World Scientific Research
ISSN: 2411-6661
Vol. 2, No. 1, 1-9, 2015
http://www.asianonlinejournals.com/index.php/WSR

General Tit-For-Tat Strategy in The Three Players Prisoner's Dilemma Game

Essam El-Seidy[1*] --- Karim Mohamed[2]

[1]Department of Mathematics, Faculty of Science, Ain Shams University, Abbassia, Cairo, Egypt
[2]Department of Mathematics, Zewail City of Science and Technology, Cairo, Egypt

Abstract

Tit-For-Tat Strategy which introduced by Robert Axelrod is a highly effective strategy in the iterated Prisoner's Dilemma. Most game theory research on the prisoner's dilemma has focused on two players, but it is possible to create a Prisoner's Dilemma involving three or even more players. In this Paper, we discuss a prisoner's dilemma game involving three players which is infinitely iterated "iterated three player Prisoner's Dilemma game (I3PD)". The all possible strategies which depend on the previous outcomes are represented by finite state of automata. Four different new strategies are presented in order to discuss the general Tit-For-Tat concept in details, and we the compute the all payoff values for these strategies with the strategy ALLC and the strategy ALLD.

Keywords: Iterated games, Prisoner's dilemma, Finite automata, Tit-for-tat strategy, Perturbed payoff, Symmetric games.
JEL Classification: 91A06, 91A10, 91A15, 91A20.

Contents

* *Corresponding Author*

1. Introduction

The Prisoner's Dilemma (PD) is a traditional game model for the study of decision making and self- interest. It is only one of many illustrative examples of the logical reasoning and complex decisions involved in game theory. In order to make the best choice, each player would have to know what would the other do, but the structure of prisoner's dilemma prohibits players from having such knowledge, unless the situation or the game is repeated. When understood properly, this dilemma can multiply into hundreds of other more complex dilemmas. The mechanisms that drive the prisoner's dilemma are the same as those that are faced by marketers, military strategists, poker players, and many other types of competitors. A plethora of disciplines have studied the game, including artificial intelligence, biology, business, mathematics, philosophy, sociology, and political science [1, 2].

In the Prisoner's Dilemma game (PD) two players are faced with a choice, they can either cooperate or defect. Each player is awarded points (called payoff) depending on the choice they made compared to the choice of the opponent. Each player's decision must be made without knowledge of the other player's next move. There can be no prior agreement between the players concerning the game. If both players cooperate they both receive a reward, R points. If both players defect they both receive a punishment, P points. If one player defects and the other cooperate, the defector receives a reward, T points, the temptation to defect, while the player who cooperates is punished with the sucker's payoff, S points. Rapoport and Chammah [3]. We can represent the payoff matrix as the following:

$$\begin{array}{c c} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{pmatrix} R & S \\ T & P \end{pmatrix} \end{array} \qquad (1)$$

Where, T > R > P > S should be satisfied.

Assume a rational player is faced with playing a single game (known as one shot) of the Prisoner's Dilemma described above and that the player is trying to maximize his/her reward. If the player thinks that his/her opponent will cooperate, the player will defect to receive a reward, T points as opposed to the cooperation which would have earned him/her only, R points. However if the player thinks that his/her opponent will defect, the rational choice is to also defect and receive, P points rather than cooperate and receive the sucker's payoff of, S points. Therefore the rational decision is to always defect. But assuming the other player is also rational he/she will come to the same conclusion as the first player. Thus both players will always defect, earning rewards of, P points rather than the, R points that mutual cooperation could have yielded. Game theory has proved that always defecting is the dominant strategy for this game (the Nash Equilibrium). This holds true as long as the payoffs follow the relationship, T > R > P > S, and the gain from mutual cooperation is greater than the average score for defecting and cooperating, R > (S + T)/ 2. The dilemma here is that if both players defect, they both score worse than if both had cooperated [4].

The Iterated Prisoner's Dilemma (IPD) is an interesting variant of the above game in which two players play repeated games of the Prisoner's Dilemma against each other. In the above discussion of the Prisoner's Dilemma the dominant mutual defection strategy relies on the fact that it is a one-shot game, with no future. The key to the IPD is that the two players may play each other again; this allows the players to develop strategies based on previous game interactions [5]. Therefore a player's move now may affect how his/her opponent behaves in the future and thus affect the player's future payoffs. This removes the single dominant strategy of mutual defection as players use more complex strategies dependent on game histories in order to maximize the payoffs they receive. In fact, under the correct circumstances mutual cooperation can emerge. The length of the (IPD) (i.e. the number of repetitions of the Prisoner's Dilemma played) must not be known to either player, if it was the last iteration would become a one-shot play of the Prisoner's Dilemma and as the players know they would not play each other again, both players would defect [6].

