Exploring the iterated Prisoner's Dilemma: Genetic Algorithm as a means of developing population stability and tournaments with noise

Ryan Vasios

Tufts University

17 Burnham St., Somerville, MA

ryan.vasios@gmail.com

Abstract

My program and experiments have focused on using the Genetic Algorithm as a means of exploring strategies within the iterated Prisoner's Dilemma. The Algorithm works by creating an initial strategy of random strategies that go on to compete and reproduce respective to their fitness. The highly evolved strategies were examined to inspect two particular qualities: stability, or the ability of a converged population to resist invasion, and the ability of evolved strategies to successfully play alongside Tit-For-Tat in an environment with noise.

1 Basis of the Genetic Algorithm Implementation:

1.1 Pseudocode of Genetic Algorithm:

```
-GENERATION = CREATFIRSTGENERATION()
-WHILE( GENERATION HAS NOT CONVERGED):
        CONDUCT_TOURNAMENT(GENERATION)
        COMPUTE_FITNESS(GENERATION)
        NEXT_GEN= REPRODUCE(GENERATION)
        GENERATION = NEXT_GEN
```

1.2 Representing Strategies as Genomes

This method was first proposed by Robert Axelrod in his collection of essays, 'The Complexity of Cooperation'. Essentially, a player's collection can be represented as a binary string, replacing the customary '0' and '1' with the characters 'C' and 'D' corresponding to the two strategies offered in the iterated Prisoner's Dilemma, Cooperation and Defection. Every index within this string corresponds to the strategy preferred by this player given a particular recent history in the game. In my implementation, player's are able to remember the last four moves of the game played by both itself and its opponent. Because these four games are represented in 8 characters (such as for example 'CC' 'DC' 'CD' 'DD') the portion of the genome holding strategy bits must be 256 characters long (two to the eighth power). Furthermore, players must have a first move predestined for play. This is stored in the last 8 bits of our genome string as a faux preliminary history. Therefore our entire genome is 264 characters long holding the possibility for 2^{264} distinct strategies!!!

1.2 Player Python class:

```
class player:
        def __init__(self):
                self.genome = makegenome()
        self.lastmove = self.genome[-firstmovestart:]
        self.score = 0
        self.tourneyscore = 0
        self.fitness = 0
        self.isnice = isitnice(self)
        self.numrep = 0
```

In addition to their strategy genome, players need access to their most recently played move, their score within their current game, their score within the entire generation's tournament, their fitness (likelihood for reproduction), and the number of reproductions they have made so far. This is so certain successful strategies don't have excessive influence in the process of reproducing future generations. I've also added the additional feature of seeing if a strategy is 'nice' in Axelrod's terminology. That is, if a player cooperates on the first move, and continues to cooperate until encountering a defection.

1.3 Fitness Function:

The purpose of the fitness function is to translate the relative performance of all player's into a likelihood to reproduce. The function I have employed was borrowed from a 2005 paper

by a Mr. Adnan Haider entitled “Using the Genetic Algorithm to Develop Strategies for the Prisoner's Dilemma”. It is computed as follows:

player.fitness = a * player.tourneyscore = b

'a' and 'b' are two coefficients computes as follows:

a = (c -1) * (avg_score) / (max_score – avg_score)

b = avg_score * (avg_score – (c * score_avg)) / (max_score – avg_score)

with max_score being the highest score obtained in a generation's tournament, avg_score being the mean score of all player's in a generation's tournament and 'c' is the number of times an individual is allowed to reproduce (2 in my implementation).

1.4 Means of reproduction

Having selected two strategies from the population the Genetic Algorithm proceeds to mate these two *parents* and produce their two *children*. This reproduction is a mirror of sexual reproduction in which the genetic material of the parents is combined to produce the children. In this research reproduction allows exploration of the search space and provides a means of reaching new and hopefully better strategies. Reproduction is accomplished using a simple yet effective genetic operator called crossover. Crossover is an artificial implementation of the exchange of genetic information that occurs in real-life reproduction. This method consists of breaking both the parent chromosomes at the same randomly chosen point and then rejoining the parts with their compliment from their mate.

