lower gave a better result than alpha higher than 0.5. This result makes sense relative to our theory of cyclic motions; the fitness should reflect how well the individual is doing over multiple rounds in the past and not just the present play.

- Rounds before evolution occurs: 5

  With evolution after every round, the results showed as if the players are countering just the last move played and not a pattern of plays. A lot of draws occurred and a sequence of win for player 1 then immediate win for player 2.

  With 5 rounds before evolution happens the individuals are having more ‘experience’ towards their opponent.

A. *Average Fitness of Populations*

The best way to see coevolution happening is by analyzing the average fitness of both populations with time. When one population is doing better than the other, it is expected to see the other population evolving and becoming the better population after a short time and so on. Figure 5 displays the average fitness of both populations over time. A drop/rise in fitness occurs every time an evolution happens because the fitness of all new individuals is around 0.5.

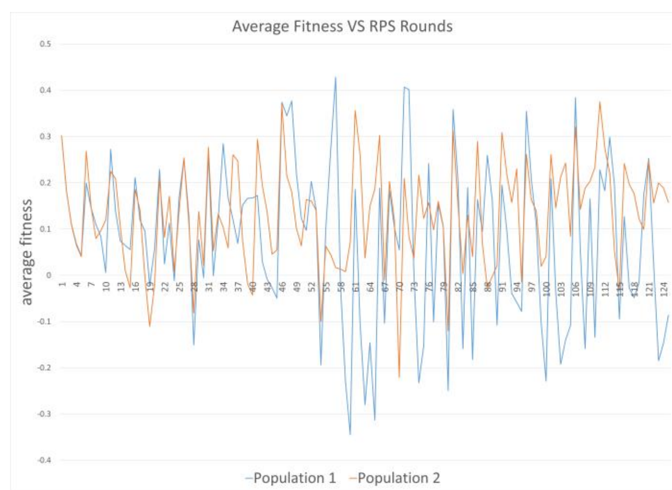

Fig. 5. Average fitness of both populations over 125 rounds of RPS. The size of each population is 50000 individual, with same parameters as stated above. The graph shows the result of a coevoultionary environmnet where every population keeps on countering the strategy of other population. The average fitness of a population shows how well the population is doing agianst the champion of the other population.

Figure 6 shows the score of the champions in the perspective of champion 1. The coevolution is also clear as with every winning period we expect a losing period after it.

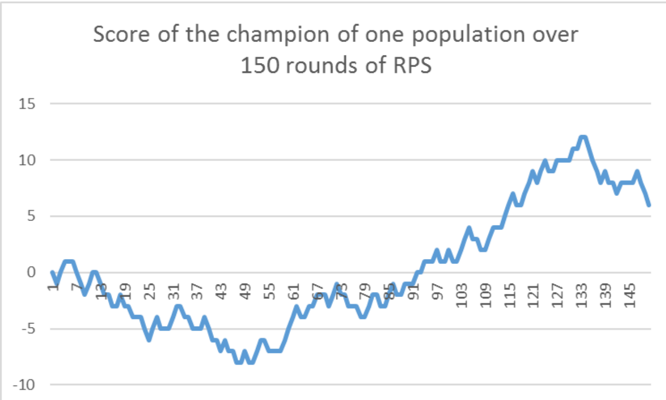

Fig. 6. Score of one champion over 150 rounds of RPS. If the champion wins score is incremented (increasing slope), if it loses score is decremented (decreasing slope), and if a draw occurs the score stays the same (straight horizontal line). The effect of coevolution is well seen as half the time the champion was losing more than winning and at the other half of time the champion was winning more than losing. Which means the population adapted to the strategies of the other population.

B. *Average Number of Rules*

The number of rules per individual in a population affects the result of population. However, after a while of playing both populations saturate at an average number of rules between 10 and 30. Average number of rules is calculated as the sum of all rules of individuals in a population divided by the number of individuals in the population. Figure 7 shows the average number of rules of both populations over 2500 rounds of RPS.