2. Iterated Three Prisoner's Dilemma Game (I3PD)

Most game theory research on the prisoner's dilemma has focused on two player games. But it is possible to create a prisoner's dilemma game involving three or even more players. The strategies from the two player game do not necessarily extend to a three person game in a natural way. We consider a simple game with three players. Each player has two pure strategies C and D. Each round in the game leads one of the eight possible outcomes CCC, CCD, CDC, CDD, DCC, DCD, DDC or DDD, where the first position represents the player under consideration, the second and the third positions represent the opponents. For example, DCC represents the payoff to a defecting player if his both opponents cooperate. And CCD represents the payoff to a cooperating player if one of his two opponents cooperates and the other opponent defects [7]. Since we assume a symmetric game matrix, YCD could be written as YDC, where Y may be C or D. These outcomes are specified by the player's payoff R, K, S, T, L or P which can be numbered by $i = 1,2,3,4,5,6$ respectively. We impose three rules about the payoffs for the three player PD game:

(i) Defection should be the dominant choice for each player. In other words, it should always better for a player to defect, regardless what the opponents do. This rule gives three constraints:

(1) DCC > CCC, (T > R), (both opponents cooperate).
(2) DDD > CDD, (P > S), (both opponents defect).
(3) DCD > CCD, (L > K), (one opponent cooperates, one defects).

(ii) A player should always be better off if more of his opponents choose to cooperate.

(1) DCC > DCD > DDD, (T > L > P).
(2) CCC > CCD > CDD, (R > K > S).

(iii) If one player's choice is fixed, then the other two players should be left in a two player prisoner's dilemma. This rule gives the following constraints:

(1) CCD > DDD, (K > P).
(2) CCC > DCD, (R > L).

Finally, suppose the payoff matrix of three player PD game is

$$\begin{array}{cc} & \begin{array}{ccc} CC & CD & DD \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{pmatrix} R & K & S \\ T & L & P \end{pmatrix} \end{array} \qquad (2)$$

Where, $T > R > L > K > P > S$.

Assume a rational player under consideration is faced with playing a single game (known as one shot) of the three player Prisoner's Dilemma described above and that the player is trying to maximize his/her reward. The three players will always defect, earning rewards of, P, points rather than the, R, points that cooperation could have yielded. We can say that defection is the dominant strategy for the 3PD. This holds true as long as the payoffs follow the relationship,

$T > R > L > K > P > S$ [7].

Consider now the iterated game which consists of repeating the simple game infinitely often, i.e. with probability 1. In the above discussion of the three Prisoner's Dilemma the dominant defection strategy relies on the fact that it is a one-shot game, with no future. The key to the (I3PD) is that the three players may play each other again; this allows the players to develop strategies based on the previous game interactions. Therefore a player's move now may affect how his/her opponents behave in the future and thus affect the player's future payoffs. This removes the single dominant strategy of defection as the players use more complex strategies dependent on game histories in order to maximize the payoffs they receive. We assume that the three players take their decision according to the last choice of the opponents, only the last choice, by this assumption we call our game, an iterated three prisoner's dilemma with memory one. The length of the I3PD (i.e. the number of repetitions of the three Prisoner's Dilemma played) must not be known to the players, unless the all players would defect [8].

In the I3PD, each player has two choices either defect or cooperate after each outcome of the six outcomes T, R, L, K, P, S, so the total number of strategies can be composed as $2^6 = 64$ different strategies. The 64 possible strategies can be labeled by $(u_1, u_2, u_3, u_4, u_5, u_6)$ of zeroes and ones. Here u_i is 1 if the player plays C and 0 if he/she plays D after outcome i (i=1,2,3,4,5,6). For convenience, we label these rules by S_j, where j ranges from 0 to 63 and it is the integer given by (in binary notation): $u_1\, u_2\, u_3\, u_4\, u_5\, u_6$ [9, 10].

We can describe our strategies by finite state automata, more precisely, two state automata only. Each of the three players is now an automaton which can be in one of two states through any given round of the iterated 3PD). These states are corresponding to the two possible moves C and D. The state of the player in the following round depends on the present state and on the opponent's move. Hence each such automaton is specified by a graph with two nodes C and D and three oriented edge issuing from each node, one labeled C and the other D, which specify the transition from the current state to the state in the next round [8, 9]. For examples, the transition rule (1, 0, 0, 1, 1, 0) represents the strategy S_{38} represented in Fig (1).

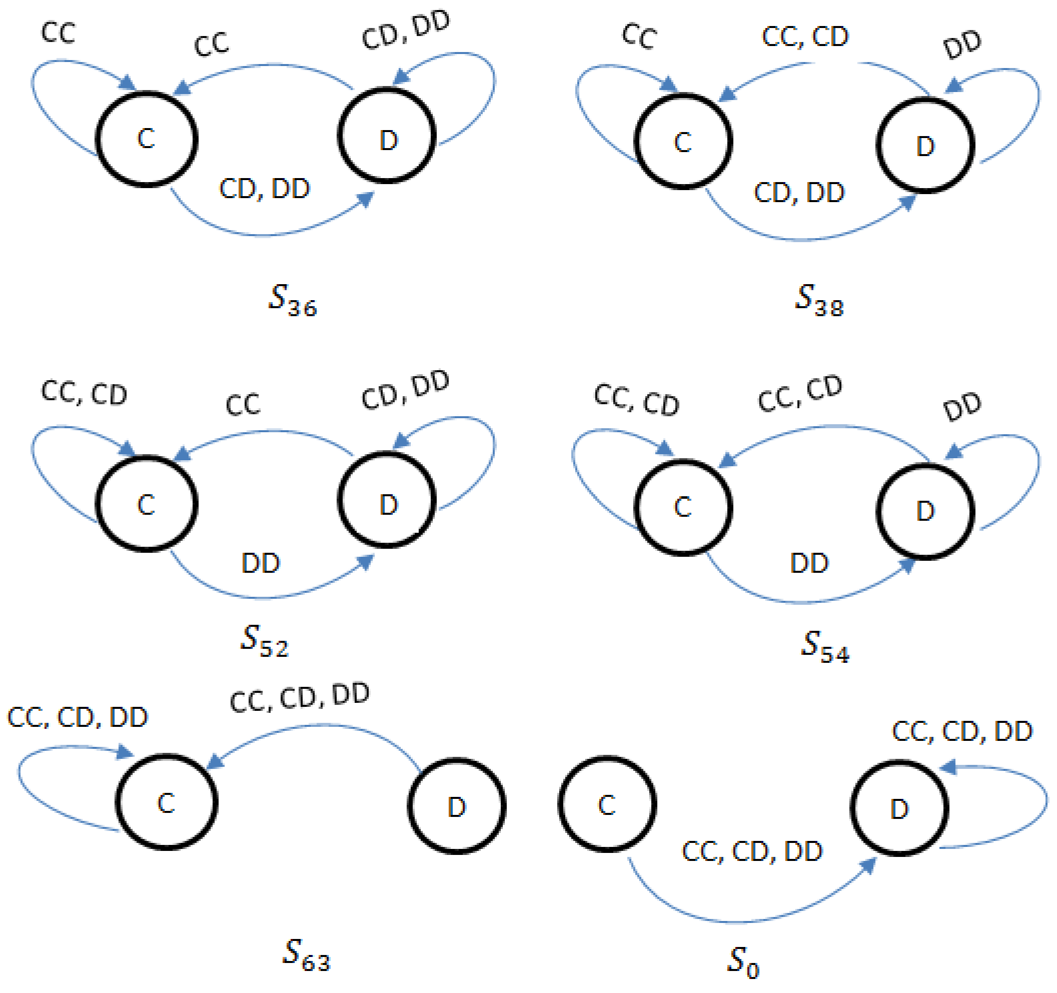

Fig-1.

How one rule fares against another depends, of course, on the initial condition of this rule [9]. Let us consider, for instance, an automaton with rule S_{36} (a retaliator never relents after defection from any one of his/her opponents unless they both cooperate again) against the two automata S_{38} (a retaliator but is more forgiving than S_{36}) and S_{52} (slow to anger than his/her opponents and slow to forgive than S_{36}):

a) If all three automata start with a C, they will keep playing C forever. The sequence looks as follows:

```
S36    C  C  C  C  C  C  C  C  C…
S38    C  C  C  C  C  C  C  C  C…
S52    C  C  C  C  C  C  C  C  C…
```

b) If all three automata start with a D, the sequence looks as follows:

```
S36    D  D  D  D  D  D  D  D  D…
S38    D  D  D  D  D  D  D  D  D…
S52    D  D  D  D  D  D  D  D  D…
```

c) If S_{36} and S_{38} start with a C while S_{63} starts with a D, the sequence looks as follows:

```
S36    C  D  D  D  D  D  D  D  D…
S38    C  D  C  D  D  D  D  D  D…
S52    D  C  D  D  D  D  D  D  D…
```

d) If S_0 and S_{63} start with a C while S_{52} starts with a D, the sequence looks as follows:

```
S36    C  D  C  D  C  D  C  D  C…
S38    D  C  D  C  D  C  D  C  D…
S52    C  C  C  C  C  C  C  C  C…
```

e) If all three automata start with a D, the sequence looks as follows:

```
S36    C  D  D  D  D  D  D  D  D…
S38    D  C  D  D  D  D  D  D  D…
S52    D  D  D  D  D  D  D  D  D…
```

f) If S_0 and S_{63} start with a C while S_{52} starts with a D, the sequence looks as follows:

```
S36    D  C  D  C  D  C  D  C  D…
S38    C  D  C  D  C  D  C  D  C…
S52    C  C  C  C  C  C  C  C  C…
```

g) If all three automata start with a D, the sequence looks as follows:

```
S36    D  D  D  D  D  D  D  D  D…
S38    C  D  D  D  D  D  D  D  D…
S52    D  D  D  D  D  D  D  D  D…
```

h) If all three automata start with a D, the sequence looks as follows:

```
S36    D  D  D  D  D  D  D  D  D…
S38    D  C  D  D  D  D  D  D  D…
S52    C  D  D  D  D  D  D  D  D…
```

The payoff in the infinitely repeated game is simply the average payoff per round. In our example, for the player under consideration who is using the transition ruleS_{36}, the payoff is R in case (a), (K+T)/2 in cases (d) and (f), and P in cases (b), (c), (e), (g) and (h). This directly means that for the player who is using the transition rule S_{38} the payoff is R in case (a), (T+T)/2 in cases (d) and (f), and P in cases (b), (c), (e), (g) and (h). Finally for the player using the transition rule S_{52}, the payoff is R in case (a), (K+T)/2 in cases (d) and (f), and P in cases (b), (c), (e), (g) and (h). We note that the payoffs are independent of the initial condition, i.e. of the moves of the players in the first round.

We can use a more direct approach [9] where the eight possible initial conditions lead (in unperturbed runs) to three possible regimes A, B and E, where A denotes the run where the three players use C, while B is the run where the S_{52}-player always play C and the other two opponents always one of them plays C and the other plays D, finally E denotes the run where the three players use D. Suppose we are in a regime A, rare perturbation causes one of the three players to play D; what follows either scenario (f), (d) or (c), and hence leads after few steps with probability 2/3 to regime B and with probability 1/3 to regime E. Suppose now that a perturbation occurs in regime B, it leads with probability 1/3 to regime A and with probability 2/3 to regime E. Suppose now that a perturbation occurs in regime E, it leads always to regime E. Hence, the corresponding transition matrix is

$$\begin{pmatrix} 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 1 \end{pmatrix} \qquad (3)$$

The corresponding stationary distribution vector is (0,0,1). This means that, an iterated game between S_{36} , S_{38} and S_{52} will be always in the regime E. The S_{36}-player receives an average payoff, P per round. This argument, repeated for each of the 64 x 64 x 64 = 262144 entries, yields a 64 payoff matrix each of them is 64x64 payoff matrix.

It is clear that, the previous manner for calculations takes a long time. Therefore, it is more practical to computerize them using any programming language. In order to achieve this; we designed an algorithm and implemented it using the Java programming language to make these calculations, and the designed algorithm is presented in the Appendix section. That code takes the three strategies corresponding to the three players as an input and gives us the regimes and their transition matrix as an output. For the player who used the strategy S_{52} against the two players using S_0 (ALL D) and S_{63} (ALL C), the output would be the only two regimes K and L such that their transition matrix is

$$\begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}. \quad (4)$$

With a stationary distribution vector $(\frac{1}{2}, \frac{1}{2})$. It means that the player who is using S_{52} receives payoff, $\frac{(K+L)}{2}$, the player using S_0 receives payoff, $\frac{(T+L)}{2}$ and the player who is using S_{63} receives payoff, $\frac{(K+S)}{2}$.

Examples:

In table 1, we present the payoff values for the three players considering that they play using either S_{10}, S_{20} or S_{30}.

Table-1.

1st player strategy	2nd player Strategy	3rd player strategy	1st player payoff	2nd player payoff	3rd player payoff
10	10	10	P	P	P
10	10	20	$\frac{2P + 3K + 3L}{8}$	$\frac{2P + 3K + 3L}{8}$	$\frac{2P + 3T + 3S}{8}$
10	10	30	P	P	P
20	20	20	P	P	P
20	20	10	$\frac{9L + 6K + 5P}{20}$	$\frac{9L + 6K + 5P}{20}$	$\frac{9S + 6T + 5P}{20}$
20	20	30	$\frac{P + L}{2}$	$\frac{P + L}{2}$	$\frac{P + S}{2}$
30	30	30	P	P	P
30	30	10	P	P	P
30	30	20	P	P	P
10	20	30	$\frac{P + 3T}{4}$	$\frac{P + 3K}{4}$	$\frac{P + 3K}{4}$

3. General Tit-For-Tat Strategy

Although there is no one best strategy for all circumstances, one that works extremely well over a wide variety of environments is a simple tit-for-tat strategy. In this strategy, one begins by cooperating and then mimics the other player's moves. Tit-for-Tat is "nice" in that it is willing to cooperate and it does not bear a grudge. It also cannot be exploited because any "defection" from cooperation will be returned. If each of the two players uses a Tit-For-Tat strategy, he/she will cooperate on the first round. If he/she discovers that the second player has defected, he/she will defect on the next round. If, after he/she realizes that the second player cooperated again, he/she becomes ready to cooperate, a Tit-For-Tat strategy is ready to begin cooperating [11]. The power of Tit-For-Tat in encouraging cooperation in unusual places has been explored by Robert Axelrod in The Evolution of Cooperation. What does Tit-For-Tat mean in the (I3PD)? Should the player defect if either the opponents defected on the previous round or only if both opponents defected? Is either of the strategies nearly as effective in the (I3PD) game as tit-for-tat is in the (I2PD) game?

To answer the previous questions, we will present four different strategies in which we can discuss the Tit-For-Tat concept in details. These four strategies are called TFT1 (S_{52}), TFT2 (S_{38}), TFT3 (S_{54}) and TFT4 (S_{36}) presented by the automata in Fig (1). The player who uses the TFT1 strategy plays the iterated prisoner's dilemma game under the condition that "If the two opponents have chosen the same choice, on the previous move, then he/she makes the choice which they did, else he/she stays on his/her choice". The player who uses the TFT2 strategy plays the iterated prisoners' dilemma game under the condition that "If the two opponents have chosen the same choice, on the previous move, then he/she makes the choice which they did, else he/she shifts his/her choice". The player who uses the TFT3 strategy plays the iterated prisoners' dilemma game under the condition that "If the two opponents have chosen the same choice, on the previous move, then he/she makes the choice which they did, else he/she plays C always". The player who uses the TFT4 strategy plays the iterated prisoners' dilemma game under the condition that "If the two opponents have chosen the same choice, on the previous move, then he/she makes the choice which they did, else he/she plays D always".

4. Tit-For-Tat, ALLC and ALLD Competition

Consider that the three players will choose one of the six strategies TFT1, TFT2, TFT3, TFT4, ALLC and ALLD, and we are going to calculate the payoff values and write it down in table form such that (in each table) the third player is using a fixed strategy, each row denotes the 1st player strategy, each column denotes the 2nd player strategy and each 6-tuble (V_1, V_2, V_3, V_4, V_5, V_6) is the stochastic vector for the 1st player asserting that it has a payoff value = $\frac{(V1.R + V2.K + V3.S + V4.T + V5.L + V6.P)}{(V1 + V2 + V3 + V4 + V5 + V6)}$.

Table-2. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=52

Third player using S_{52}	52	38	54	36	63	0
TFT1 (52)	(1,0,0,0,0,1)	(1,0,0,0,0,1)	(1,0,0,0,0,1)	(1,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT2 (38)	(1,0,0,0,0,1)	(1,2,1,1,2,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT3 (54)	(1,0,0,0,0,1)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT4 (36)	(1,0,0,0,0,1)	(0,0,0,0,0,1)	(1,0,0,0,0,1)	(0,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
AllC (63)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,1,0,0,0)
AllD (0)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,1,1,0)	(0,0,0,0,0,1)

Table-3. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=38

Third player using S38	52	38	54	36	63	0
TFT-1 (52)	(1,0,0,0,0,1)	(1,2,1,1,2,1)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT-2 (38)	(1,2,1,1,2,1)	(1,2,1,1,2,1)	(5,9,0,6,3,1)	(1,3,6,0,9,5)	(2,1,0,1,0,0)	(0,0,1,0,1,2)
TFT-3 (54)	(1,0,0,0,0,0)	(5,12,3,3,0,1)	(1,0,0,0,0,0)	(2,6,3,0,3,2)	(1,0,0,0,0,0)	(0,0,3,0,3,2)
TFT-4 (36)	(0,0,0,0,0,1)	(1,0,3,3,12,5)	(2,3,0,3,6,2)	(0,0,0,0,0,1)	(2,3,0,3,0,0)	(0,0,0,0,0,1)
All C (63)	(1,0,0,0,0,0)	(1,1,0,0,0,0)	(1,0,0,0,0,0)	(1,3,0,0,0,0)	(1,0,0,0,0,0)	(0,1,1,0,0,0)
All D (0)	(0,0,0,0,0,1)	(0,0,0,0,1,1)	(0,0,0,0,3,1)	(0,0,0,0,0,1)	(0,0,0,1,1,0)	(0,0,0,0,0,1)

Table-4. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=54

Third player using S_{54}	52	38	54	36	63	0
TFT-1 (52)	(1,0,0,0,0,1)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT-2 (38)	(1,0,0,0,0,0)	(5,9,0,3,6,1)	(1,0,0,0,0,0)	(2,3,3,3,3,2)	(1,0,0,0,0,0)	(0,0,3,0,3,2)
TFT-3 (54)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,14,6,0,6,4)
TFT-4 (36)	(1,0,0,0,0,1)	(2,3,0,3,6,2)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
All C (63)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,0,0,0,0)
All D (0)	(0,0,0,0,0,1)	(0,0,0,0,3,1)	(0,0,0,14,12,4)	(0,0,0,0,0,1)	(0,1,0,0,0,0)	(0,0,0,0,0,1)

Table-5. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=36

Third player using S_0	52	38	54	36	63	0
TFT-1 (52)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,1,0,0,1,0)	(0,0,0,0,0,1)
TFT-2 (38)	(0,0,0,0,0,1)	(0,0,1,0,1,2)	(0,0,3,0,3,2)	(0,0,0,0,0,1)	(0,1,0,0,1,0)	(0,0,0,0,0,1)
TFT-3 (54)	(0,0,0,0,0,1)	(0,0,3,0,3,2)	(0,14,6,0,6,4)	(0,0,0,0,0,1)	(0,1,0,0,0,0)	(0,0,0,0,0,1)
TFT-4 (36)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,1,0)	(0,0,0,0,0,1)
All C (63)	(0,1,1,0,0,0)	(0,1,1,0,0,0)	(0,1,0,0,0,0)	(0,0,1,0,0,0)	(0,1,0,0,0,0)	(0,0,1,0,0,0)
All D (0)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,1,0)	(0,0,0,0,0,1)

Table-6. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=0

Third player using S_{63}	52	38	54	36	63	0
TFT-1 (52)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,0,0,1,0)
TFT-2 (38)	(1,0,0,0,0,0)	(2,1,0,1,0,0)	(1,0,0,0,0,0)	(2,3,0,3,0,0)	(1,0,0,0,0,0)	(0,1,0,0,1,0)
TFT-3 (54)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,0,0,0,0)
TFT-4 (36)	(1,0,0,0,0,0)	(2,3,0,3,0,0)	(1,0,0,0,0,0)	(0,0,0,0,1,0)	(1,0,0,0,0,0)	(0,0,0,0,1,0)
All C (63)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,0,0,0,0)
All D (0)	(0,0,0,1,1,0)	(0,0,0,1,1,0)	(0,0,0,1,0,0)	(0,0,0,0,1,0)	(0,0,0,1,0,0)	(0,0,0,0,1,0)

Table-7. The payoff values of S_i against S_j and S_k, where i.j = 52,38,54,36,63,0 ,k=63

Third player using S_{36}	52	38	54	36	63	0
TFT-1 (52)	(1,0,0,0,0,1)	(0,0,0,0,0,1)	(1,0,0,0,0,1)	(0,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT-2 (38)	(0,0,0,0,0,1)	(1,3,6,0,9,5)	(2,3,3,3,3,2)	(0,0,0,0,0,1)	(2,3,0,3,0,0)	(00,0,0,0,1)
TFT-3 (54)	(1,0,0,0,0,1)	(2,6,3,0,3,2)	(1,0,0,0,0,0)	(0,0,0,0,0,1)	(1,0,0,0,0,0)	(0,0,0,0,0,1)
TFT-4 (36)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,1,0)	(0,0,0,0,0,1)
All C (63)	(1,0,0,0,0,0)	(1,3,0,0,0,0)	(1,0,0,0,0,0)	(0,0,1,0,0,0)	(1,0,0,0,0,0)	(0,0,1,0,0,0)
All D (0)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,0,0,1,0)	(0,0,0,0,0,1)

5. Results

From the previous two sections, we concluded that: there is no dominant strategy. TFT3 is more relenting than TFT1 and TFT3 more forgiving than TFT2. TFT1 will Defect forever like S_0 (ALLD) if his two opponents are S_0, one of his opponents is S_0 after being on the state D or, if his two opponents are S_{63} & S_0 after being on the state D. TFT1 is slow to anger and slow to forgive. TFT1 will Cooperate forever like S_{63} (ALLC) if his two opponents are S_{63}, one of his opponents is S_{63} after being on the state C or his two opponents are S_{63} & S_0 after being on the state

C. TFT2 will Defect forever like S_0 (ALLD) if his two opponents are S_0. TFT-2 is fast to anger and fast to forgive. TFT-2 will cooperate forever like S_{63} (ALLC) if his two opponents are S_{63}. TFT-2 will hesitate between Defection and Cooperation if his two opponents are S_{63} & S_0. TFT3 will defect forever like S_0 (ALLD) if his two opponents are S_0. TFT3 is slow to anger and fast to forgive. TFT-3 will cooperate forever like S_{63} (ALLC) if at least one of his two opponents is S_{63}. TFT3 tends to cooperate more than Defection. TFT4 will cooperate forever like S_{63} (ALLC) if his two opponents are S_{63}. TFT4 is fast to anger and slow to forgive. TFT4 will defect forever like S_0 (ALLD) if at least one of his two opponents is S_0. TFT4 tends to defect more than Cooperation.

References

[1] J. Golbeck, "Evolving strategies for the prisoner's dilemma," *Advances in Intelligent Systems, Fuzzy Systems, and Evolutionary Computation,* vol. 306, p. 299, 2002.

[2] W. Poundstone, *Prisoner's dilemma: Anchor books*. New York: Random House, INC, 2011.

[3] A. Rapoport and A. M. Chammah, *Prisoner's dilemma a study in conflict and cooperation*. Ann Arbor: University of Michigan Press, 2009.

[4] J. M. Smith, *Evolution and the theory of games*. Cambridge: Cambridge University Press, 1982.

[5] A. Errity, *Evolving strategies for the prisoner's dilemma*. Ireland: Dublin City University, 2003.

[6] A. Hilbe, M. A. Nowak, and K. Sigmund, "Evolution of extortion in iterated prisoner's dilemma games," in *Proceedings of the National Academy of Sciences,* 2013, pp. 6913-6918.

[7] M. Matsushima and T. Ikegami, "Evolution of strategies in the three-person iterated prisoner's dilemma game," *Journal of Theoretical Biology,* vol. 195, pp. 53-67, 1998.

[8] A. F. Kleimenov and M. A. Schneider, "Cooperative dynamics in a repeated three-person game with finite number of strategies," presented at the 16th IFAC World Congress, Prague, Czech Republic, 2005.

[9] M. A. Nowak, K. Sigmund, and E. El-Sedy, "Automata, repeated games and noise," *Journal of Mathematical Biology,* vol. 33, pp. 703-722, 1995.

[10] E. El Seidy, H. K. Arafat, and M. A. Taha, "The trembling hand approach to automata in iterated games," *Mathematical Theory and Modeling,* vol. 3, pp. 47-65, 2013.

[11] L. A. Imhof, D. Fudenberg, and M. A. Nowak, "Tit-for-tat or win-stay, lose-shift?," *Journal of Theoretical Biology,* vol. 247, pp. 574-580, 2007.

Bibliography

[1] R. M. Axelrod, *The evolution of cooperation*. New York, USA: Basic Books, 1984.

[2] A. Haider, *Using genetic algorithms to develop strategies for the prisoners dilemma*. Germany: University Library of Munich, 2006.

[3] M. Nowak and K. Sigmund, "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game," *Nature,* vol. 364, pp. 56-58, 1993.

Appendix

The following is the designed algorithm used for the calculations of the (I3PD) game:

```
Constant Integer ITERATIONS = 9;
Declare Integer row1[ITERATIONS], row2[ITERATIONS], row3[ITERATIONS];
Declare Character w [ITERATIONS];
FOR player1 = 0 to 63 do
    FOR player2 = 0 to 63 do
        FOR player3 = 0 to 63 do
            runGame(player1, player2, player3)
        ENDFOR
    ENDFOR
ENDFOR
procedure runGame(player1, player2, player3)
    Declare Integer x[6] = calculate_binary_of  (player1);
    Declare Integer y[6] = calculate_binary_of  (player2);
    Declare Integer z[6] = calculate_binary_of  (player3);
    Declare List of Character[ ] resultsOfBloks;
    FOR count = 0 to 8 do
        initialize_rows(count);
        FOR q = 0 to ITERATIONS – 1 do
            excute_core_game (q);
        ENDFOR
        FOR i = 0 to ITERATIONS do
            construct_result_row (i);
        ENDFOR
        Declare Character blockOfResult[ ] = calculateRepeatedSequence()
        resultsOfBloks.add(blockOfResult);
    ENDFOR
    Declare Integer transitionMatrix[][] =
            transferToTransitionMatrix(x, y, z, resultsOfBloks);\
end procedure
function calculateRepeatedSequence( )
    Declare Integer  repeatIndex = 0;
    Declare Integer  startindex = 0;
    Declare Integer  endIndex = 0;
    Declare Boolean found = false;
    FOR i = 0 to q-1
```

```
                FOR k = i+1 to q-1
                        IF (w[i] = w[k]) {
                                found = true
                                startindex = i
                                endIndex = k - 1
                                "Leave the loop"
                        ENDIF
                ENDFOR
                IF (found = true)
                        "Leave the loop"
                ENDIF
        ENDFOR
        Declare Character elements[endIndex - startindex + 1]
        FOR i = startindex to endIndex-1
                elements[i-startindex] = w[i]
                repeatIndex= repeatIndex+1
        ENDFOR
        elements[endIndex - startindex] = w[endIndex];
        return elements;
End function
procedure transferToTransitionMatrix (x, y, z, resultsOfBloks)
        LOOP resultsOfBloks
        Declare List of Character[ ] newResults
        For i = 0 to blokResult.size-1
                Declare Integer entity[ ] = elements.get(blokResult[i]);
                runNewInitials(entity, newResults); // play game 3 times with 3
        ENDFOR
        LOOP newResults
                int blockResultPosition = getBlockPosition(resultsOfBloks, blokResult);
                int newResultPosition = getBlockPosition(resultsOfBloks, newResult);
                transitionMatrix[blockResultPosition][newResultPosition] += 1;
        ENDLOOP
        ENDLOOP
End procedure
Procedure runNewInitials(Integer[ ] entity, List of Character[] newResults)
        Integer newEntity1[ ] = swapDigitOfEntity(entity, 0);
        Integer newEntity2[ ] = swapDigitOfEntity(entity, 1);
        Integer newEntity3[ ] = swapDigitOfEntity(entity, 2);
        newResults.add(excute_core_game(newEntity1));
        newResults.add(excute_core_game(newEntity2));
        newResults.add(excute_core_game(newEntity3));
End procedure
Function getBlockPosition(List of Character[] resultsOfBloks, Character[] block)
        LOOP resultsOfBloks
                IF block = resultsOfBlok
                        return i
                ENDIF
        ENDFOR
        return -1;
End function.
```

Where swap Digit of Entity just replaces 0 by 1 and vice versa, the function initialize_rows (count) is as the following:

f count =	**Set row1[0] =**	**Set row2[0] =**	**row3[0] =**
0	1	1	1
1	0	0	0
2	1	1	0
3	1	0	1
4	1	0	0
5	0	1	1
6	0	1	0
7	0	0	1

,the function excute_core_game(q) is as the following:

If row1[q] =	**row2[q] =**	**row3[q] =**	**Set row1[q+1] =**	**row2[q+1] =**	**row3[q+1] =**
0	0	0	x[5]	y[5]	z[5]
0	0	1	x[4]	y[4]	z[2]
					Continue

0	1	0	x[4]	y[2]	z[4]
0	1	1	x[3]	y[1]	z[1]
1	0	0	x[2]	y[4]	z[4]
1	0	1	x[1]	y[3]	z[1]
1	1	0	x[1]	y[1]	z[3]
1	1	1	x[0]	y[0]	z[0]

and construct_result_row(i) is as the following:

If row1[q] =	**row2[q] =**	**row3[q] =**	**Set w[q] =**
0	0	0	‘P’
0	0	1	‘L’
0	1	0	‘l’
0	1	1	‘T’
1	0	0	‘S’
1	0	1	‘K’
1	1	0	‘k’
1	1	1	‘R’