Project report: another method of evaluating the performance of strategies for the iterated Prisoners' Dilemma game

Fynn Aschmoneit, Niels Bohr Institute, 2nd Block 2014/15

supervisors: Jan Härter, Florian Uekermann

Abstract:

This project report outlines a method of evaluating the performance of given strategies for the iterated Prisoners' Dilemma game. Instead of calculating the mean payoff of every strategy, this method follows counter-strategies to end up at some cluster of strategies. These strategies are considered most superior in the given set. This method is tested on two tournaments of 14 strategies: an unperturbed and a slightly perturbed one.

1. Introduction:

The Prisoners' Dilemma game is a model that investigates, how cooperation between two not directly communicating, competing individuals can evolve. The only way for them communicating is through their history of former interactions.
The game assumes two prisoners (players) alleged to have committed some crime together. Both are given the option of accusing their opponent (defect) and hence receive a smaller penalty, but only if one is not being betrayed self (highest payoff, T: ”temptation to defect”). If a player is being accused while not accusing the other one, the players' payoff is smaller (S: “sucker's payoff”) compared to the case where both players accuse each other (P: “mutual defection”). Both players not defecting each other (cooperating) is called R: “mutual cooperation”, higher payoff than P. There should not exist a benefit in alternate defecting and being betrayed to mutual cooperation.
This leads to the two governing rules:

$$T > R > P > S$$
$$2*R > T + S$$

and the payoff matrix for the two players (the first entry corresponds to player one):

		player one	
		cooperate	defect
player two	cooperate	R , R	T , S
	defect	S , T	P , P

Table 1: possible outcomes of every round

Independent of whether either player is guilty or innocent, it is better for the individual to defect. As they cannot communicate, by cooperating (and hoping for mutual cooperation) a player puts himself in an exploitable situation.
The repeated Prisoners' Dilemma game assumes that the two players are faced with this decision making repeatedly (rounds). Because both players remember their own and their opponent's history of decisions, they now have a way to communicate by means of responding to former decisions. If both players knew how many rounds were to be played, some cooperation between the players could, in principle, evolve, but they would still defect in the last round, as there is no revenge possible thereafter. But with that in mind, they would also defect in the second-last round, as betrayal could not be revenged in the last round. Hence, in a finite game, in which both players know how many rounds are played, both players play defect in every round and cooperation does, in fact, not evolve.
This argument doesn't hold in the infinitely repeated game. That is where the dilemma appears:
What decision should the individual player make to maximize his long-term payoff, when the opponent may take revenge and the payoff for mutual cooperation is not highest?
An attempt to solve this, is to develop strategies that dictate a decision, taking the history of former decisions of both players (and therefore both players' payoff in each round) into account. Descriptions of strategies for the simulation are found on page 3f.
In Axelrod [1], a tournament is described in which different strategies are played against one-another and the payoff for both players is added up (collective-payoff). Using the same payoff values as in [1], (T=5, R=3, P=1, S=0), the payoff matrix after 100 rounds with the four strategies TIT FOR TAT (S0), TAT FOR TIT (S1), COOPERATE (S11) and DEFECT (S12) looks like:

	S0	S1	S11	S12	mean
S0	600	500	600	203	476
S1	500	200	599	200	375
S11	600	599	600	500	575
S12	203	200	500	200	276

Table 2: four strategy payoff matrix after 100 rounds with mean payoff.

Axelrod assumes the strategy with the highest collective-payoff, averaged over all strategy competitions, to be the best strategy in the given set of strategies. Looking at the (constructed) tournament above, we find, that COOPERATE has the highest averaged collective-payoff. As COOPERATE cooperates in every round, without taking the opponent's decision into account, it reaches a state of repeated mutual cooperation (P) with TIT FOT TAT, TAT FOR TIT and itself, and a state if repeated Sucker's payoff (S) against DEFECT. Thus, the performance of COOPERATE against DEFECT is the worst possible but this is not taken into account in [1].
It is therefore interesting to find a method, that describes the performance of a given strategy in terms of its own payoff, (and not the collective-payoff), as, after all, the two players are opponents and mutual cooperation should be regarded as some method to maximize the payoff only.
Furthermore, I want to find out, how exploitable strategies are: TIT FOR TAT and TAT FOR TIT are not very exploitable, as betrayal is always revenged afterwards. DEFECT is not at all exploitable but it has a very poor performance. COOPERATE is maximum exploitable, what certainly lowers its performance.
Finally, I am interested in how stable strategies are, when subjected to small perturbations (noise).

