The Best Evolutionary Solution to the Iterated Prisoner’s Dilemma

Angel Kuri Morales
Instituto Tecnológico Autónomo de México
Río Hondo No. 1
México 01000, D.F.
akuri@itam.mx

Abstract. In this paper we discuss the methodology and program which won the contest of the Iterated Prisoner’s Dilemma (IPD) which was held during the Congress on Evolutionary Computation 2000 (CEC2000) in San Diego, California in July of 2000. This results were recognized as the best during the congress but have not been published before. In the first part of the paper we make an introduction to the IPD problem. In the second part we describe the algorithm we used to tackle the problem. This algorithm is based on a co-evolutionary Genetic Algorithm (GA). In the third part we present our conclusions and possible lines of future research.

Keywords. Cooperation, dilemma, genetic algorithms, co-evolution, CEC2002.

1 Introduction

In this paper we address a classical problem from game theory which sheds new light on several problems which have been discussed by philosophers and politicians [1], [2] throughout history. The interest it has generated allows to even propose ethical problems which may be found in the Web [3], [4]. This problem refers to a situation in which we are to decide which is the rational option of an individual as part of a group and for the group in its entirety. It helps us understand how such dilemmas may be solved to obtain the greatest individual and collective benefit and its implications reach far beyond a mere game: under the light of the IPD, it has been possible to analyze the problem of the arms race [5], the adequate selection of providers of goods and services in an open economy [6] and the policies of funding for science and technology [7], among others.

1.1 The iterated prisoner’s dilemma

The iterated prisoner’s dilemma takes its name from the following hypothetical situation:

“In a cell the police keeps two political prisoners. The interrogator is trying to convince them to confess their liaison with an illegal opposition party. The prisoners know that if neither confesses, the investigator will not be able to press charges but he may continue his interrogation for three months without setting them free. On the other hand, if both confess their help will be considered and they will only be in jail for five months. The prisoners are questioned in isolation; they do not know whether the other one has confessed but both know about the deal that is being offered. The dilemma is: What is the best strategy? To confess (defect) or not to confess (cooperate)?”

This dilemma (PD) may be thought of as a “game” in which players are graded according to the following table. Depending on the mutual responses, each player will receive a number of points. In the case just described, the grade reflects the losses arising from any given answer, as shown in Table 1. in this case the objective is to minimize the losses. Alternatively, the problem may be defined in terms of a benefit in which case we would try to maximize it.

This problem is called the “Iterated Prisoner’s Dilemma” if the process is repeated several times. The true interest of this problem lies, precisely, in iterating

<table>
<thead>
<tr>
<th>Play Player 1</th>
<th>Play Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>Defect</td>
</tr>
<tr>
<td>Cooperate</td>
<td>Cooperate</td>
</tr>
<tr>
<td>Defect</td>
<td>Defect</td>
</tr>
<tr>
<td>Defect</td>
<td>Cooperate</td>
</tr>
</tbody>
</table>

Table 1. Grading Table for the PD
the actions as described. When this happens it is that the players may learn to adjust their behavior depending on the behavior of the other player. The points of each player are the sum of those he obtained in each play. Thus, a game between two players may be as follows:

<table>
<thead>
<tr>
<th>Plays</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Player 2</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Table 2. An Example of a Sequence of IPD

In the iterated version we wish to find the strategy which minimizes the damage (or maximizes the profit) given that we remember the last \( n \) plays. In the example of table 2 player 1 receives a damage of 35 whereas player 2 only receives a damage of 27: player 2 has won.

