To Cooperate or To Defect: That's the Prisoner's Dilemma

Yüce Tekol
Eastern Mediterranean University
Computer Engineering Department
Gazimagusa, TRNC via Mersin 10, TURKEY
ytekol@ieee.org

Abstract

Prisoner's Dilemma is the most studied game among evolutionary computation researchers. The game is extremely simple to play: each player has only two choices, either cooperation or defection. Nevertheless, this simplicity is just deceiving; the game has very interesting consequences that effected economics, biology and evolutionary computation. In this paper, we take a small trip together to explore the most popular variant of the game: the iterated prisoner's dilemma, following the now classical approach to ”coevolve” strategies for it by using a genetic algorithm.

1 Introduction

Since its development in 1960's, prisoner's dilemma (PD) attracted attentions of many scientists from several diverse fields such as economics, biology, game theory, computer science and political science. The game is used to model the emergence of cooperative behavior [10] in a population of selfish individuals. Dawkins [3] and Dugatkin [4] give some interesting examples about this phenomenon in biology. Applications of the game in economics can be found in [8], and political science in [13].

Since Axelrod's seminal paper [1], the dominant method to study PD has been genetic algorithms (GAs), however other methods were also used; such as Evolutionary Programming [5] and Ant Colony Optimization [12]. In this paper, we will explain how to evolve strategies for PD using GAs. This paper is organized as follows: In Sections 2 and 3, I explain PD and GAs respectively in some detail; Section 4 contains the experiment setup and results. Conclusions and future directions are given in Section 4.

1.1 Prisoner's Dilemma

Prisoner's dilemma is a *non–cooperative*, *non–zerosum* game, played between two players, where non–cooperative denotes that players have no communication prior to the game and they don't know each other's decision until they both decided what to do, and non–zerosum denotes that one player's win does not necessarily mean other player's loss.

The story of PD is as follows: Joe and Bill are arrested by the police after a bank robbery and put in seperate cells to be questioned. The investigators offer them both the same deal. If one of them confesses and the other one keeps silent, the former will be released while the latter will be sentenced to many years in prison. If both confess, their sentence will be less. The prisoners also know that the investigators don't have much evidence, so if they both keep quiet, their penalty will not be very serious. Knowing these facts, what should the prisoners do? Trust each other and cooperate by keeping quiet or confess the crime and defect to the other one? Thus, each player has two choices, either cooperation (C) or defection (D), and the payoff they get for their choices are calculated according to Table 1. In this table, each cell holds two values; left one is the row player's payoff, right one is the column player's payoff. The letters, R, S, T, P denote, *reward* payoff for mutual cooperation by keeping quiet, *sucker's* payoff to cooperate against a defecting opponent, *temptation* payoff to defect against a cooperating opponent and *punishment* payoff for mutual defection, respectively. For instance, if row player cooperates and column player defects, then the former will get sucker's payoff and the latter will get temptation payoff.

		Column Player	
		Cooperate	**Defect**
Row	**Cooperate**	R, R	S, T
Player	**Defect**	T, S	P, P

Table 1: Payoff matrix for prisoner's dilemma.

The PD is defined by the following inequalities on the values of S, P, R and T.

$$T > R > P > S \quad (1)$$

$$2R > S + T \quad (2)$$

		Column Player	
		Cooperate	**Defect**
Row	**Cooperate**	3,3	0,5
Player	**Defect**	5,0	1,1

Table 2: Payoff matrix used throughout the paper.

An instance of the table with appropriate values is shown in Table 2. According to the tables 1 and 2, any rational player will choose defection no matter what the other player chooses ($5 > 3$ and $1 > 0$), so both players will get 1 point. If they cooperated though, they would get 3 points each, which is better than 1 point they have. Unfortunately, in a one-shot game, the players can not resist to the temptation to defect, so mutual cooperation never arises.

To overcome this problem, Axelrod [2, 1] used a variant of the game—iterated prisoner's dilemma—which, as its name implies, repeats the conventional game several times with the number of repetitions unknown to both players. Extending the game this way gives players the chance to build trust, retaliate defection, so the hope of cooperation.

Using Axelrod's representation, each chromosome in the population carries the starting moves plus the cells of a lookup table which encodes all the possible moves of two players which can remember M previous games. For $M = 1$, the lookup table would be as in table 3.