2. Method:

In order to investigate the performance of strategies competing against one-another, I simulated a tournament of 14 strategies and write the payoff for one player for every competition in a payoff matrix. From there, I analyze what strategies are superior to a given strategy, i.e. if I knew what strategy my opponent is to play this round, what strategy could I best beat that with. This leads ti a network representation of the tournament after a certain number of rounds. Assuming, that there are strategies, that are not the best choice to beat any other given strategy, I exclude them from my network. Doing that, I detect clusters of fix-points of my strategy network. For the tournament, I implemented the following 14 strategies:

S0: TitForTat
Starts with cooperating in the first round. Copies opponent's last decision thereafter. This strategy is the winning strategy in the tournament described in [1]. Due to its success, many other strategies are modifications to TitForTat.

S1: TatForTit
Like TitForTat with defecting in the first round.

S2: TitForTwoTats
Cooperates in the first two rounds. Defects when opponent defected the last two rounds otherwise cooperates.

S3: TitForThreeTats
Cooperates in the first three rounds. Defects when opponent defected the last three rounds otherwise cooperates.

S4: ExploitTFTT
Designed to exploit forgiving behavior of TitForTwoTats. Starts with defecting and alters its decision thereafter.

S5: Joss
Like TitForTat with 10% chance of defecting, when TitForTat would cooperate.

S6: NaivePeacemaker
Like TitForTat with 10% chance of cooperating, when TitForTat would defect.

S7: Friedmann
Unforgiving; cooperates until opponent defects the first time. Defects thereafter.

S8: Pavlov
Cooperates at first round. Repeats last choice if payoff was 5 or 3.

S9: Adaptive
Starts with the sequence c,c,c,c,c,c,d,d,d,d,d and continues with decision that has given the better average payoff for every round thereafter.

S10: Envious
Starts with the sequence c,c,c,c,c,c,d,d,d,d,d and calculates the mean payoff for both players in every round thereafter. Defects if opponent's payoff is higher.

S11: Cooperate
Cooperates in every round.

S12: Defect
Defects in every round.

S13: Random
Cooperates and defects with equal probability in every round.

I had every strategy compete 100 000 times against every other strategy (156 000 000 rounds) and added up the player-one payoff in a payoff matrix (see table 3). Assuming my opponent plays the row-strategies, the payoff matrix reads as: “My score in playing against my opponent with strategy S3 and me playing with strategy S9 over 100 000 rounds is 100156.”

From here I get an idea of how a strategy performs against any given strategy (as opposed to the payoff matrix in table 2). I am only interested in what strategy performs best against a given strategy, so I look for the maxima in every row of my payoff matrix. Replacing the maxima in every row with 1 and all other entries with 0, I have created a connection matrix (see table 4). Doing that, I have defined that every strategy has at least one strategy beating it, while itself might not beat any other strategy.
In order to visualize these dependencies better, I transformed the connection matrix to a network, as seen in figure 1. The circles represent the different strategies (nodes) and the arrows point to the strategies, that beat the given strategy best. Hence, more that one arrow leaving a node means, that there more that one equally good responses to that strategy. Arrows that point outward of a given node are called out-link; those that point towards a given node are in-links. As described above, by definition, all nodes necessarily have out-links. The nodes that do not have in-links represent strategies, that are not most superior to any other strategy. These are not interesting when deciding what strategy to be played next.
Furthermore, assuming some node j being part of a cluster, where the nodes point at each other. If this node now had another out-link to another cluster, that has no other in-link, it is the second cluster of nodes that I am interested in: coming from node j, this cluster is equally promising as the first one. Eventually, decision making will lead to cluster 2, from where there is no way out. Therefore, it is not only important to reduce the network with respect to nodes, that have no in-links but also those, that do not belong to a fix-point cluster.
Implementing this was the most difficult part of this project and was done is several steps. First, all nodes without in-links are taken away. Second, all nodes, that don't have a self-link, i.e. an arrow pointing to itself, have all their in-links substituted to all their out-links (rewiring), and are then taken away from the network. If the network should initially not have any self-links, they will inevitably appear due to rewiring. In the third step, all of these remaining nodes are filtered, if they have an out-link other than to itself.
Thus, I can be sure of, that the remaining nodes belong to distinct fix-point clusters. The interim network of the simulation is found in figure 2. Having found the nodes, that certainly belong to fix-point clusters, I finally reconnect these nodes to those, that they were originally out-linked to. And connect these reconnected nodes to those, they were originally out-linked to and so forth. That way, I have isolated my fix-point clusters from the original network, see figure 3.
In order to simulate, how stable these strategies are in a little perturbed system (although the strategies are not designed for that), I added a noise factor to the tournament: Every time a strategy dictates a decision, it is inverted with a probability of 0.2%. This has a surprisingly high impact, as shown in the figures 4 and 5.

3. Results:

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
S0	300000	250000	300000	300000	250003	100022	300000	300000	300000	299991	299991	300000	100004	224835
S1	250000	100000	299997	299997	250000	100003	299994	100003	199998	299988	249994	299997	100000	225224
S2	300000	300002	300000	300000	400000	100753	300000	300000	300000	100104	299995	300000	100008	311915
S3	300000	300002	300000	300000	400000	107050	300000	300000	300000	100156	299999	300000	100012	356166
S4	249998	250000	150000	150000	200000	254770	240080	299997	225000	299991	249996	150000	300000	224922
S5	100020	100003	100771	142084	230043	100024	211871	100015	189431	270150	100010	269691	100004	207995
S6	300000	299999	300000	300000	265165	237490	300000	300000	300000	299991	299991	300000	139928	242172
S7	300000	100003	300000	300000	50007	100019	300000	300000	300000	100004	100014	300000	100004	49972
S8	300000	200003	300000	300000	225000	207017	300000	300000	300000	300000	200004	300000	300000	225641
S9	299991	299993	100034	100021	50021	319739	90173	100059	299985	299990	299990	18	100024	50060
S10	299991	249999	299990	299989	250001	100026	299991	100019	200009	299990	299990	299985	100024	224773
S11	300000	300002	300000	300000	400000	320274	300000	300000	300000	499988	300010	300000	500000	400088
S12	99999	100000	99998	99997	50000	99999	89989	99999	50000	99994	99994	0	100000	50099
S13	224968	224948	187054	168715	223664	232519	217280	299483	224713	299849	225246	149667	300644	226638

table 3: payoff matrix one-player after 100 000 rounds per entry

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
S0	1	0	1	1	0	0	1	1	1	0	0	1	0	0
S1	0	0	1	1	0	0	0	0	0	0	0	1	0	0
S2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S3	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S4	0	0	0	0	0	0	0	0	0	0	0	0	1	0
S5	0	0	0	0	0	0	0	0	0	1	0	0	0	0
S6	1	0	1	1	0	0	1	1	1	0	0	1	0	0
S7	1	0	1	1	0	0	1	1	1	0	0	1	0	0
S8	1	0	1	1	0	0	1	1	1	1	0	1	1	0
S9	0	0	0	0	0	1	0	0	0	0	0	0	0	0
S10	1	0	0	0	0	0	1	0	0	0	0	0	0	0
S11	0	0	0	0	0	0	0	0	0	0	0	0	1	0
S12	0	1	0	0	0	0	0	0	0	0	0	0	1	0
S13	0	0	0	0	0	0	0	0	0	0	0	0	1	0