The minimax solution given by game theory looks to minimize the maximum damage an opponent may inflict. This is determined by comparing the maximum damage under cooperation against the maximum damage under defection. If the first player cooperates (C,-) the greatest damage is when the second player defects (CD) yielding a damage of 8 for the first player. If the first player defects (D,-), the greatest damage occurs, again, when player 2 defects (DD). Now the damage to player 1 is 5. Therefore, the first player minimizes his losses by defecting always. This line of reasoning is symmetric so that (DD) is the best minimax solution. It is easy to see, however, that the best strategy is the one in which both players cooperate. For example, in a sequence of length 4 (4 iterations) minimax strategy indicates that the best strategy would be DD;DD;DD;DD. The loss for player 1 (and for player 2 as well) is 5+5+5+5=20. But, clearly, strategy CC;CC;CC;CC induces a loss for player 1 of 3+3+3+3=12; much better than minimax’s.

It is more common to set the cost table of the IPD as one of gains rather than losses. In such case, it is possible to generalize the game with a table of variable values which, to preserve the spirit of the game, ought to comply with the constraints shown in table 3. This constraints are identified with the following first letters: C (cooperate); L (low); H (high) and D (defect).

<table>
<thead>
<tr>
<th>Player’s move</th>
<th>Opponent’s move</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>L</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>H</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

\[ L < D < C < H; \quad H + L \leq 2C \]

Table 3. Grading table for the IPD with variable values

For instance, the values C=3, L=0, H=5 and D=1 indicate that player 1 will win 3 points if both players cooperate (CC), 0 points if only he does (CA), 5 points if he defects and player 2 cooperates (AC) and 1 point if both defect. Here, obviously, 0<1<3<5 and 5+0<6, fulfilling the constraints. For this table the minimax strategy of permanent defection translates into constant gains of 1 point. In the example (4 iterations) each player wins 4 points. With a strategy of constant cooperation, on the other hand, would yield 12 points for each player. Several variations to this problem have been studied [3]. For example, when there are several moves per player, a non-symmetric grading in the table, multiple players, etc.

1.2 Strategies

We call a “strategy” to a set of one or more rules which tell us how to play the IPD. Some strategies are “Always cooperate” (AC) or “Always Defect” (AD). One common and simple strategy is called “Tit-for-Tat” (TT). In it the player starts cooperating; thereafter he or she repeats the opponents last move. Surprisingly, TT shows to be very efficient. If we play games with 5 iterations using TT, AD and AC with the values C=3, L=0, H=5 and D=1, we would get results as in table 4.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Action/Points</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>C 0</td>
<td>D 1</td>
</tr>
<tr>
<td>AD</td>
<td>D 5</td>
<td>D 1</td>
</tr>
<tr>
<td>TT</td>
<td>C 3</td>
<td>C 3</td>
</tr>
<tr>
<td>AC</td>
<td>C 3</td>
<td>C 3</td>
</tr>
<tr>
<td>AD</td>
<td>D 5</td>
<td>D 5</td>
</tr>
<tr>
<td>AC</td>
<td>C 0</td>
<td>C 0</td>
</tr>
</tbody>
</table>

Table 4. Confrontation of Strategies TT, AD and AC.
Another strategy is Pavlov’s (PS) and is shown in figure 1 as a two-state automaton.

![Diagram of Pavlov's Strategy](image)

Figure 1. Pavlov’s Strategy.

This strategy calls for “C” as long as the previous plays (both from the player and his/her opponent) are alike. A 5 iteration game, with grading table 3, where PS and AD face each other would be as follows:

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Action/Points</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>D 1 D 5 D 1 D 5 D 1</td>
<td>13</td>
</tr>
<tr>
<td>PS</td>
<td>D 1 C 0 D 1 C 0 D 1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5. Results of Confronting Strategies AD and PS

There are many strategies [8] and software with which one may build original strategies and play tournaments between them. For instance, the program WINPRI may be gotten from [8].

1.3 Design of a Strategy

A strategy may depend on $N$ previous plays for integer $N$ and $N \geq 0$. Thus, AD and AC do not depend on the previous moves; TT depends only on the last response and PS depends both on the last response as well as on the latest (own) play.

