Axelrod Tournament 2.0

Carlotta Bonsignori, Gianpaolo Caramellino, Marco Ghiani

Game theorists have been dealing with the concept of *rationality* since many decades ago.
The so called Prisoner Dilemma provides an example of Nash Equilibrium that is not Pareto optimum. In this simple game, there are two players that have to choose an action out of two different possibilities; each player does not know, before she moves, which action has been chosen by the other player. After each player has selected an action, the payoff of the game is revealed.
Since the payoffs that the two players get depend both on their own action and on the other player's strategy, game theorists have developed a concept of rationality in order to include strategic behaviours, namely to include behaviours that anticipate or reply to particular actions of the opponent.
In this simple game, given the payoff structure, the theory suggests that there is an incentive to cheat for both players so that in the one shot Prisoner Dilemma the equilibrium turns out to be Pareto inefficient.
The impossibility to reach a Pareto efficient solution of the game, has led game theorists to reconsider the concept of rationality to be able to explain cooperative behaviour of people. Under certain conditions, they found that cooperation is perfectly rational when the game is infinitely repeated.
In the context of repeated games, a *strategy* is the multi period correspondence to the concept of *action* in the one shot game. To be more precise, a *strategy* is a complete plan of action, one for every period the player is called to move.
The aim of our program is the simulation of a simplification of the Axelrod Tournament.
Axelrod is an American political scientist that organized a tournament in which each participant designed a strategy to be used in a game in which the Prisoner Dilemma was iterated 200 times. Despite the complexity of some of the strategies, it turned out that the winner of the tournament was the *Tit-For-Tat* strategy, that cooperates on the first shot and then replicates what the opponent did in the previous round. To sum up, Axelrod concluded that a good strategy for the finitely repeated Prisoner Dilemma should have some features: being *nice* (never be the first to defect), being *provocable* (return defection when the other player defeated, cooperation when the other chose cooperate), being *forgiving* (do C again when your partner chooses C again) pays off.

In what follows we briefly summarize the structure of the game.

The game The game is the classical Prisoner Dilemma, where two players choose simultaneously their actions and get an outcome which depends both on their own actions and on that of the competitor.
We assume the following payoff matrix:

		2	
		C	D
1	C	$3,3$	$0,5$
	D	$5,0$	$1,1$

By simple inspection, one can easily see that if the game is played only once, no matter what the other player's action is, it is always better to defeat.
The payoff structure is a crucial determinant for the result of our simulation: by fixing those payoffs we will find that certain strategies will behave better than others. If the payoffs would have been fixed differently, the tournament would have led to different classifications of the strategies. What we want to stress again is that with this payoff structure, if the game were played just once (or finitely many times), the players would have an incentive to defect (i.e., to play D).

The tournament The tournament is played by sixteen players, to each of which will be assigned one strategy. Every player has to play against all the others and every round consists in repeating the static Prisoner Dilemma a number of times. The number of times, that we made a choice variable, is a factor that characterizes the classification of the strategies.
Players (and therefore strategies) are classified by evaluating the average payoffs they get in the entire tournament. In our program, the average payoff is the number that is written on the player after selecting the *Start-meeting* command.

The players We'll create 16 players, with specific strategies. We have the following 8 strategies:

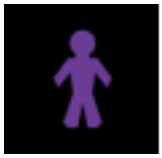
- always C: the player will always play C, no matter what the past history of the game is (i.e., no matter what she or her competitor played the previous shot);

- always D: the player will always play D, no matter what;

- Tit-for-Tat: the player will initially choose C and then, after seeing what the competitor chose in the last shot, she will play the opponent's previous action;

- Tit-for-2Tat: the structure is the same of Tit-for-Tat; what differs is that the player will play D only after 2 consecutive Ds of the competitor, so she is less easily provokable;

- 2Tit-for-Tat: the structure is the same of Tit-for-Tat; what differs is that the player will again play C (after he played D) only after 2 consecutive Cs of the competitor, so she is less forgiving;



- random 1: the player will play $\frac{1}{2}$ of the times C and $\frac{1}{2}$ D;

- random 2: the player will play $\frac{8}{10}$ of the times C and $\frac{2}{10}$ D, so she is more cooperative then random1;

- random 3: the player will play $\frac{2}{10}$ of the times C and $\frac{8}{10}$ D, so she is less cooperative then random1.

The Worlds We decided to give the possibility to choose among three different *worlds*.

- Baseline world: in this world the 16 players are paired up with the 8 different strategies, so that there is a couple of players for each strategy;

- Cooperative world: in this world there are 9 players that play the strategy *always C* and the other 7 players are endowed each one with one of the remaining 7 strategies;

- Defective world: in this world there are 9 players that play the strategy *always D* and the other 7 players are endowed each one with one of the remaining 7 strategies.