1.4 Convergence of Algorithm

Quite simply, the algorithm converge when all players in a given generation produce the score with one another. In this case, the minimum, average, and maximum score are identical.

2 Testing the stability of Converged Populations:

2.1 Basis of first experiment

In my first experiment, evolving generations of players compete within themselves. Therefore the environment driving evolving strategies is a shifting plane, changing with the strategies of the players. In many instances, final converged populations usually obtain average scores at or near

600 points. Because individual games between players are represented in 200 rounds, this corresponds to the value of both players cooperating throughout the entire game. Its very impressive to see that a collection of random players can actually go on to develop a likelihood to cooperate with one another. However, in a handful of instances, final populations can have average scores below 400 and in at least one round that I conducted, all mutual defections with a score of 100.

It was clear that while most outcomes favored nice and cooperative strategies, other strategies would emerge in other cases because of their resilience to resist invasion. Following Robert Axelrod's ecological model of the Prisoner's Dilemma strategies, I wondered if converged strategies had a higher aptitude to be 'stable' irrespective of their final average scores. To review, the quality of being 'stable' is that within a population of identical strategies (or near identical strategies such as our final, converged generation of players), no outside strategy can obtain a better score with a native strategy than two native strategies can achieve with one another.

To test this, a player from our final strategy was paired against a series of random opponents to see how long it could withstand invasion, ie- encounter a strategy that obtained a better score with the native, converged player than it did with its peer.

2.2 Results of first experiment

Below are the results of 6 trials

Algorithm converged after 20 rounds

Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
CDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCC
CCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDC
CCCDDDCCCCDDDCCCCDDD
PLAYER2:
CDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCC
CCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDCCCCDDDC
CCCDDDCCCCDDDCCCCDDD
Final scores of 426 426
population is stable!! (most likely)
the first 5000 random opponents could not obtain better scores from interacting with population member than between pop members

Algorithm converged after 26 rounds
Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
DCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCD
CCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDC
CCCDCCCCDCCCCDCCCC
PLAYER2:
DCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCD
CCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDCCCCDC

CCCDCCCCDCCCCDCCCC
Final scores of 520 520
population is not particularly robust having been invaded by the 1 th random challenger

Algorithm converged after 25 rounds
Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
PLAYER2:
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
CDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDCCDCDC
Final scores of 440 440
population is stable!! (most likely)
the first 5000 random opponents could not obtain better scores from interacting with population member than between pop members

Algorithm converged after 27 rounds
Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
CCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCC
CDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCC
DDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCC
PLAYER2:
CCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCC
CDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCC
DDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCCCDDDCC
Final scores of 402 402
population is stable!! (most likely)
the first 5000 random opponents could not obtain better scores from interacting with population member than between pop members

Algorithm converged after 16 rounds
Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
PLAYER2:
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Final scores of 600 600
population is stable!! (most likely)
the first 5000 random opponents could not obtain better scores from interacting with population member than between pop members

Algorithm converged after 23 rounds
Below is an example of a typical game played between two players in our converge environment:
PLAYER1:
DCDCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
PLAYER2:
DCDCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Final scores of 596 596
population is not particularly robust having been invaded by the 12 th random challenger

2.3 Observations

Notice there is no correlation between the highness of a population's score and its stability. Of these six trials the two populations with the two lowest native scores, 402 and 460, are both stable and have each resisted invasion by 5000 native strategies. Compare this to the third highest scoring native strategy that is invade by the very first random strategy!!! It is worth pointing out though that the score with the best native score of 600, a consistent series of Cooperations, is stable, while the second highest scorer, at 596 native score, is invaded by the 12th random strategy.

Furthermore there appears to be no correspondence between the number of generations used to reach convergence and the stability of the final generation produced.

In all, of the six converged generations examined here, 4 are stable, suggesting that highly evolved strategies tend to be 'stable' by and large.