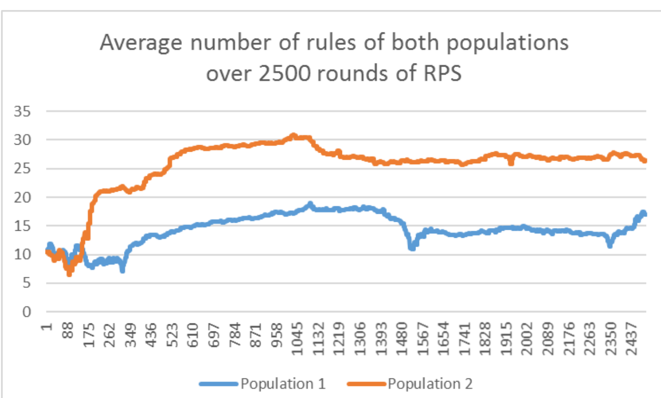

Fig. 7. Average number of rules of both populations over 2500 rounds of RPS.

C. *Average Score of Rules*

Another important result that shows the progress of coevolution is the accuracy of the rules of individuals. Using the score scale explained in previous sections of this paper, the average score of an individual is the sum of scores of all the rules divided by the number of rules. And the average score of the population is the sum of the average score of all individuals in the population divided by the number of individuals in the population. Figure 8 shows that as populations evolve their rules are getting more accurate. This is mainly because of the selection pressure in the parent selection where two individuals of similar fitness are compared based on their average rule score.

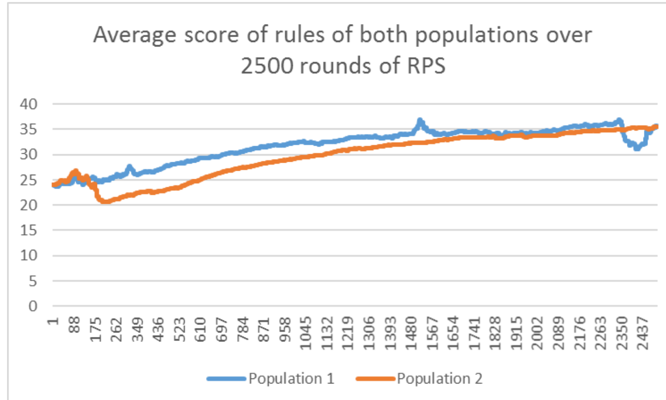

Fig. 8. Average rule score of both populations over 2500 rounds. The rules are getting more accurate as coevoulation happens. Both population are adapting to each other and advancing in their strategies to win.

V. Future Work

A. *Fitness Evaluation*

The fitness function used now gives a fitness around 0.5 for every new individual. This comes from the idea that a new born will have nothing to base his fitness on and its fitness is only updated by its future. However, another approach can be made by evaluating the new individual by how well it does base on the previous history, 10 rounds back in history per say. This method of evaluating fitness need to be tested and compared to the fitness method used now. By evaluating the fitness of every new individual, the drop/rise of average fitness of a population after every evolution will disappear.

B. *Higher Level of Intellegence*

A higher level of intelligence means instead of having a set of rules in an individual, every individual will have multiple states where every state contains a set of rules. The states will work as a state machine with transition conditions between states similar to the conditions used in rules. So for example, an individual can keep track of multiple strategies played by the opponent. If a specific pattern is detected from the history, the individual will switch to the state that satisfies that pattern and start playing with the set of rules inside that state.

C. *RPS EA VS. Humans*

So far all the testing is happening in a coevolution environment. After developing a mobile app where humans can play RPS against an evolving population, we can test how well an evolutionary algorithm can detect the 'cyclic motion' of the human. We believe with a higher level of intelligence the results will be promising.

VI. Conclusion

To conclude, this approach of evolving a zero-sum game players using coevolution has shown a lot of success. RPS is a simple game but has the same complexities as many problems and situations that we face in real life every day. This study will help us understand this type of problems and be able to analyze a solution strategy towards it.

References

[1] E. Picardo, "Zero-Sum game," Investopedia, 2003. [Online]. Available: http://www.investopedia.com/terms/z/zero-sumgame.asp. Accessed: May 1, 2016.

[2] Z. Wang, B. Xu, and H.-J. Zhou, "Title: Social cycling and conditional responses in the rock-paper-scissors game," Apr. 2014. [Online]. Available: https://arxiv.org/abs/1404.5199. Accessed: May 1, 2016.

[3] A. E. Eiben and J. E. Smith, Introduction to evolutionary computing, Second ed. Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2003.