The world can be selected with a slider. We find different rankings of the strategies depending on what world has been selected.

Comment of the procedures

The procedure start-meetings is the core of the program since it lets players meet, it makes each player collect her own payoff and it stores the relevant information in players memory. In what follows we present a concise description of the procedure.

```
to start-meetings
  let i 0
  while [i < iterations]
  [
  ask players [
    let partner [who] of other players
      while [length partner > 0]
        [set currentpartner first partner
         set partner but-first partner
         if [who] of player currentpartner > [who] of self

           [choose_action
           ask player currentpartner [choose_action get_payoff]
           get_payoff
           set partner_defected_before? replace-item currentpartner partner_defected_before? item currentpartner partner_defected?
           set partner_defected? replace-item currentpartner partner_defected? [defect?] of player currentpartner
           ask player currentpartner
           [
             set partner_defected_before? replace-item [who] of myself partner_defected_before? item [who] of myself partner_defected?
             set partner_defected? replace-item [who] of myself partner_defected? [defect?] of myself
           ]
          ]

         ]

        ]
  ask players [
    fd (payoff / (15 * (i + 1)) - ycor)
    set label precision (payoff / (15 * (i + 1))) 2
    ]
 wait (1 / (i + 1))
 set i i + 1
  ]
end
```

First, one player is selected randomly through the instruction *ask players*; this player is asked to create a list containing the *who* values of all the other (unselected) players, sorted randomly; this list is called *partner*.

Once the list *partner* is created, the procedure asks the selected player to assign to the global variable *currentpartner* the value of the first *who* stored in *partner*.

As it will be clear in what follows, it is essential the definition of *currentpartner* as a global variable: indeed we want the local definition of its content to be mantained also out of the procedure *start-meetings*. Then, for reasons that will be clearer afterwards, we eliminate the first element of the list partner.

There are now two possibilities: either the initially selected agent plays with the player whose *who* is equal to *currentpartner* or she doesnt. In order to

be sure that each player meets all the others, playing only one match with each one of her opponents in each tournment, we decide to let each player play only with the opponents having an higher *who* than she has.
In this way we are sure that all the players play the same number of games and that eachone meets everybody exactly one time before the tournment ends. So the procedure asks to the selected player to compare her own *who* with the value she previously assigned to *currentpartner*; if her own *who* is higher the procedure asks her to go back to the list *partner* (that meanwhile has been emptied of its first element) and, anew, attribute to the global variable *currentpartner* the value of the first element of the updated list.
If instead, the *who* of the selected player is lower than the *who* assigned to *currentpartner*, then the meeting starts: agents choose their actions, collect their payoffs, memorize what each player they met played and then the ask players command can run again selecting another player and repeating all the commands until all players have executed the commands in the block.
The choice of an action is accomplished by asking the player to refer to another procedure, named *chooseaction*; *chooseaction* asks the player to choose her strategy according to who she is and according to the world that has been set in the Chooser. Subsequently, each player is asked to execute the procedure *payoff*. We have defined eight distinguished strategies; in order to illustrate how to read them, we describe three of them.

```
to cooperate
  set defect? false
  set color red
end
```

If a player is sent by the procedure *chooseaction* to the procedure *cooperate*, we want her to cooperate no matter what other players do (or have done). We exploit the property *defect?* previously attributed to all players: when a player plays the cooperative strategy, she actually gives the value false to her own property *defect?*.

```
to tit_for_tat
  ifelse [who] of self != currentpartner
  [
    ifelse item currentpartner partner_defected?
    [set defect? true]
    [set defect? false]]
  [

    ifelse item [who] of myself partner_defected?
    [set defect? true]
    [set defect? false]]
  set color yellow
end
```

If a player is sent by the procedure *chooseaction* to the procedure *tit-for-tat*, we want her to imitate what her opponent did in the former round played. We must distinguish two cases in terms of computer instructions: the case in which the selected player is executing *tit-for-tat* and the case in which the player whose *who* is equal to *currentpartner* is doing it. The first *ifelse* fulfill this role. If the selected player is executing the procedure, the first block of instructions is performed: the player inspects her property *partner-defected?* which is a list of true/false values (initialized as a list of sixteen falses); in particular she checks whether the element corresponding to the player associated to the variable *currentpartner* is a false. If this is the case, the player is asked to set *defect?* as false, othervise as true. If the procedure is being executed by the player associated to *currentpartner*, the instructions must be modified taking into account that the element of *partner-defected?* that must be checked is the one corresponding to the initially selected player, easily identified by the use of *myself* (since the initially selected player is asking to the player identified by *currentpartner* to execute procedure *tit-for-tat*).

```
to random2
  ifelse random 10 < 8
  [set defect? false]
  [set defect? true]
  set color orange
end
```