3 Testing the Genetic Algorithm in an environment with Noise:

3.1 Basis of second experiment

In “The Complexity of Cooperation”, Robert Axelrod devotes a chapter to discussing playing the Prisoner's Dilemma in environment's with noise. That is, these are games where players have their moves converted against their will unknown to their opponents. Its a very interesting and realistic feature of the game's model and application to real-world situations. Although Axelrod did not use the GA to explore elaborating strategies in environments with noise, I thought it would be an interesting experiment.

In my trials, generations of players compete against a version of TIT-FOR-TAT, the winner of Axelrod's computer tournaments. This simple strategy cooperates on the first move and then does whatever the other player does on the previous move in follow rounds. Note that the genome for TIT-FOR-TAT is a string of alternating C's and D's since the lowest order indexing bit is the last move of the opponent in the previous round.

Both evolving strategies and TIT-FOR-TAT have a one in six chance of having their moves “flipped” in every round. Consistent with Axelrod's research, I expected final strategies to demonstrate the qualities of “generosity” and “contrition”. That is to say, players are “generous” and assume the best intentions from opponents although they could have originated as Defections. Players are also expected to have increased “contrition”, which is to say, they will forgive some

Defections from their opponent, since they could have originated from a “flip” of the player's move, or is even a response to a “flipped” move on their part.

Because this population does not reach convergence I let the Genetic Algorithm run for 1000 generations for comparing the initial and final generations. Consistent with the qualities of “generosity” and “contrition”, I expect to share **more** common Cooperation genes with TIT-FOR-TAT and **less** common Defection genes with TIT-FOR-TAT by the final round.

3.2 Outcome of second experiment

Six random trials of my program produced these outcomes

first generation shared 0.484375 percent Cooperations with TFT
final generation shared 0.5546875 percent Cooperations with TFT
first generation shared 0.4765625 percent Defections with TFT
final generation shared 0.3984375 percent Defections with TFT

first generation shared 0.484375 percent Cooperations with TFT
final generation shared 0.5625 percent Cooperations with TFT
first generation shared 0.5 percent Defections with TFT
final generation shared 0.4765625 percent Defections with TFT

first generation shared 0.484375 percent Cooperations with TFT
final generation shared 0.5 percent Cooperations with TFT
first generation shared 0.484375 percent Defections with TFT
final generation shared 0.4921875 percent Defections with TFT

first generation shared 0.484375 percent Cooperations with TFT
final generation shared 0.4609375 percent Cooperations with TFT
first generation shared 0.5 percent Defections with TFT
final generation shared 0.5234375 percent Defections with TFT

first generation shared 0.5 percent Cooperations with TFT
final generation shared 0.4921875 percent Cooperations with TFT
first generation shared 0.484375 percent Defections with TFT
final generation shared 0.5078125 percent Defections with TFT

first generation shared 0.4921875 percent Cooperations with TFT
final generation shared 0.515625 percent Cooperations with TFT
first generation shared 0.484375 percent Defections with TFT
final generation shared 0.3671875 percent Defections with TFT

3.3 Observations

The first two trials showed exactly the kind of behavior I anticipated. Final generations in shared more of the Contrition of TIT-FOR-TAT and less Defections for the sake of assuming the best of an opponent and forgiving more. However, further trials showed this was not a consistent trend. The fifth trial, for instance, shows my expectations being reversed. It is consoling to see that these deviances are in a smaller percentage range than those first encouraging runs.

Cooperations are more likely to increase over time. Defections are more fickle and commonly develop in either direction. The sixth trial however did demonstrate a very impressive decrease in defections.

References

Axelrod, Robert The Evolution of Cooperation. Basic Books. 1984

Axelrod, Robert The Complexity of Cooperation. Princeton University Press. 1997

Haider, Adnan Using Genetic Algorithms to Develop Strategies for the Prisoner's Dilemma. Department of Economics, Pakistan Institute of Development Economics. Nov 2005