A form of representing strategies is with a vector [9] where every position represents the answer that should be given in each case. For instance, if during the last 2 plays the sequence DCDC has been recorded, the response will be derived from the corresponding positions of the vector. Therefore, we need to assign to every response a number and the simplest form is in binary. Hence, if the two last plays have been DCDC we change the Cs into 1s and the Ds into 0s, getting the string 1010 (which corresponds to number 12 assuming traditional weighted binary encoding). We should, then, answer with the element of the vector whose index is 12. In the case of PS we may represent the strategy as in table 6.

With this notation we may see that the answer of PS is CDDC. In the case of TT the answer would be DDCC. if it is deemed necessary to use strategies which depend on more (older) previous plays we may simply repeat the sequence. That is, the response for AD corresponds to DDDDD; it does not depend on any previous play. Therefore, the length of the string which represents a strategy depends on how many previous plays ($l$) it takes into consideration and is given by $2^l$.

<table>
<thead>
<tr>
<th>Previous Plays</th>
<th>Player/Opponent</th>
<th>State</th>
<th>Vector’s Index</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>AD</td>
<td>2</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>DC</td>
<td>PS</td>
<td>2</td>
<td>2</td>
<td>D</td>
</tr>
<tr>
<td>CD</td>
<td>PS</td>
<td>1</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>CC</td>
<td>PS</td>
<td>1</td>
<td>4</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 6. Determining the Answers from a Vector

2 Solving the IPD using a Genetic Algorithm

Given the representation above, it is possible to use a Genetic Algorithm (GA) to solve the problem and find a strategy which solves the IPD [10]. The basic algorithm used to win in CEC200 is described in what follows.

1) Generate a set $S$ of $m$ strategies at random.
2) From $i \leftarrow 1$ to $6 \times 6m$ make [from $a)$ to $c)$]:
   a) Select 2 strategies $s_1$ and $s_2$ from $S$ randomly.
   b) Confront $s_1$ and $s_2$ a predefined number of times.
   c) Register the results for $s_1$ and delete $s_2$.
3) From the $S$ strategies select the best $t$ (those whose average behavior has been best). We then take this set of strategies ($T$) as a base against which we must test the individuals of a GA (in the best T we conventionally include TT).
4) Generate a random initial population of $n$ candidate strategies ($C$)
5) While stopping criterion has not been met do
   a) From $i \leftarrow 1$ to $n$.
      i) Select strategy $c_i$ of $C$.
      ii) From $j \leftarrow 1$ to $t$
         (1) Confront $c_i$ against $s_j$
         (2) Register the sum of the results as the fitness of population $C$.
   b) From the $n$ individuals in the population plus the $t$ individuals in the bases set, select the best $t$.
      These will be the basis set for the next generation.
   c) Obtain a new $C$ using an EGA.
6) The best strategy is the best from set $T$. 


This is a co-evolutionary algorithm because the set of best strategies (T) evolves at the same time as the set of strategies (C) which evolve with the GA. The GA we used is not conventional (we have called it an eclectic GA, or EGA). It includes full elitism, deterministic pairing and annular crossover. It also self-adapts the probability of mutation, the probability of crossover and the number of descendants. Finally, it is poly-algorithmic in that it alternates with a random mutation hill-climber. The detailed description of the EGA may be found in [11], [12].

The EGA has been used to solve the IPD and the corresponding software may be gotten from [10]. this software only needs the values of the grading table (which we denoted by L, D, C and H) and the number of plays (g) which the strategy is to “remember”. The output of the program is a string of size $2^{2g}$ which encodes the resulting strategy.

2.1 International Contest CEC2000

The Congress on Evolutionary Computation held in San Diego, California, was hosted by the mathematics department of Iowa State University. It took place on July 16-19, 2000. As part of this congress four contests were presented to the international community:

1) *Reason vs Evolution: Prisoner’s Dilemma Competition*
2) *Time series prediction competition*
3) *Dow Jones Prediction Competition*
4) *Visualization Competition*

The rules for each contest were specified in [13] and, in general, sought for the applications of evolutionary techniques to each of the listed problems. We describe the particular conditions for the IPD.