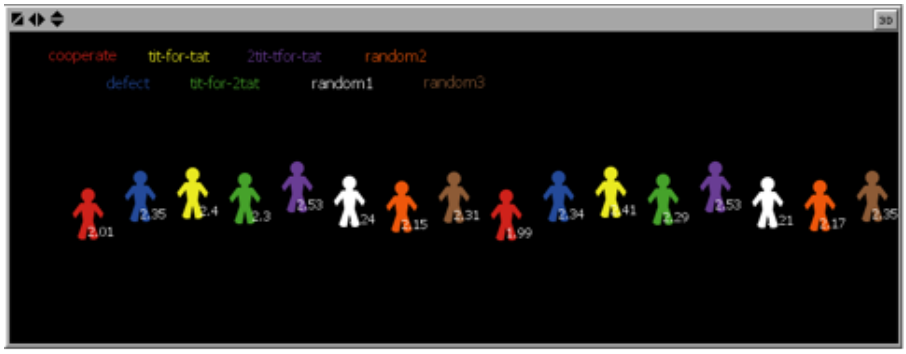
If the selected player is sent by the procedure *chooseaction* to the procedure *random2*, then a number in the interval [0, 9] is randomly generated and, according to the outcome, is determined the value to attribute to *defect?*: if the generated number is smaller than eight the player cooperates (sets *defect?* as false) otherwise she defects (therefore with 20% of probability). Once actions have been set and payoffs have been collected, we require each agent to memorize somehow what her opponent has done. Each player must be able at each stage to recollect what her opponent did up to two periods earlier. At this scope, in the setup, we set two properties of the agents (both of them initialized as lists of sixteen falses): *partner-defected?* and *partner-defected-before?*. It is worth to notice that instructions are slightly different for the initially selected player and the *currentpartner*: the initially selected player memorizes what her partner has done setting the element corresponding to her partner in the list *partner-defected-before?* as the value corresponding to *defect?* of her partner; moreover she has to update the list *partner-defected?* replacing the item corresponding to her partner with the value of the *defect?* of her partner. The player corresponding to *currentpartner*, should do the same but referring to the initially selected player which is identified (in terms of NetLogo language) by *myself*.

Experiments

Baseline world, 200 iterations We first present the result of the experiment run in the baseline world -the one in which each strategy is played by two players- for 200 iterations.

Here it is very interesting to notice that the winning strategy is the *2Tit-for-Tat*, closely followed by the *Tit-For-Tat*. Remember that the difference between the two lies in the fact that *2Tit-for-Tat* is a less forgiving strategy, so one which will play a greater number of *Ds*. The two strategy behave the same when paired up with *Cooperate*, *Defect*, *Tit-for-2Tat* and against themselves, however when playing with *random** the less forgiving strategy can turn out more effective, especially against *random2*, which plays C 80% of the times.

The strategy behaving worse is *cooperate*, this was predictable since such strategy ties with *Tit-for-Tat*, *Tit-for-2Tat*, *2Tit-for-Tat*, and itself, but it looses with *Defect* and all three *Random**.



Cooperative world, 200 iterations We present here the result of the experiment run in the cooperative world -the one in which each strategy but *cooperate* is played by one player and *cooperate* is played by nine players- for 200 iterations.

As expected the winner in this experiment is the player following the strategy *defect*. In fact, we are in a world in which most of the games are played against a player following the strategy *cooperate*, and no strategy loses against her:

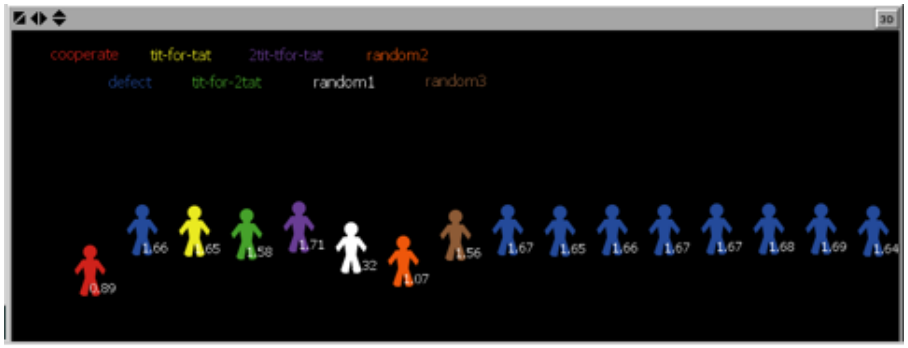
- the three strategies *Tit-for-Tat*, *Tit-for-2Tat* , *2Tit-for-Tat* tie with her;

- the three strategies *Random1*, *Random2*, *Random3* win with her, with *Random3* getting the most, followed by *random1* and last *random2*;
- the strategy *defect* wins the most.