The first letter in left column shows the player's self–move, and the other letter shows the opponent's move; so the table is read as, "if I *cooperated* and my opponent *cooperated* in the previous game then my action will be X for this game", and "if I *defected* but the other player *cooperated* then I will do Z", etc.

CC	X
CD	Y
DC	Z
DD	W

Table 3: Lookup table for $M = 1$. X, Y, Z, W are either C or D.

In order to have a compact representation, cells of the right column of the lookup table are written in a string from top–to–bottom; becoming $XYZW$. The players need to decide what to do in the first game, so a presumed game is also appended at the beginning of the string. Tit–For–Tat —which is the most famous strategy of prisoner's dilemma—begins the game with a cooperation and does whatever its opponent does in subsequent games. Using the method explained above, it is represented as CCCDCD. If we use 0 for cooperation and 1 for defection, the string now becomes 000101. This binary representation is well-suited to GAs, so virtually all the research done on evolutionary IPD employed a GA with binary strings. It should also be noted that, for a complete PD strategy with memory size M, we need 2^{2M} bits for the lookup table and $2M$ bits for the presumed game, totally:

$$2^{2M} + 2M\text{bits}. \tag{3}$$

Most of the work done on evolving strategies for IPD, also this paper, uses $M = 3$ strategies, owing to the fact that, the search space for this memory size is huge enough (2^{70}) to be intractable by exact search algorithms, yet it is small enough for practical research. Because of its significance, we show a little portion of the $M = 3$ lookup table in Table 4. In the table, odd bits are the player's self prior moves, and even bits are the opponent's, sorted in order of time (i.e., the move in first bit is older than the one in third). For example, according to Table 4, the entry CDDCCC is read as "if I *cooperated* but my opponent *defected* three games before, and I *defected* but my opponent *cooperated* two games before, and both I and my opponent *cooperated* in the last game then do Y".

CCCCCC	X
$\vdots$	$\vdots$
CDDCCC	Y
$\vdots$	$\vdots$
DCDDCC	Z
$\vdots$	$\vdots$
DDDDDD	W

Table 4: Part of the lookup table for $M = 3$. X, Y, Z, W are either C or D.

2 Genetic Algorithms

Genetic Algorithms (GAs) are a class of search methods inspired by *Darwin's Theory of Evolution*. GAs were first implemented by John Holland [9] and later popularized by David Goldberg [7]. GAs operate on a population of candidate solutions, called *individuals* or *chromosomes*. Conventional GAs allowed only binary encoding of chromosomes; but today, real numbers, permutations, and even parse trees are also used. There are three fundamental operators of GAs:

1. Selection (reproduction) operators: A selection operator chooses chromosomes from the population to be processed by crossover, by taking account their fitness. This way, fitter chromosomes pass to the next generation with a higher probability.
2. Crossover (recombination) operators: As its name implies, a crossover operator forms new chromosomes by combining (generally) two chromosomes with a (usually) pre–determined *crossover probability*, p_c. p_c depends on the problem and other parameters, but it is often taken about $70 - 80\%$. Crossover is the main search operator of GAs.
3. Mutation operators: They slightly change a gene of a chromosome with a *mutation probability* p_m. p_m is generally taken as small (e.g., $p_m \leq 1\%$. Mutation is the secondary search operator of GAs.

Another important concept in GAs is the presence of an *objective function* which converts the genotype of a chromosome to phenotype and calculates its *fitness*. Usually, the objective function is explicit; that is, it evaluates every individual in the population and returns their fitness; but in our case, the fitness of an individual depends on how it performs against other members of the population, so there is no explicit objective function. This is called *coevolution*.

```
procedure GA__IPD__Run
    Initialize__Population($P_{old}$)
    while termination criteria not satisfied do
        for each chromosome $c_i$ in $P_{old}$ do
            Evaluate($c_i$,$P_{old}$)
        end
        Generate__New__Population($P_{new}$,$P_{old}$)
        $P_{old} \leftarrow P_{new}$
    end
end
```

Figure 1: Generic genetic algorithm.

Figure 1 shows the algorithm of a conventional GA. In the figure, `Initialize__Population(`P_{old}`)` function merely fills the chromosomes of population P_{old} with 0s and 1s, randomly. `Evaluate(`c_i`,`P_{old}`)` function runs chromosome c_i against every member of population P_{old} including itself to compute its fitness. `Generate__New__Population(`P_{new}`,`P_{old}`)` function, as its name says, generates population P_{new} using P_{old}. The function is given in algorithm form in Figure 2.

```
procedure Generate_New_Population($P_{new}$,$P_{old}$)
    $P_{new} \leftarrow \emptyset$
    while Size($P_{new}$)<Size($P_{old}$) do
        // Selection
        $c_1 \leftarrow$Select($P_{old}$)
        $c_2 \leftarrow$ Select($P_{old}$)
        // Crossover
        if $p_c < r(\cdot)$ then
            Crossover($c_1$,$c_2$)
        end
        // Mutation
        for $i = 1$ to chromosome_length do
            if $r(\cdot) < p_m$ then
                $c_{1_i} \leftarrow \neg c_{1_i}$
            end
            if $r(\cdot) < p_m$ then
                $c_{2_i} \leftarrow \neg c_{2_i}$
            end
        end
        $P_{new} \leftarrow P_{new} \triangleleft c_1 \triangleleft c_2$
    end
end
```

Figure 2: Algorithm for generating a new population.

In Figure 2, `Size(`P_{new}`)` returns the number of individuals in P_{new}. `Select(`P_{old}`)` is a function that implements *tournament selection* of tournament size $q = 4$. Tournament selection takes q random individuals from the population and returns the fittest one. $r(\cdot)$ is a function that returns a random number in the interval $[0, 1)$. `Crossover(`c_1`,`c_2`)` function implements *uniform crossover* as explained in the next paragraph. $\triangleleft$ operator inserts the chromosome on the right–handside to the population on the left–handside. c_1, c_{1_i}, p_c and p_m are, 1st chromosome, ith bit of 1st chromosome, crossover probability and mutation probability respectively.

Uniform crossover recombines two chromosomes by swapping each bit at the corresponding positions, with a fixed probability (usually 0.5%). An example recombination is seen on Figure 3.

3 The Experiment

In order to see the effects of iteration on prisoner's dilemma, we made an experiment similar to Axelrod's. The employed GA had following parameters: the population size was $m = 40$, crossover probability and mutation probabilty was $p_c = 70\%$ and $p_m = 0.1\%$ respectively. Each chromosome played $g = 150$ games with each of the other chromosomes and itself, and they remembered past $M = 3$ moves. The GA was allowed to iterate 1000 times. Assuming, $\sigma(c_k, c_j, i)$ is a function that returns c_k's score against c_j at ith game, the fitness $\phi(c_k)$ for each chromosome c_k was average per-step payoff, calculated by:

Before crossover

	1	2	3	4	5	6
chromosome 1	**1**	1	**0**	**0**	0	1
chromosome 2	**0**	1	**1**	**0**	1	1

After crossover

	1	2	3	4	5	6
chromosome 1	**0**	1	**1**	**0**	0	1
chromosome 2	**1**	1	**0**	**0**	1	1

Figure 3: A uniform crossover example. Bold bits indicate the swapping positions.

$$\phi(c_k) = \sum_{j=1}^{m} \sum_{i=1}^{g} \sigma(c_k, c_j, i) \tag{4}$$

Now, consider Figure 4 which shows one of the experiment runs. Previously, we mentioned that, for a one–shot game the players will always choose defection; but as we observe from this figure, on the average chromosomes cooperated most of the time (which is indicated by average per-step payoff=3). It is obvious from Table 2 that, why the players always choose defection for the one-shot game; but it is not so obvious at first sight why they cooperate when the game is repeatedly played among a population of players. First of all, we see an initial increase of payoff of the best strategy, while a decline of the average payoff of the population in the figure. This is because defectiveness paid, and cooperative strategies are exploited by some non–cooperators. Until about sixth generation, more and more chromosomes are chosen to be defective and average payoff continues to decrease. After sixth generation though, strategies which can both retaliate defection and reward cooperation start to emerge; finally after eleventh generation, the population is filled with these strategies and mutual cooperation is achieved.

4 Conclusions and Directions

In this paper, we explained evolution of strategies for prisoner's dilemma using genetic algorithms. It is observed that, the payoffs of the chromosomes decreased at the beginning of the experiments, but later stabilized after the appearance of retaliative strategies, confirming Axelrod's results.

PD has many variants besides IPD, such as free–rider (or lift) problem, which is generated by eliminating Equation 2, where players take turns to receive sucker's and temptation payoffs [11, 8]. Another important variant is N–player IPD, in which, payoff of a player depends on more than one player [14]. In the spatial PD, players play the game in the one–shot fashion, but they are located on a grid playing only with their closest neighbours [6]. O'Riordan gives a detailed review of some PD variants in [11].

References

[1] R. Axelrod. The Evolution of Strategies in the Iterated Prisoner's Dilemma. In L. Davis, editor, *Genetic Algorithms in Simulated Annealing*, pages 32–41. Pitman, London, 1987.

[2] R. Axelrod and W. D. Hamilton. The Evolution of Cooperation. *Science*, 211:1390–1396, March 1981.

[3] R. Dawkins. *The Selfish Gene - 2nd ed.* Oxford University Press, Oxford, 1989.

[4] L. A. Dugatkin. Animal Cooperation Among Unrelated Individuals. *Naturwissenschaften*, 89:533–541, 2002.

[5] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, New York, 1995.

[6] M. R. Frean and E. R. Abraham. A Voter Model of the Spatial Prisoner's Dilemma. *IEEE Transactions on Evolutionary Computation*, 5(2):117–121, 2001.

[7] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, 1989.

[8] S. P. H. Heap and Y. Varoufakis. *Game Theory : A Critical Introduction*. Routledge, New York, 1995.

[9] J. H. Holland. *Adaptation In Natural Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[10] K. Lindgren and M. G. Nordahl. Cooperation and community structure in artificial ecosystems. In C. G. Langton, editor, *Artificial Life: An Overview*. The MIT Press, 1997.

[11] C. O'Riordan. Iterated Prisoner's Dilemma: A Review. Technical Report NUIG-IT-260601, National University of Ireland, Galway, Department of Information Technology.

[12] Y. Tekol and A. Acan. Ants Can Play Prisoner's Dilemma. Congress on Evolutionary Computation (CEC) 2003, Canberra, Australia. (To appear), 2003.

[13] A. M. van der Veen. The Evolution of Cooperation In Society: Transforming The Prisoners Dilemma. 2000 Annual Meeting of the American Political Science Association, Washington, 2000.

[14] X. Yao and P. J. Darwen. An Experimental Study of N-Person Iterated Prisoner's Dilemma Games. *Informatica*, 18:435–450, 1994.

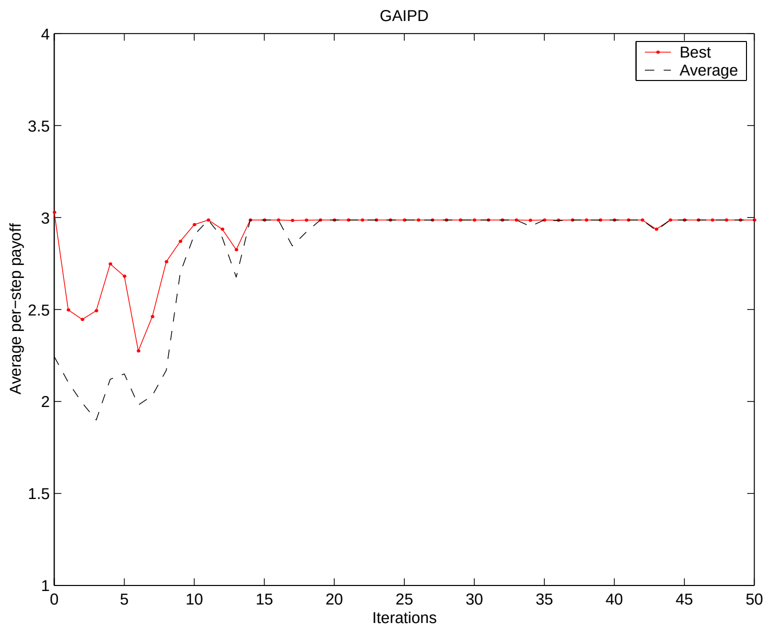

Figure 4: $M = 3$ GAIPD run. The run is 1000 generations long, but shown here partially.