1. The code containing the set of strategies may be published in the WWW after the contest. It should be contained in a flat ASCII file and written in C++ with:

   Definition of types/structures/global variables
   
   int firstact() { code which initializes the player and returns the first play }
   int pla (int lastact) { Code which processes the opponent’s play “lastact” and returns the present play }

2. Must use “0” to denote “C” and “1” for “D”.

3. Strategies which require excessive memory or time will be disqualified.

4. All submitted strategies plus some previously designed will form a set. This set will be complemented with a C++ interface which may invoke any of the strategies.

5. Strategies will be added so that their number will be close to a multiple of ten to obtain a balance between evolutionary and non-evolutionary techniques.

6. 100 iterations will be performed following the next procedure:

   a) The full set will be split in groups of 10.
   b) Random values for $L, D, C, H$ will be selected in a range between 0 and 10.
   c) $N$ rounds will be played; $N$ will be normally distributed with mean 300 and variance 25.
   d) 10 tournaments will be played in each group. Every pair of players will play $N$ round of IPD.
   e) The player with the highest score in every group of 10 will receive 10 points, the next one 9, and so on.
   f) The winner will be determined by the total points scored.

A program which will be described in what follows was sent to the mentioned tournament. This program, as pointed out in the abstract, turned out to be the winner of the tournament.

2.1.1 Winning Program of the International IPD Contest in CEC2000

The strategy which we sent to this contest was found with the co-evolutionary algorithm described above. However, since the points assigned in each individual contest were, according to the tournament rules, variable we determined the following.

1. We used a memory of 4 plays. That is, the EGA considered the 8 last plies (1 ply is equal to one player’s move) to determine its strategy. The search space, hence, consists of

   $$2^{2^8} = 2^{256} = (2^{10})^{25} = 2^{6225} \approx 64 \times 10^{25}$$

   possible solutions. It is a tribute to the analytical capabilities of the EGA that it was able to find a good solution in reasonable time.

2. Since the program receives no information as to the values of $L, D, C$ and $H$ we generated all possible combinations of integer positive values which satisfied...
the problem’s conditions (250 in all) and selected 10 of these combinations at random (see ahead).

3. We evolved (using the co-evolutionary GA) the best solutions for each one of the parameter sets.

2.1.2 Considerations

The choices mentioned in the last section obey the following considerations:

1. We selected a memory of 4 plays \((m=8; 8\) plies) because previous experiences [14] had found satisfactory behaviors for \(m=6\). Evidently, we sought to improve on this record. On the other hand, the evolution time for this value are still practical.

2. We selected 10 combinations because an analysis of cluster determination using self-organizing maps (Kohonen’s neural networks) indicated that the groups of parameters were sufficiently characterized considering 10 elements.

3. We assumed that, given the bases of the contest, it was reasonable to expect that the worst performances of each of the 10 opposing strategies, on the average, would be below ours, which were co-evolved.

Thus, the program we sent chose at random from one of the 10 strings of length \(2^5=256\) and the first 4 plays (when we still could not use the strategy for lack of information) were tackled with TT. Its appearance was deceptively simple and its workings practically unintelligible.

Next we show the 10 strings sent to the contest. The values which appear as a commentary /// correspond to the values of \(L, D, C, H\). Notice that each string is 256 bits long. This is because, since \(m = 8\), the set of \textit{histories} (sequences of foregoing plies of the opponent and ones own) is \(2^8 = 256\). For the parameters \((0-7-6-4)\), for instance, the historic sequence DDDDDDC (index = 3) triggers a response, on our part, of defection (0 or D); but the historic sequence DDDDCDD (index = 4) triggers our response “cooperate” (\(l \circ C\). On the other hand, and as a last example, we mention that for the parameters (1-9-7-5) the same sequences (indices = 3 and 4) trigger the responses C and C.
3. Conclusions and Future Work

3.1 Implications
The fact that the international community was invoked to tackle the IPD using evolutionary techniques implies:
   a) This problem’s importance transcends the merely formal.
   b) Although it is cast as a game, its implications hold interest outside game theory.
   c) Evolutionary tools have succeeded where alternative techniques of analysis and heuristic search have failed
   d) The international community which shows interest in artificial intelligence and its applications is ready to assimilate the importance of this kind of problems and tackle successfully the problems inherent to the purported solution of the IPD with GAs.