Selfish world, 200 iterations We present here the result of the experiment run in the selfish world -the one in which each strategy but *defect* is played by one player and *defect* is played by nine players- for 200 iterations.

Interestingly in this world the consistent winner is *2Tit-for-Tat*, this is explained by the fact that when *2Tit-for-Tat* plays against *defect*, after the first iteration in which the latter wins, the two play the whole game always defecting one another and so at the end *defect* wins by little. However *2Tit-for-Tat* gains much more points when playing against the other two *Tit-for-Tat* strategies, and it might also perform better against the *random** ones.

Obviously with so many players following strategy *defect*, the two more cooperative strategies, *cooperate* and *random2*, perform the worst.



Comparison among the three possible worlds It very interesting also to compare the three results presented above.

The striking result is that the strategy *cooperate* is the one losing in all three possible worlds, however the *Cooperative World* is the one in which the average payoffs are significantly higher then in the other two worlds. In fact the average of the average payoffs in the *Cooperative World* is 2.75, while in the *Baseline World* it is 2.29 and in the *Selfish World* it is 1.55. Notice that the average payoff of the worst strategy in the *Cooperative World*, 2.49, is much higher than the average payoff of the best strategy in the *Selfish World*, 1.71, and just slightly lower than the average payoff of the best strategy in the *Baseline World*, 2.53.

World		Baseline		Cooperative		Selfish
Winner		2Tit-for-Tat		Defect		2Tit-for-Tat
Looser		Cooperate		Cooperate		Cooperate
Players	AvP_Coop_1	2.01	AvP_Coop_1	2.5	AvP_Coop_1	0.89
	AvP_Def_1	2.35	AvP_Def_1	3.82	AvP_Def_1	1.66
	AvP_Tit-for-Tat_1	2.4	AvP_Tit-for-Tat_1	2.7	AvP_Tit-for-Tat_1	1.65
	AvP_Tit-for-2Tat_1	2.3	AvP_Tit-for-2Tat_1	2.64	AvP_Tit-for-2Tat_1	1.58
	AvP_2Tit-for-Tat_1	2.53	AvP_2Tit-for-Tat_1	2.77	AvP_2Tit-for-Tat_1	1.71
	AvP_random1_1	2.24	AvP_random1_1	3.15	AvP_random1_1	1.32
	AvP_random2_1	2.15	AvP_random2_1	2.8	AvP_random2_1	1.07
	AvP_random3_1	2.31	AvP_random3_1	3.61	AvP_random3_1	1.56
	AvP_Coop_2	1.99	AvP_Coop_2	2.49	AvP_Def_2	1.67
	AvP_Def_2	2.34	AvP_Coop_3	2.51	AvP_Def_3	1.65
	AvP_Tit-for-Tat_2	2.41	AvP_Coop_4	2.49	AvP_Def_4	1.66
	AvP_Tit-for-2Tat_2	2.29	AvP_Coop_5	2.51	AvP_Def_5	1.67
	AvP_2Tit-for-Tat_2	2.53	AvP_Coop_6	2.49	AvP_Def_6	1.67
	AvP_random1_2	2.21	AvP_Coop_7	2.49	AvP_Def_7	1.68
	AvP_random2_2	2.17	AvP_Coop_8	2.52	AvP_Def_8	1.69
	AvP_random3_2	2.35	AvP_Coop_9	2.49	AvP_Def_9	1.64
Average		2.28625		2.74875		1.548125
Max		2.53		3.82		1.71
Min		1.99		2.49		0.89

Baseline world, 10 iterations We present the result of the experiment run in the baseline world -the one in which each strategy is played by two players- for 10 iterations.

Given the few iterations, different experiments can lead to different results given the presence of players behaving randomly. This effect is most viewable in the *Baseline World* since six out of sixteen payers, 37.5%, follow a *random** strategy.

We present here two emblematic cases:

and

Notice that in the first experiment the winning player is one following strategy *defect*, while in the second experiment the winning player is one following strategy *2Tit-for-Tat*, tying with *random3*. Also notice that a player following strategy *2Tit-for-Tat*, as we have seen above a consistent winner in the game repeated many many times, only classifies seventh in the first experiment.