table 4: connection matrix according to payoff matrix in table 3

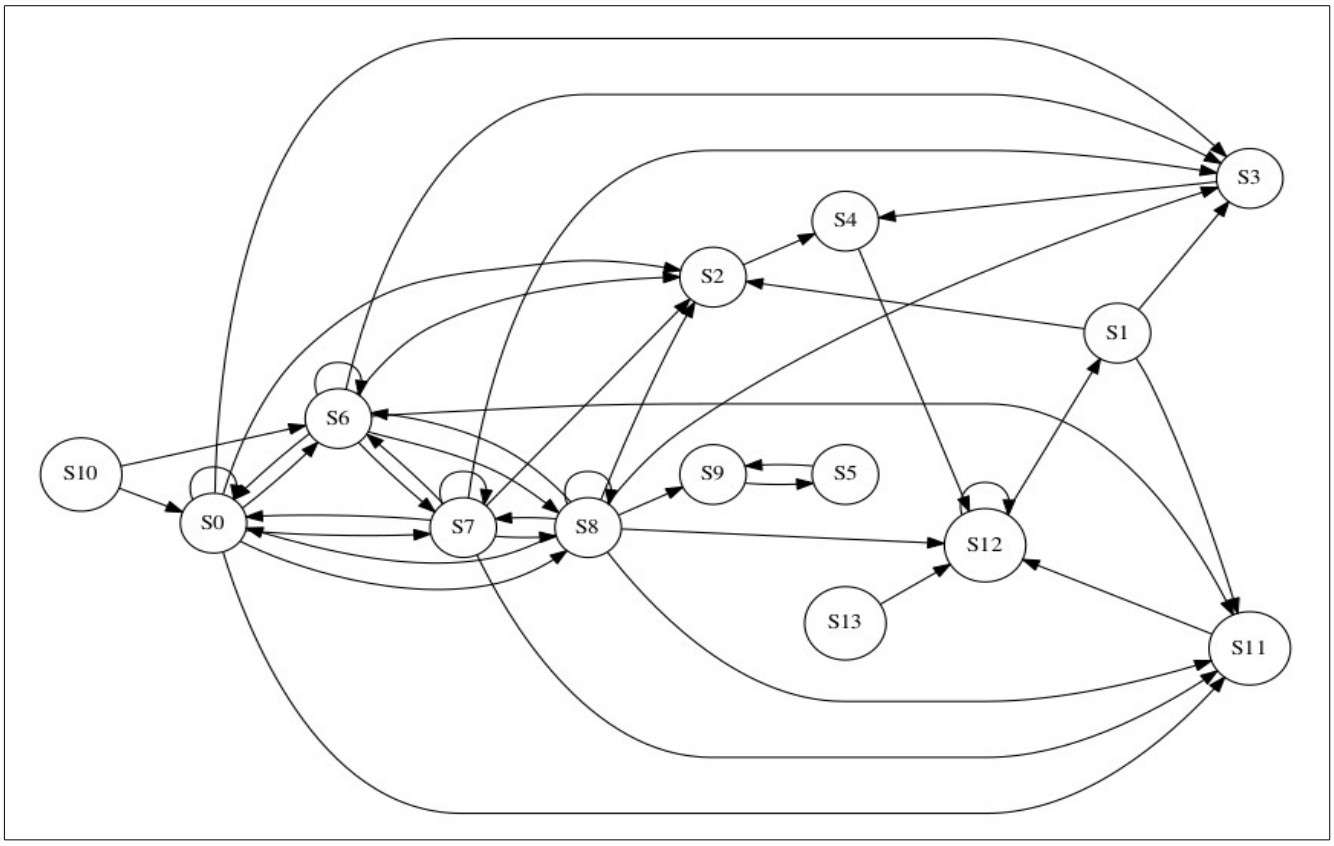

figure 1: network representation of tournament described above

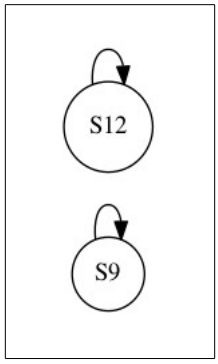

figure 2: interim network representation

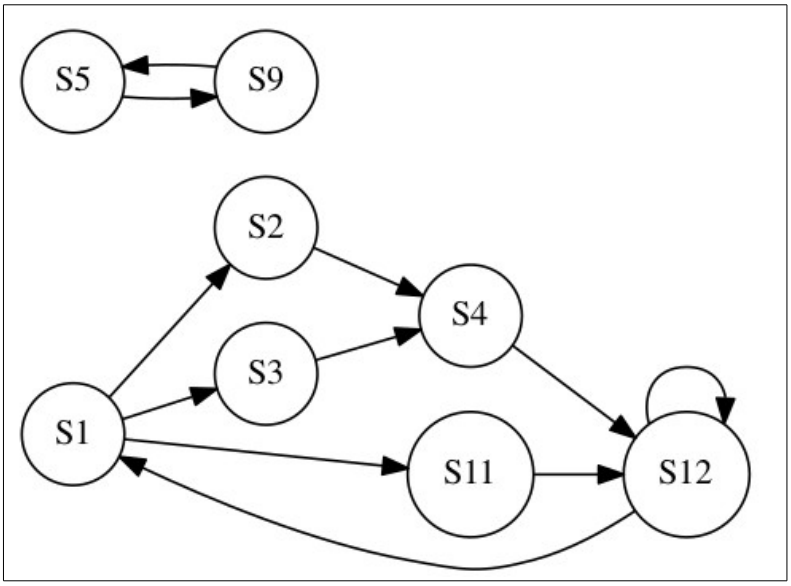

figure 3: fix-point cluster network

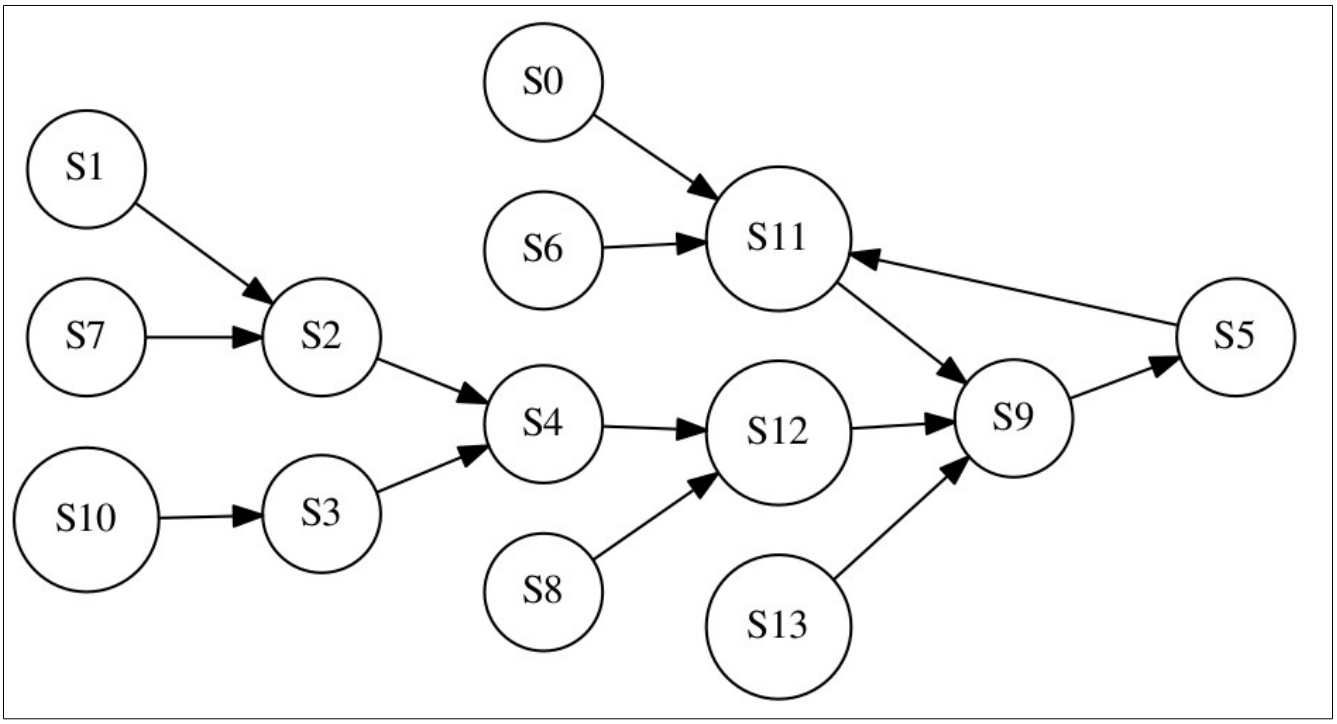

figure 4: network representation of perturbed tournament with 15 600 000 rounds played

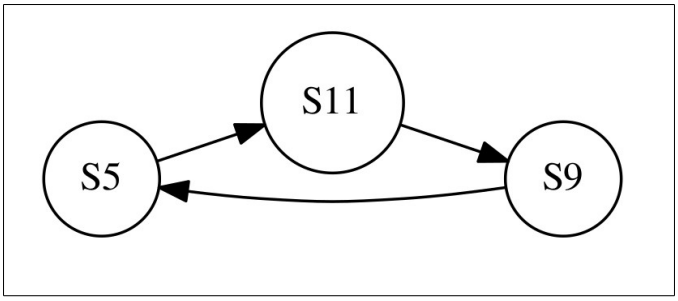

figure 5: fix-point cluster network of perturbed network

4. Conclusion:

I have implemented the method of finding isolated fix-points or fix-point clusters on a 14-strategy tournament of the iterated Prisoners' Dilemma game.
Of these 14 strategies, the method finds the most successful strategies and detects how equally good strategies counter each other in fix-point clusters. This gives a good impression on how well strategies perform with respect to each other. Thus, it does not provide one with a list of highest scoring strategies but rather with a map of performance of strategy and counter-strategy.
Finally, by having the 14 strategies compete in a little perturbed tournament, this method may be used to test the stability of strategies.

5. Reference:

[1] Robert Axelrod, The evolution of cooperation, Basic Books Inc. Publishers (1984)