3.2 Future Lines of Research
To avoid the random selection of the possible combinations of $L, C, D, H$ it is possible to change the program that finds the strings [10]. Instead of adding the points which are gotten for each play we could keep a record of how many times we got an L or a D or some other value. Thus, the result of a contest between two strategies would not be a number but, rather, a linear combination of the values of $L, C, D, H$. Then using the 250 combinations of possible values we would know which of the two strategies would win for any value of the parameters. Afterwards we should perform a statistical analysis which compares the behavior of the string resulting from this new method against the program that won the contest to measure its performance.

3.3 Acknowledgements
It is interesting to mention that we have applied new concepts to solve this problem (non-conventional GAs; sampling selection of strategies) but relying on previous experiences and developments (the evolutionary methods themselves; the concept of co-evolution). Because of this we wish to acknowledge to those researchers who have preceded in the search for the best IPD algorithm: to John Holland for his initial work in the area of GAs, to Robert Axelrod for his initial motivation, to Douglas Hofstadter and others who have sensitized the community about the implications of the IPD beyond the restricted scope of game theory, to Dan Ashlock for having proposed and supervised this international contest.

References
The iterated prisoner's dilemma game is fundamental to many theories of human cooperation and trust. Based on the assumption that the game can model transactions between two people requiring trust, cooperative behavior in populations may be modeled by a multi-player, iterated version of the game. The theory behind the game has captivated many scholars over the years. The payoff in this game is a reduction in prison sentencing of very good, fairly good, fairly bad or very bad, which is translated into a point score system as follows: Image by Sabrina Jiang © Investopedia 2020. The game is played iteratively for a number of rounds until it is ended (as if you are repeatedly interrogated for separate crimes). The Prisoner’s Dilemma knows the following solutions: Dominant strategy: "defect". Common good solution: "cooperate"/"cooperate", (-1/ -1). An interactive simulation of the iterated Prisoner's Dilemma. Tit-for-Tat in sticklebacks. That Tit-for-Tat is an efficient strategy in iterated prisoner's dilemmas has been shown in an exciting experiments with sticklebacks - a family of fish. Such sticklebacks were confronted with a Cichlid (a predator) put behind a glass wall in their aquarium. In the experiment, a mirror projection presented "cooperative" as well as "deceiving" behavior to the fish. Sticklebacks with a "cooperating mirror" stayed in the (unsafe) front half of the aquarium twice as often as sticklebacks with a "deceiving mirror". They seemed to adapt the strategy of tit-for-tat. The dilemma faced by the prisoners here is that, whatever the other does, each is better off confessing than remaining silent. But the outcome obtained when both confess is worse for each than the outcome they would have obtained had both remained silent. Particular attention is paid to iterated and evolutionary versions of the game. In the former, the prisoner’s dilemma game is played repeatedly, opening the possibility that a player can use its current move to reward or punish the other’s play in previous moves in order to induce cooperative play in the future. In the latter, members of a population play one another repeatedly in prisoners dilemma games and those who get higher payoffs reproduce more rapidly than those who get lower payoffs. The prisoner's dilemma is a standard example of a game analyzed in game theory that shows why two completely rational individuals might not cooperate, even if it appears that it is in their best interests to do so. It was originally framed by Merrill Flood and Melvin Dresher while working at RAND in 1950. Albert W. Tucker formalized the game with prison sentence rewards and named it "prisoner's dilemma", presenting it as follows: