Reinforcement Learning Produces Dominant Strategies for the Iterated Prisoner's Dilemma

Marc Harper Vincent Knight

Abstract

We present tournament results and several powerful strategies for the Iterated Prisoner's Dilemma created using reinforcement learning techniques (evolutionary and particle swarm algorithms). These strategies are trained to perform well against a corpus of over 170 distinct opponents, including many well-known strategies from the literature. All the trained strategies win standard tournaments against the total collection of other opponents. We also trained variants to win noisy tournaments.

1 Introduction

The iterated prisoner's dilemma (IPD) is an area of game theoretic research aiming to understand the evolution of cooperative behaviour from complex dynamics [15].

The Axelrod library [52, 31] is an open source software for conducting IPD research with reproducibility as a principal goal. Written in the Python programming language, to date over the library contains source code contributed by over 50 individuals from a variety of geographic locations and technical backgrounds. The library is supported by a comprehensive test suite that covers all the intended behaviors of the strategies in the library, as well as the features that conduct matches, tournaments, and population dynamics.

The library is continuously developed and as of version 3.0.0, the library contains over 200 strategies, many from the scientific literature, including classic strategies like Win Stay Lose Shift [45] and previous tournament winners such as OmegaTFT [48], Adaptive Pavlov [34], and ZDGTFT2 [50].

Since Robert Axelrod's seminal tournament [12], a number of IPD tournaments have been undertaken and are summarised in Table 1. Further to the work described in [31] a regular set of standard, noisy and probabilistic ending tournaments are carried out as more strategies are added to the Axelrod library. Details and results are available here: `http://axelrod-tournament.readthedocs.io`. This work presents a detailed analysis of a tournament with 176 strategies, more details about this are given in Section 3.

Year	Reference	Number of Strategies	Type	Source Code
1979	[12]	13	Standard	Not immediately available
1979	[13]	64	Standard	Available in FORTRAN
1991	[19]	13	Noisy	Not immediately available
2002	[49]	16	Wildlife	Not applicable
2005	[30]	223	Varied	Not available
2012	[50]	13	Standard	Not fully available
2016	[31]	129	Standard	Fully available

Table 1: An overview of a selection of published tournaments. Not all tournaments were 'standard' round robins; for more details see the indicated references.

In this work we describe how collections of strategies in the Axelrod library have been used to train new strategies specifically to win IPD tournaments. These strategies are trained using generic strategy archetypes based on e.g. finite state machines, arriving at particularly effective parameter choices through evolutionary or particle swarm algorithms. There are several previous publications that use evolutionary algorithms to evolve IPD strategies in various circumstances [2, 3, 7, 9, 10, 17, 23, 39, 51, 57]. See also [26] for a strategy trained to win against a collection of well-known IPD opponents and see [24] for a prior use of particle swarm algorithms. Our results are unique in that we are able to train against a large collection of well-known strategies available in the scientific literature. Crucially, the software used in this work is openly available and can be used to train strategies in the future in a reliable manner, with confidence that the opponent strategies are correctly implemented, tested and documented. Moreover, as of the time of writing, we claim that this work contains the best known strategies for the iterated prisoner's dilemma.

2 The Strategy Archetypes

The Axelrod library now contains many parametrised strategies trained using machine learning methods. Most are deterministic, use many rounds of memory, and perform extremely well in tournaments as will be discussed in Section 3. Training of these strategies will be discussed in Section 4.

These strategies can encode a variety of other strategies, including classic strategies like Tit For Tat, handshake strategies, and grudging strategies that always defect after an opponent defection.

The various archetypes will be described in the following sections.

2.1 LookerUp

The LookerUp strategy is based on a lookup tables and encodes a set of deterministic responses based on the opponent's first n_1 moves, the opponent's last m_1 moves, and the players last m_2 moves. If $n_1 > 0$ then the player has infinite memory depth, otherwise it has depth $\max m_1, m_2$. This is illustrated diagrammatically in Figure 1.

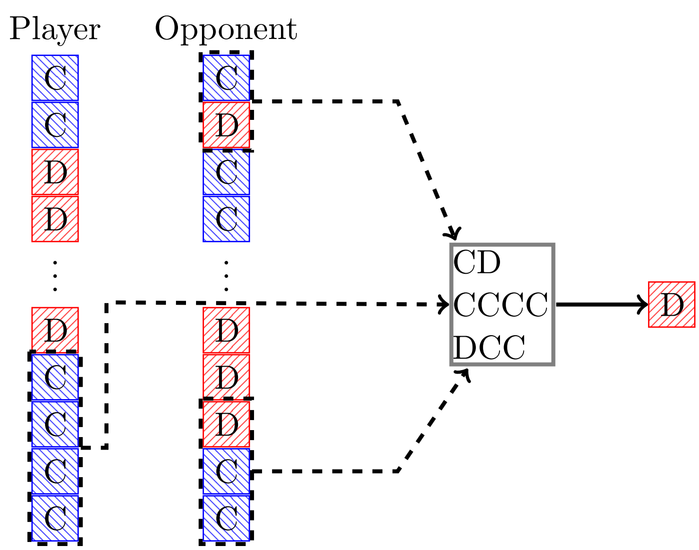


Figure 1: Diagrammatic representation of the Looker up Archetype

Training of this strategy corresponds to finding maps from histories to either a cooperation or a defection.

Although various combinations of n_1, m_1, and m_2 have been tried, the best performance at the time of training was obtained for $n_1 = m_1 = m_2 = 2$ and generally for $n_1 > 0$. A strategy called EvolvedLookerUp2_2_2 is among the top strategies in the library.

This archetype can be used to train deterministic memory-n strategies with the parameters $n_1 = 0$ and $m_1 = m_2 = n$. For $n = 1$, the resulting strategy cooperates if the last round was mutual cooperation and defects otherwise.

Two strategies in the library, Winner12 and Winner21, from [40], are based on lookup tables for $n_1 = 0$, $m_1 = 1$, and $m_2 = 2$. The strategy Winner12 emerged in less than 10 generations of training in our framework using a score maximizing objective. Strategies nearly identical to Winner21 arise from training with a Moran process objective.

2.2 Gambler

Gambler is a stochastic variant of LookerUp. Instead of deterministically encoded moves the lookup table emits probabilities which are used to choose cooperation or defection. This is illustrated diagrammatically in Figure 2.

Training of this strategy corresponds to finding maps from histories to a probability of cooperation.

The library includes a strategy trained with $n_1 = m_1 = m_2 = 2$ that is *mostly deterministic*, with most of the probabilities being 0 or 1. At one time this strategy outperformed EvolvedLookerUp2_2_2.

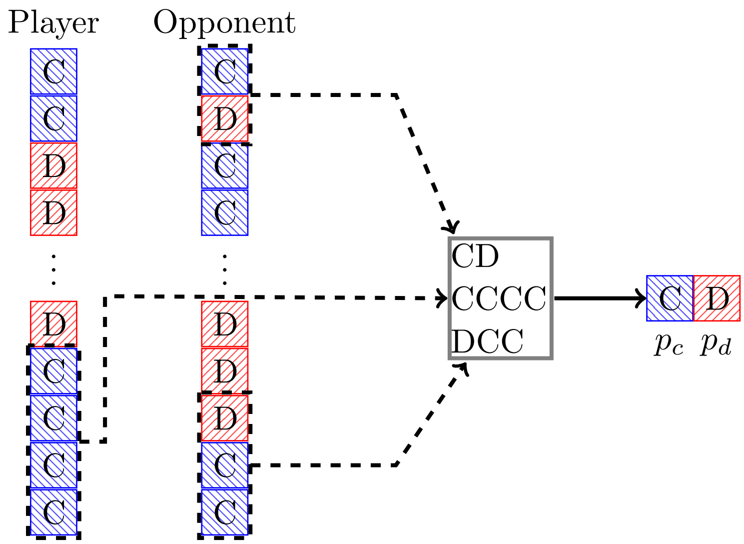


Figure 2: Diagrammatic representation of the Gambler Archetype

This strategy type can be used to train arbitrary memory-n strategies. A memory one strategy called PSO Gambler Mem 1 was trained, with probabilities $(\Pr(\mathrm{C} \mid \mathrm{CC}), \Pr(\mathrm{C} \mid \mathrm{CD}), \Pr(\mathrm{C} \mid \mathrm{DC}), \Pr(\mathrm{C} \mid \mathrm{DD})) = (1, 0.5217, 0, 0.121)$. Though it performs well in standard tournaments (see Table 2) it is not as good as the longer memory strategies, and is bested by a similar strategy that also uses the first round of play: PSOGambler_1_1_1.

These strategies are trained with a particle swarm algorithm rather than an evolutionary algorithm (though the former would suffice). Particle swarm algorithms have been used to trained IPD strategies previously [24].

2.3 ANN: Single Layer Artificial Neural Network

Strategies based on artificial neural networks use a variety of features computed from the history of play:

- Opponent's first move is C
- Opponent's first move is D
- Opponent's second move is C
- Opponent's second move is D
- Player's previous move is C
- Player's previous move is D
- Player's second previous move is C
- Player's second previous move is D
- Opponent's previous move is C
- Opponent's previous move is D
- Opponent's second previous move is C
- Opponent's second previous move is D
- Total opponent cooperations
- Total opponent defections
- Total player cooperations
- Total player defections
- Round number

These are then input into a feed forward neural network with one layer and user-supplied width. This is illustrated diagrammatically in Figure 3.

Training of this strategy corresponds to finding parameters of the neural network.

An inner layer with just five nodes performs quite well in both deterministic and noisy tournaments. The output of the ANN used in this work is deterministic; a stochastic variant that outputs probabilities rather than exact moves could be easily created.

2.4 Finite State Machines

Strategies based on finite state machines are deterministic and computationally efficient. In each round of play the strategy selects an action based on the current state and the opponent's last action, transitioning to a new state for the next round.

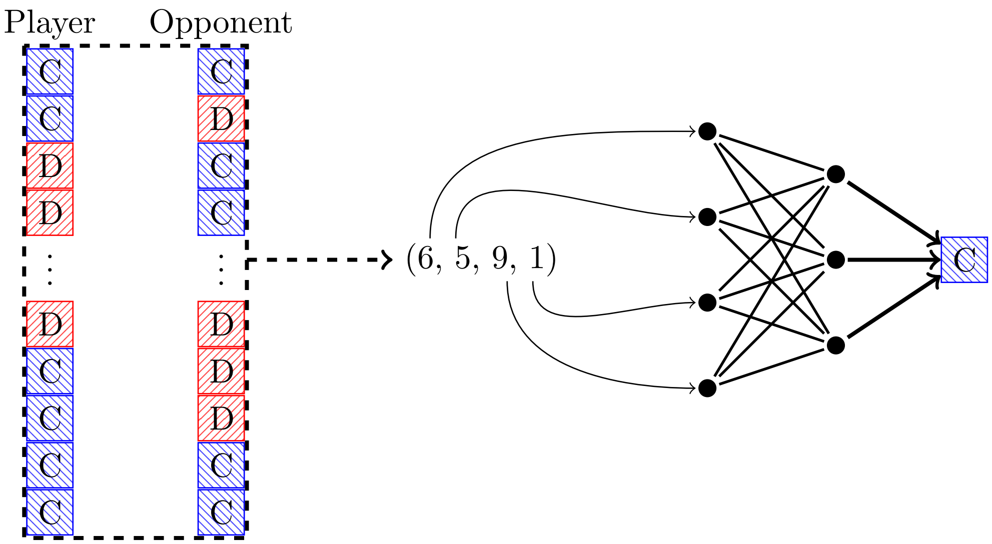


Figure 3: Diagrammatic representation of the ANN Archetype

This is illustrated diagrammatically in Figure 4.

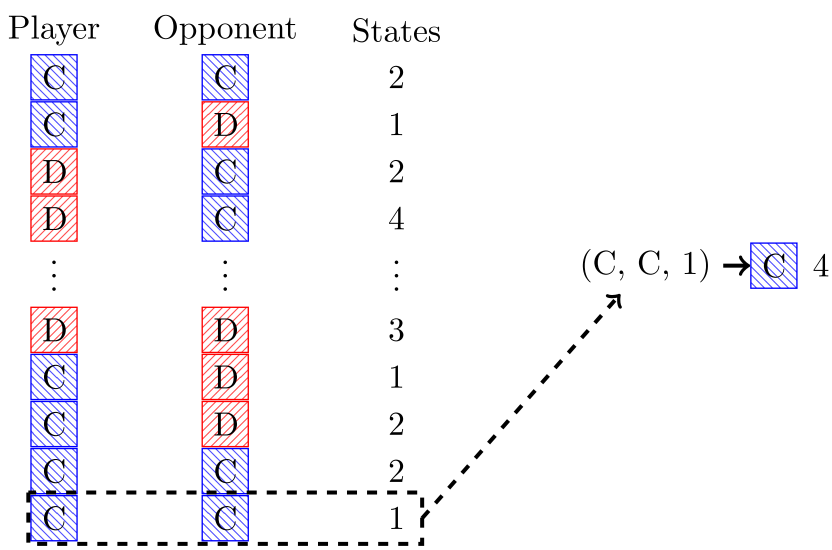


Figure 4: Diagrammatic representation of the Finite state machine Archetype

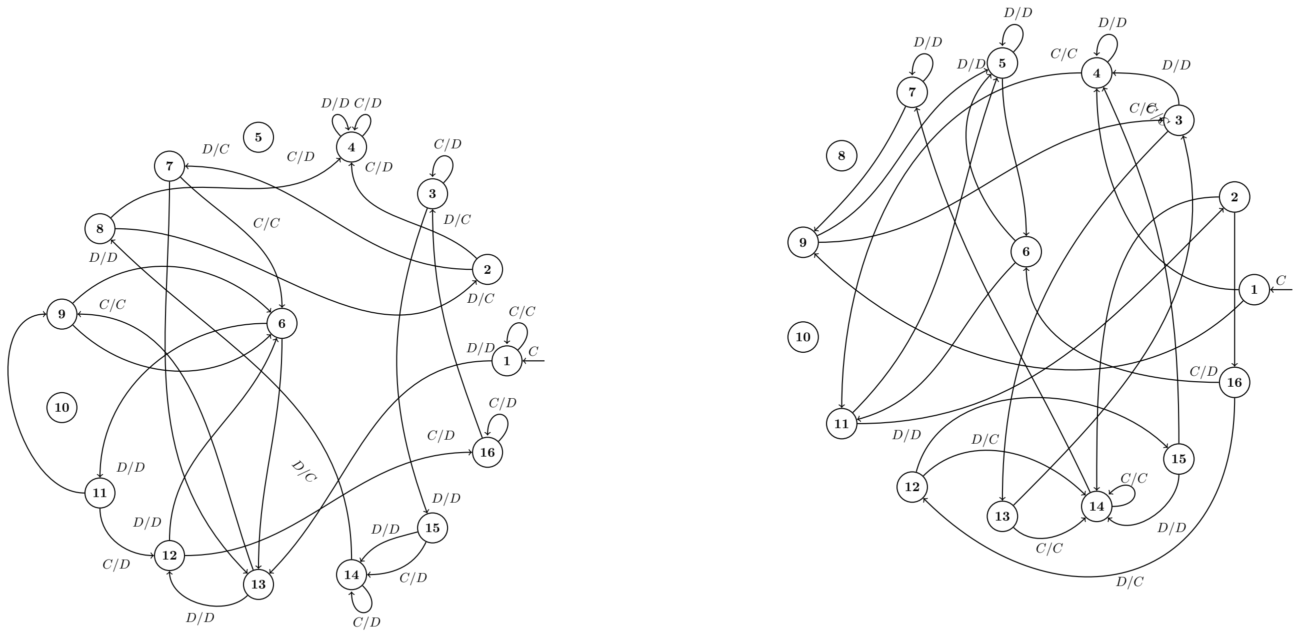
Training this strategy corresponds to finding mappings of states and histories to an action and a state. Figure 5 show two of the trained finite state machines. The layout of state nodes is kept the same between Figure 5a and 5b to highlight the effect of different training environments. Note also that two of the 16 states are not used, this is also an outcome of the training process.

2.5 Hidden Markov Models

A variant of finite state machine strategies are called hidden Markov models (HMMs). Like the strategies based on finite state machines, these strategies also encode an internal state however use probabilistic transitions based on the prior round of play to other states and cooperate or defect with various probabilities at each state. This is shown diagrammatically in Figure 6.

Training this strategy corresponds to finding mappings of states and histories to probabilities of cooperating as well as probabilities of the next internal state.

These are the best performing stochastic strategies in the library but take longer to train due to their stochasticity.



(a) Evolved_FSM_16: trained to maximise score in a standard tournament

(b) Evolved_FSM_16_Noise_05: trained to maximise score in a noisy tournament

Figure 5: Trained sixteen state finite state machine players

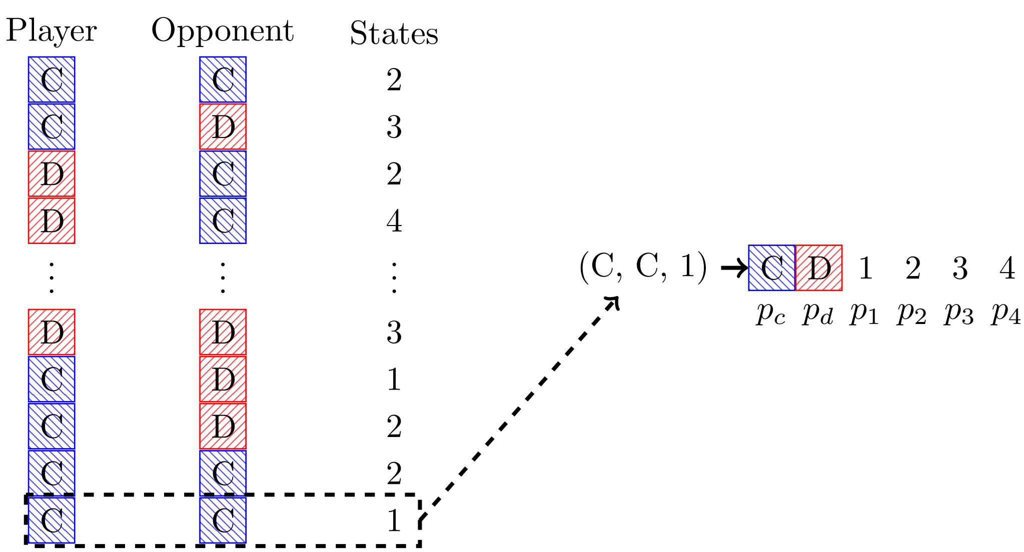


Figure 6: Diagrammatic representation of the Hidden Markov Model Archetype

2.6 Meta Strategies

Last but not least there are several strategies based on ensemble methods that are common in machine learning called Meta strategies. These strategies are composed of a team of other strategies. In each round, each member of the team is polled for its desired next move. The ensemble then selects the next move based on a rule, such as the consensus vote in the case of MetaMajority or the best individual performance in the case of MetaWinner. These strategies were among the best in the library before the inclusion of those trained by reinforcement learning.

Because these strategies inherit many of the properties of the strategies on which they are based, including using the match length to defect on the last rounds of play, not all of these strategies were included in results of this paper.

3 Results

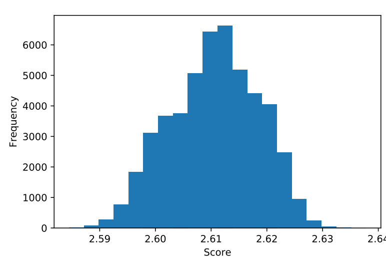
This section presents the results of a large IPD tournament with strategies from the Axelrod library, including some additional parametrized strategies (e.g. various parameter choices for Generous Tit For Tat). These are listed in Appendix A.

All strategies in the tournament follow a simple set of rules in accordance with earlier tournaments:

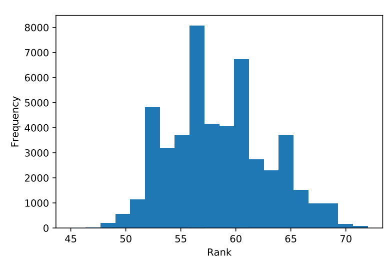
- Players are unaware of the number of turns in a match
- Players carry no acquired state between matches
- Players cannot observe the outcome of other matches
- Players cannot identify their opponent by any label or identifier
- Players cannot manipulate or inspect their opponents in any way

Any strategy that does not follow these rules, such as a strategy that defects on the last round of play, was omitted from the tournament presented here (but not necessarily from the training pool).

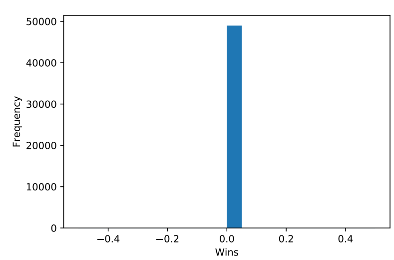
A total of 176 are included, of which 53 are stochastic. In Section 3.1 is concerned with the standard tournament with 200 turns whereas in Section 3.2 a tournament with 5% noise is discussed. Due to the inherent stochasticity of these IPD tournaments, these tournament were repeated 49000 times. This allows for a detailed and confident analysis of the performance of strategies. To illustrate the results considered, Figure 7a shows the distribution of the mean score per turn of Tit For Tat over all the repetitions. Similarly, Figure 7b shows the ranks of of Tit For Tat for each repetition (we note that it never wins a tournament). Finally Figure 7c shows the number of opponents beaten in any given tournament: Tit For Tat does not win any match (this is due to the fact that it will either draw with mutual cooperation or defect second).



(a) Scores



(b) Ranks



(c) Wins

Figure 7: Results for Tit For Tat over 49000 tournaments.

The utilities used are $(R, P, T, S) = (3, 1, 5, 0)$ thus the specific Prisoner's Dilemma being played is:

$$\begin{pmatrix} (3,3) & (0,5) \\ (5,0) & (1,1) \end{pmatrix} \tag{1}$$

3.1 Standard Tournament

The top 11 performing strategies by median payoff are all strategies trained to maximize total payoff against a subset of the strategies (Table 2). The next strategy is Desired Belief Strategy (DBS) [11]. which actively analyzes the opponent

	mean	std	min	5%	25%	50%	75%	95%	max
EvolvedLookerUp2_2_2*	2.955	0.010	2.915	2.937	2.948	2.956	2.963	2.971	2.989
Evolved HMM 5*	2.954	0.014	2.903	2.931	2.945	2.954	2.964	2.977	3.007
Evolved FSM 16*	2.952	0.013	2.900	2.930	2.943	2.953	2.962	2.973	2.993
PSO Gambler 2_2_2*	2.938	0.013	2.884	2.914	2.930	2.940	2.948	2.957	2.972
Evolved FSM 16 Noise 05*	2.919	0.013	2.874	2.898	2.910	2.919	2.928	2.939	2.965
PSO Gambler 1_1_1*	2.912	0.023	2.805	2.873	2.896	2.912	2.928	2.950	3.012
Evolved ANN 5*	2.912	0.010	2.871	2.894	2.905	2.912	2.919	2.928	2.945
Evolved FSM 4*	2.910	0.012	2.867	2.889	2.901	2.910	2.918	2.929	2.943
Evolved ANN*	2.907	0.010	2.865	2.890	2.901	2.908	2.914	2.923	2.942
PSO Gambler Mem1*	2.901	0.025	2.783	2.858	2.884	2.901	2.919	2.942	2.994
Evolved ANN 5 Noise 05*	2.864	0.008	2.830	2.850	2.858	2.865	2.870	2.877	2.891
DBS	2.857	0.009	2.823	2.842	2.851	2.857	2.863	2.872	2.899
Winner12	2.849	0.008	2.820	2.836	2.844	2.850	2.855	2.862	2.874
Fool Me Once	2.844	0.008	2.818	2.830	2.838	2.844	2.850	2.857	2.882
Omega TFT: 3, 8	2.841	0.011	2.800	2.822	2.833	2.841	2.849	2.859	2.882

Table 2: Standard Tournament: Mean score per turn of top 15 strategies (ranked by median over 49000 tournaments). The leaderboard is dominated by the trained strategies (indicated by a *

and responds accordingly. The next two strategies are Winner12, based on a lookup table, Fool Me Once, a grudging strategy that defects indefinitely on the second defection, and Omega Tit For Tat [30].

For completeness, violin plots showing the distribution of the scores of each strategy (again ranked by median score) are shown in Figure 8.

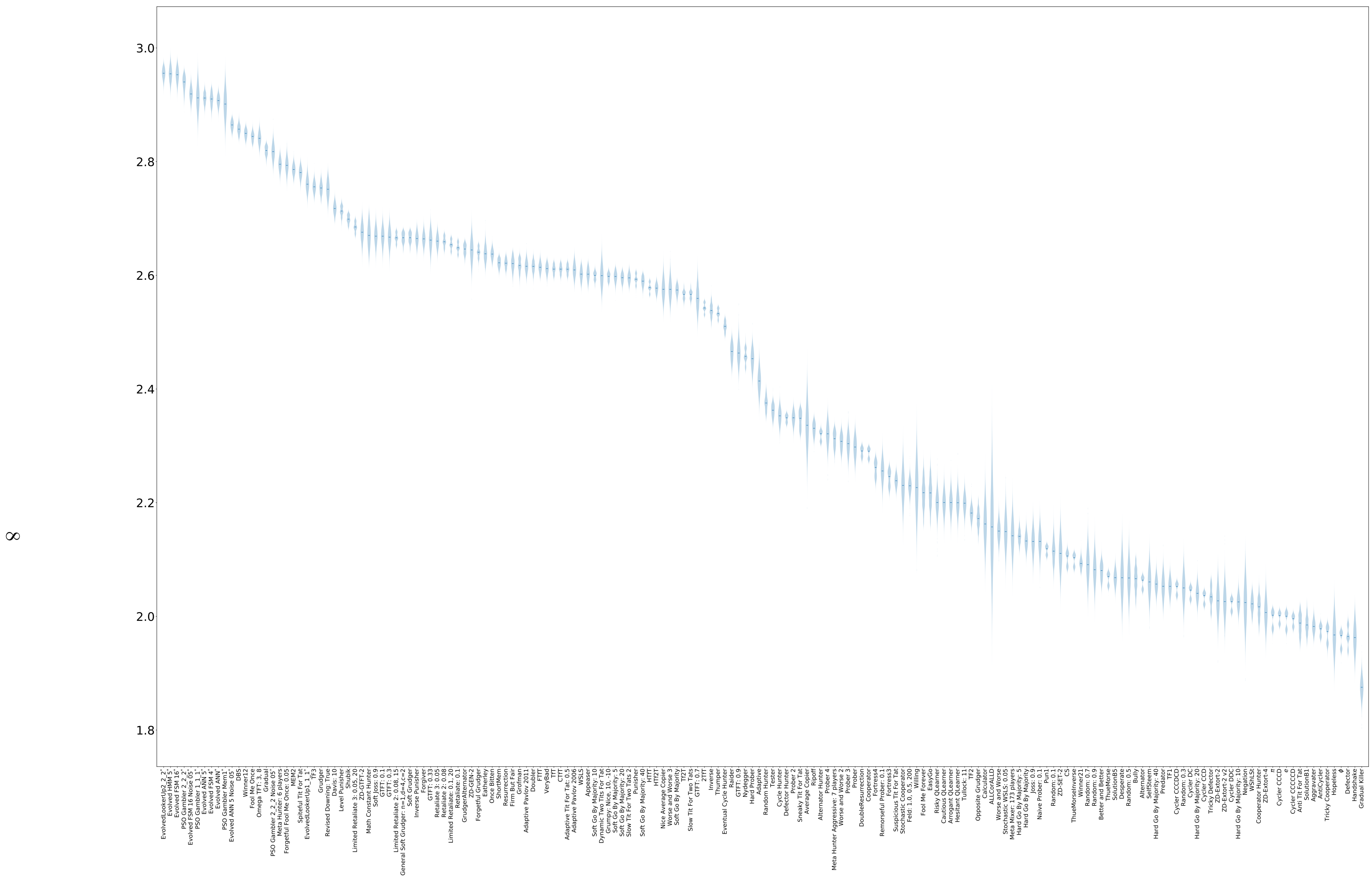


Figure 8: Standard Tournament: Mean score per turn (ranked by median over 49000 tournaments)

Pairwise payoff results are given as a heatmap (Figure 9) which shows that many strategies achieve mutual cooperation (obtaining a score of 3). The top performing strategies never defect first yet are able to exploit weaker strategies that attempt to defect.

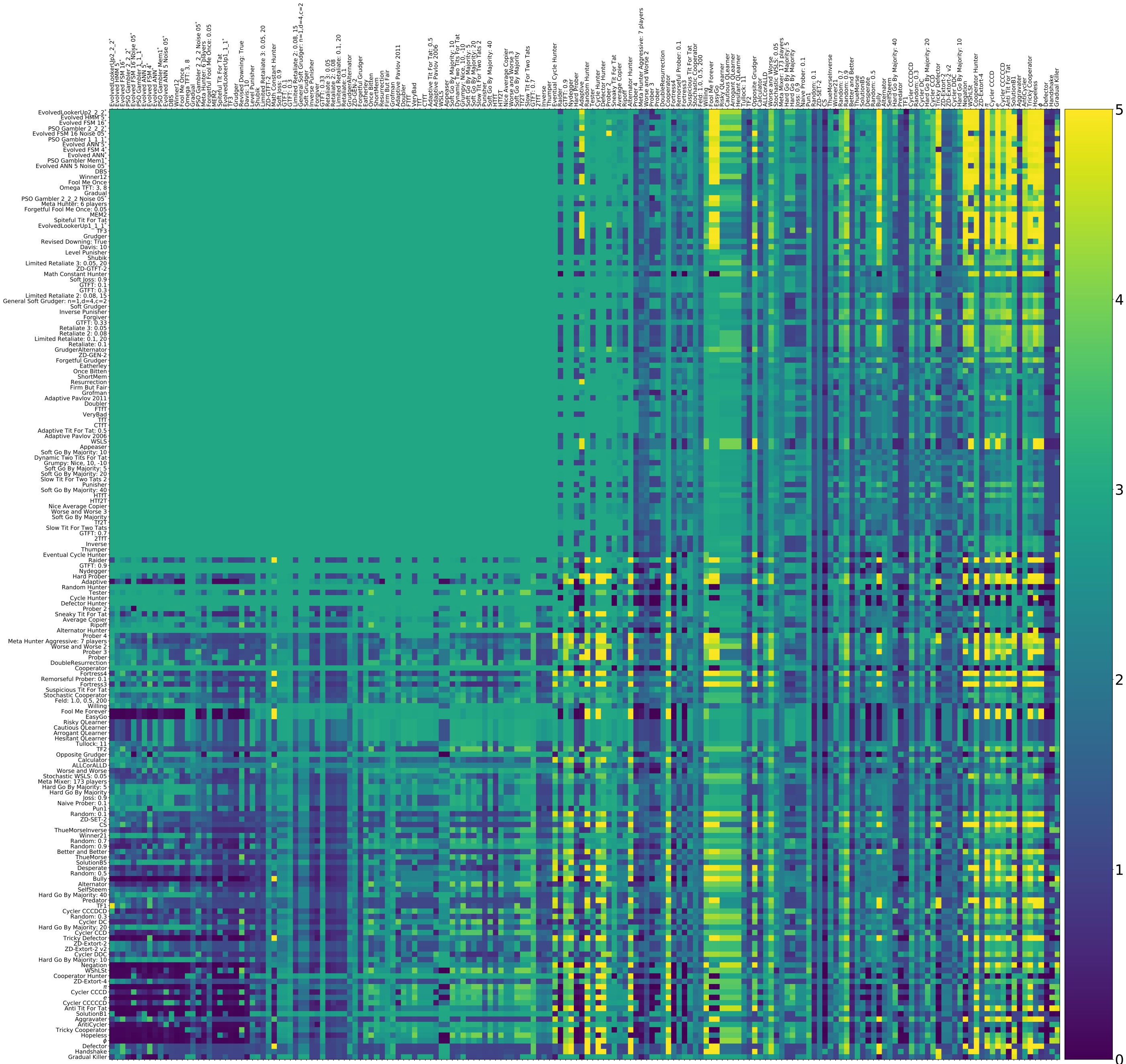


Figure 9: Standard Tournament: Mean score per turn of row players against column players (ranked by median over 49000 tournaments)

The strategies that win the most matches (Table 3 are Defector and Aggravater, followed by handshaking and zero determinant strategies. This includes two handshaking strategies that were the result of training to maximize Moran process fixation (TF1 and TF2). No strategies were trained specifically to win matches. None of the top scoring strategies appear in the top 15 list of strategies ranked by match wins. This can be seen in Figure 10 where the distribution of the number of wins of each strategy is shown.

	mean	std	min	5%	25%	50%	75%	95%	max
Aggravater	161.594	0.862	160	160.0	161.0	162.0	162.0	163.0	163
Defector	161.604	0.864	160	160.0	161.0	162.0	162.0	163.0	163
CS	159.646	1.005	155	158.0	159.0	160.0	160.0	161.0	161
ZD-Extort-4	150.602	2.661	138	146.0	149.0	151.0	152.0	155.0	162
Handshake	149.551	1.754	142	147.0	148.0	150.0	151.0	152.0	154
ZD-Extort-2	146.093	3.446	129	140.0	144.0	146.0	148.0	152.0	160
ZD-Extort-2 v2	146.292	3.426	131	141.0	144.0	146.0	149.0	152.0	160
Winner21	139.946	1.225	136	138.0	139.0	140.0	141.0	142.0	143
TF2	138.241	1.700	130	135.0	137.0	138.0	139.0	141.0	143
TF1	135.691	1.408	130	133.0	135.0	136.0	137.0	138.0	140
Naive Prober: 0.1	136.016	2.503	127	132.0	134.0	136.0	138.0	140.0	147
Feld: 1.0, 0.5, 200	136.087	1.697	130	133.0	135.0	136.0	137.0	139.0	144
Joss: 0.9	136.015	2.501	126	132.0	134.0	136.0	138.0	140.0	146
Predator	133.719	1.385	129	131.0	133.0	134.0	135.0	136.0	138
SolutionB5	125.843	1.510	120	123.0	125.0	126.0	127.0	128.0	131

Table 3: Standard Tournament: Number of wins per tournament of top 15 strategies (ranked by median wins over 49000 tournaments)

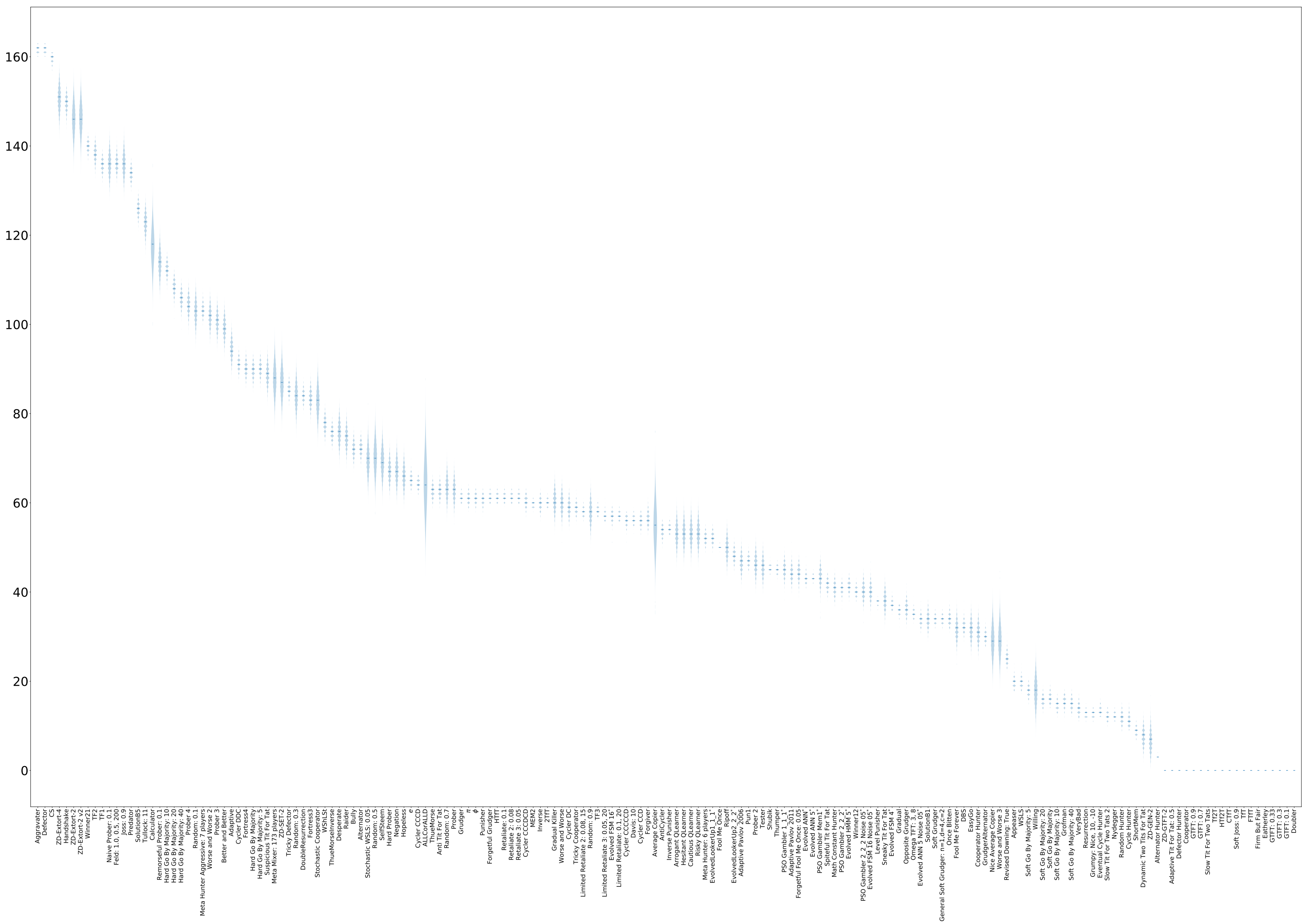


Figure 10: Standard Tournament: number of wins per tournament (ranked by median over 49000 tournaments)

The number of wins of the top strategies of Table 4 are shown in Table 4. It is evident that although these strategies score highly they do not win many matches: the strategy with the most number of wins is the Evolved FSM 16 strategy that at most won 60 ($60/175 \approx 34\%$) matches in a given tournament.

	mean	std	min	5%	25%	50%	75%	95%	max
EvolvedLookerUp2_2_2*	48.259	1.336	43	46.0	47.0	48.0	49.0	50.0	53
Evolved HMM 5*	41.358	1.221	36	39.0	41.0	41.0	42.0	43.0	45
Evolved FSM 16*	56.977	1.099	51	55.0	56.0	57.0	58.0	59.0	60
PSO Gambler 2_2_2*	40.691	1.090	36	39.0	40.0	41.0	41.0	42.0	45
Evolved FSM 16 Noise 05*	40.070	1.673	34	37.0	39.0	40.0	41.0	43.0	47
PSO Gambler 1_1_1*	45.005	1.596	38	42.0	44.0	45.0	46.0	48.0	51
Evolved ANN 5*	43.224	0.675	41	42.0	43.0	43.0	44.0	44.0	47
Evolved FSM 4*	37.227	0.951	34	36.0	37.0	37.0	38.0	39.0	41
Evolved ANN*	43.100	1.020	40	42.0	42.0	43.0	44.0	45.0	48
PSO Gambler Mem1*	43.444	1.837	34	40.0	42.0	43.0	45.0	46.0	51
Evolved ANN 5 Noise 05*	33.711	1.125	30	32.0	33.0	34.0	34.0	35.0	38
DBS	32.331	1.198	28	30.0	32.0	32.0	33.0	34.0	38
Winner12	40.179	1.037	36	39.0	39.0	40.0	41.0	42.0	44
Fool Me Once	50.121	0.422	48	50.0	50.0	50.0	50.0	51.0	52
Omega TFT: 3, 8	35.158	0.859	32	34.0	35.0	35.0	36.0	37.0	39

Table 4: Standard Tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 49000 tournaments)

Finally, Table 5 and Figure 11 show the ranks (based on median score) of each strategy over the repeated tournaments. Whilst there is some stochasticity, the top three strategies almost always rank in the top three. For example, the worst that the Evolved Lookerup 2 2 2 ranks in a given tournament is 8th.

	mean	std	min	5%	25%	50%	75%	95%	max
EvolvedLookerUp2_2_2*	2.172	1.070	1	1.0	1.0	2.0	3.0	4.0	8
Evolved HMM 5*	2.321	1.274	1	1.0	1.0	2.0	3.0	5.0	10
Evolved FSM 16*	2.489	1.299	1	1.0	1.0	2.0	3.0	5.0	10
PSO Gambler 2_2_2*	3.962	1.526	1	2.0	3.0	4.0	5.0	7.0	10
Evolved FSM 16 Noise 05*	6.300	1.688	1	4.0	5.0	6.0	7.0	9.0	11
PSO Gambler 1_1_1*	7.086	2.498	1	3.0	5.0	7.0	9.0	10.0	17
Evolved ANN 5*	7.286	1.523	2	5.0	6.0	7.0	8.0	10.0	11
Evolved FSM 4*	7.525	1.632	2	5.0	6.0	8.0	9.0	10.0	12
Evolved ANN*	7.901	1.450	2	5.0	7.0	8.0	9.0	10.0	12
PSO Gambler Mem1*	8.221	2.536	1	4.0	6.0	9.0	10.0	12.0	20
Evolved ANN 5 Noise 05*	11.363	0.873	8	10.0	11.0	11.0	12.0	13.0	16
DBS	12.195	1.125	9	11.0	11.0	12.0	13.0	14.0	16
Winner12	13.222	1.137	9	11.0	12.0	13.0	14.0	15.0	17
Fool Me Once	13.959	1.083	9	12.0	13.0	14.0	15.0	15.0	17
Omega TFT: 3, 8	14.275	1.301	9	12.0	13.0	15.0	15.0	16.0	19

Table 5: Standard Tournament: Rank in each tournament of top 15 strategies (ranked by median over 49000 tournaments)

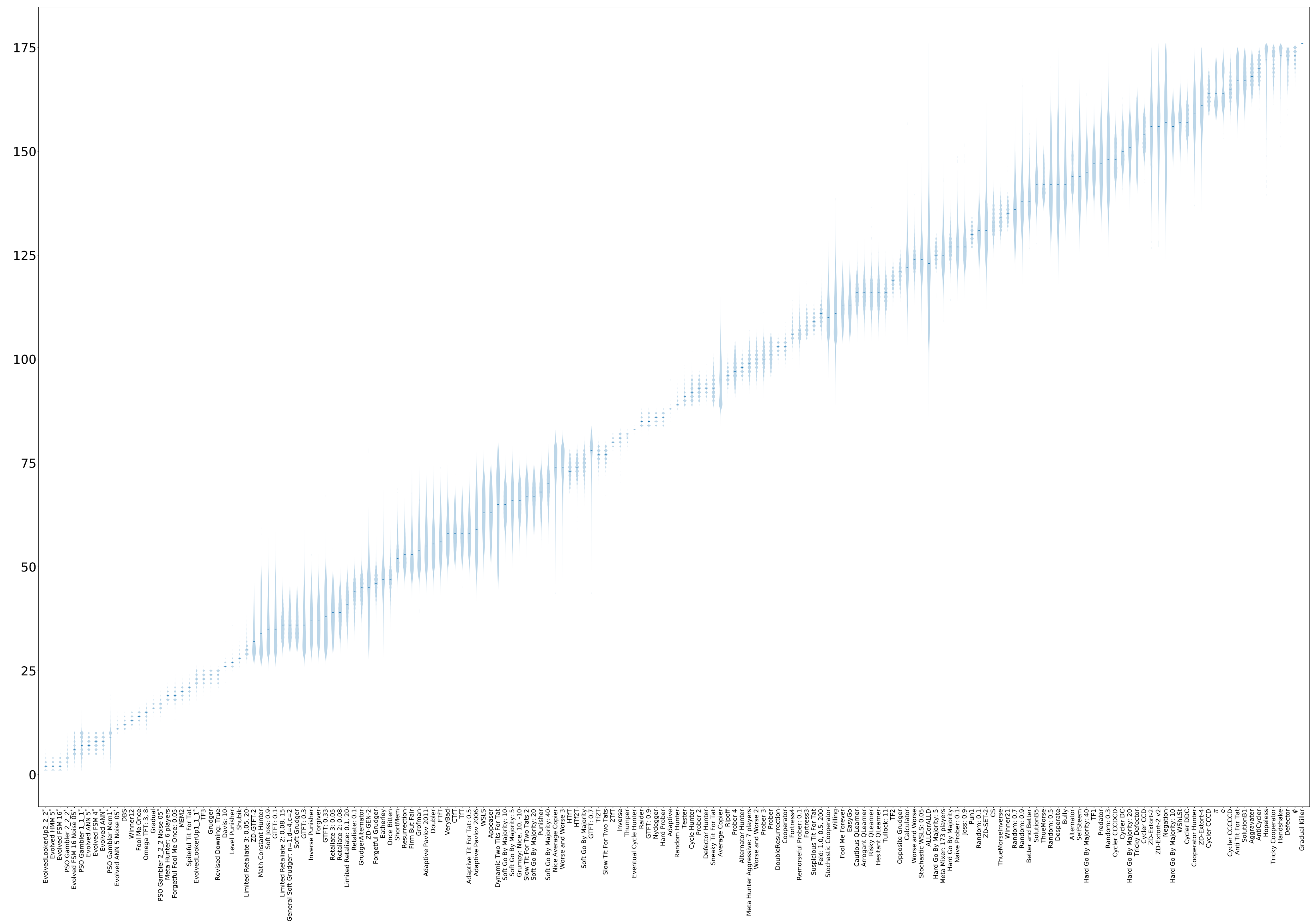
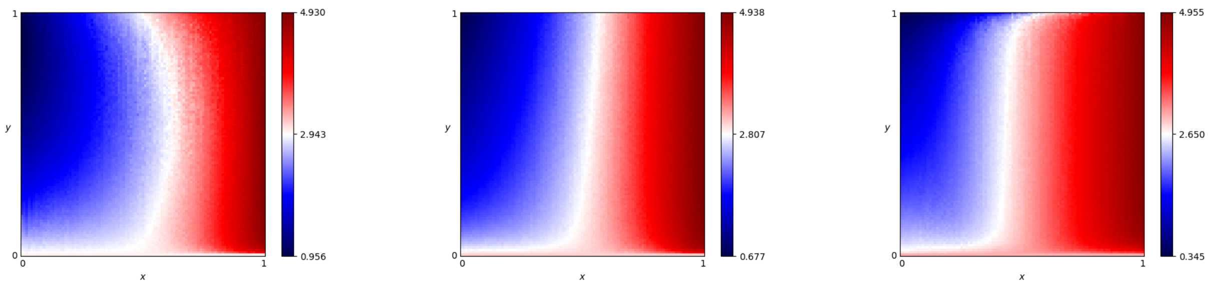


Figure 11: Standard Tournament: rank in each tournament (ranked by median over 49000 tournaments)

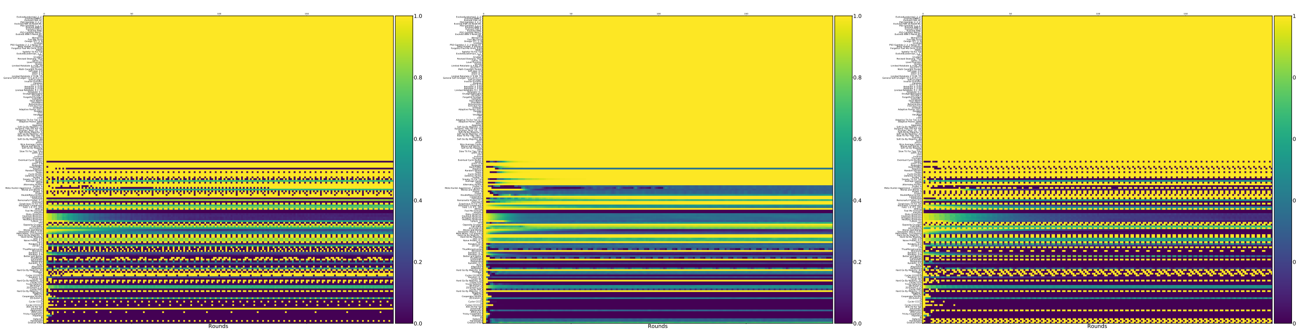
Using a numerical method of fingerprinting based on [4, 6] we can compare strategies. The fingerprints of the top performing standard strategies are shown in Figures 12. There is a striking similarity in the fingerprints.



(a) Fingerprint for EvolvedLookerUp_2_2_2 (b) Fingerprint for Evolved_HMM_5 (c) Fingerprint for Evolved_FSM_16

Figure 12: Comparison of Fingerprints for Standard Tournament Top 3

Figure 13 shows the rate of cooperation in each round for the top three strategies. The opponents in these figures are ordered according to performance by median score. It is evident that the high performing strategies share a common thread against the top strategies: they do not defect first and achieve mutual cooperation. Against the lower strategies they also do not defect first (a mean cooperation rate of 1 in the first round) but do learn to quickly retaliate.



(a) Cooperation rate per round for EvolvedLookerUp_2_2_2 (over 1000 repetitions) (b) Cooperation rate per round for Evolved_HMM_5 (over 1000 repetitions) (c) Cooperation rate per round for Evolved_FSM_16 (over 1000 repetitions)

Figure 13: Comparison of collaboration rates for Standard Tournament Top 3

3.2 Noisy Tournament

Noisy tournaments in which there is a 5% chance that an action is flipped are now described. As shown in Table 6 and Figure 14, the best performing strategies in median payoff are DBS, designed to correct for noise, followed by two strategies trained in the presence of noise and three trained strategies trained without noise. One of the strategies trained with noise (PSO Gambler) actually performs less well than some of the other high ranking strategies including Spiteful TFT (TFT but defects indefinitely if the opponent defects twice consecutively) and OmegaTFT (also designed to handle noise).

	mean	std	min	5%	25%	50%	75%	95%	max
DBS	2.573	0.025	2.474	2.533	2.556	2.573	2.589	2.614	2.675
Evolved ANN 5 Noise 05*	2.534	0.025	2.418	2.492	2.517	2.534	2.551	2.575	2.629
Evolved FSM 16 Noise 05*	2.515	0.031	2.374	2.464	2.494	2.515	2.536	2.565	2.642
Evolved ANN 5*	2.410	0.030	2.273	2.359	2.389	2.410	2.430	2.459	2.536
Evolved FSM 4*	2.393	0.027	2.286	2.348	2.374	2.393	2.411	2.437	2.505
Evolved HMM 5*	2.392	0.026	2.289	2.348	2.374	2.392	2.409	2.435	2.493
Level Punisher	2.388	0.025	2.281	2.347	2.372	2.389	2.405	2.429	2.503
Omega TFT: 3, 8	2.387	0.026	2.270	2.344	2.370	2.388	2.405	2.430	2.498
Spiteful Tit For Tat	2.383	0.030	2.259	2.334	2.363	2.383	2.403	2.432	2.517
Evolved FSM 16*	2.375	0.029	2.239	2.326	2.355	2.375	2.395	2.423	2.507
PSO Gambler 2_2_2 Noise 05*	2.371	0.029	2.250	2.323	2.352	2.371	2.390	2.418	2.480
Adaptive	2.369	0.038	2.217	2.306	2.344	2.369	2.395	2.431	2.524
Evolved ANN*	2.365	0.022	2.270	2.329	2.351	2.366	2.380	2.401	2.483
Math Constant Hunter	2.344	0.022	2.257	2.308	2.329	2.344	2.359	2.382	2.445
Gradual	2.341	0.021	2.248	2.306	2.327	2.341	2.355	2.376	2.429

Table 6: Noisy (5%) Tournament: Mean score per turn of top 15 strategies (ranked by median over 50000 tournaments) * indicates that the strategy was trained.

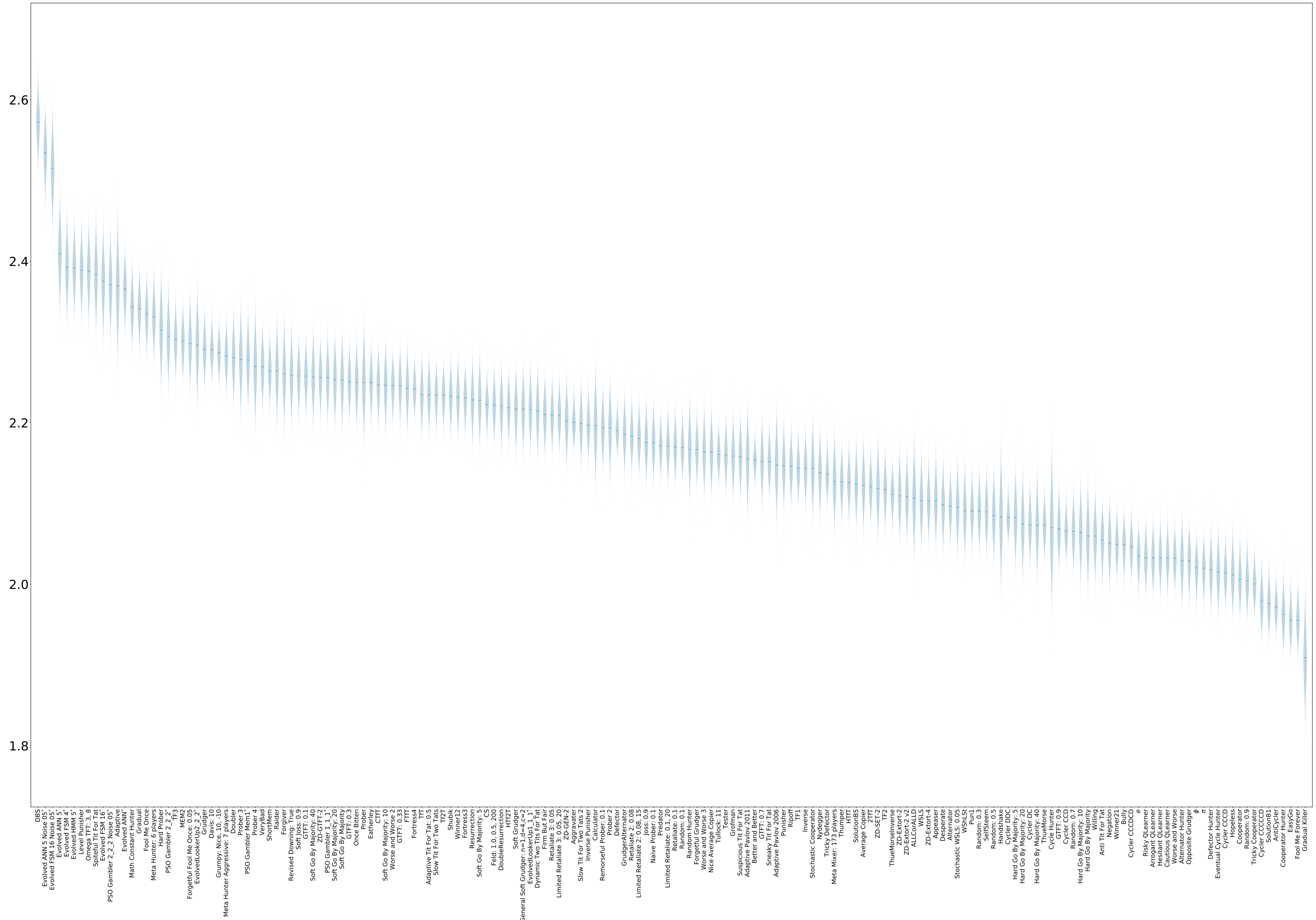


Figure 14: Noisy (5%) Tournament: Mean score per turn (ranked by median over 50000 tournaments)

Recalling Table 2, the strategies trained in the presence of noise are also among the best performers in the absence of noise. As shown in Figure 15 the cluster of mutually cooperative strategies is broken by the noise at 5%. A similar collection of players excels at winning matches but again they have a poor total payoff.

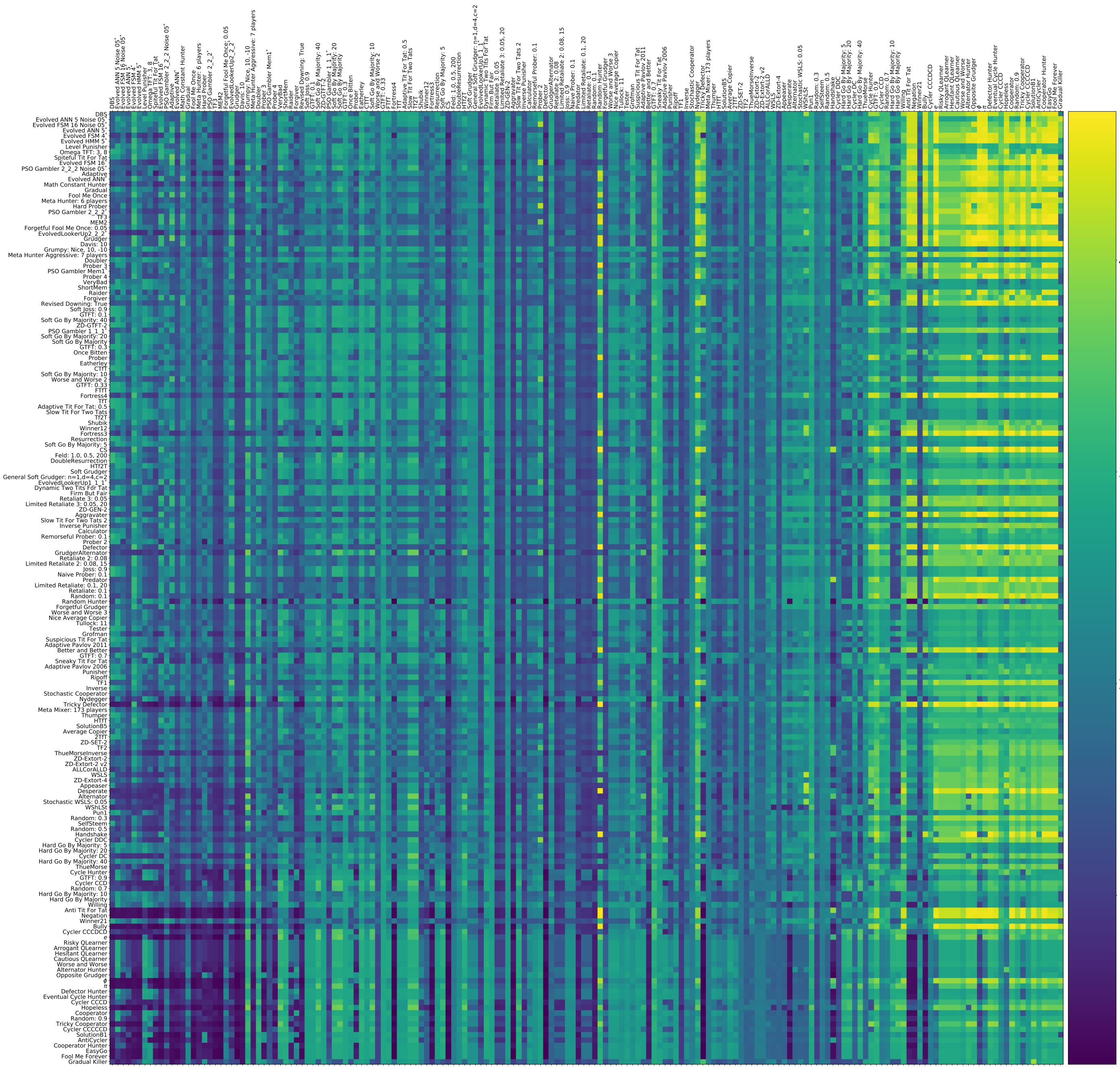


Figure 15: Noisy (5%) Tournament: Mean score per turn of row players against column players (ranked by median over 50000 tournaments)

As shown in Table ?? and Figure 16 the strategies tallying the most wins are somewhat similar, with Defector, the handshaking CollectiveStrategy, and Aggravate appearing as the top three again.

	mean	std	min	5%	25%	50%	75%	95%	max
Aggravater	156.654	3.328	141	151.0	154.0	157.0	159.0	162.0	170
CS	156.875	3.265	144	151.0	155.0	157.0	159.0	162.0	169
Defector	157.324	3.262	144	152.0	155.0	157.0	160.0	163.0	170
Grudger	155.590	3.303	143	150.0	153.0	156.0	158.0	161.0	168
Retaliate 3: 0.05	155.382	3.306	141	150.0	153.0	155.0	158.0	161.0	169
Retaliate 2: 0.08	155.365	3.320	140	150.0	153.0	155.0	158.0	161.0	169
MEM2	155.052	3.349	140	149.0	153.0	155.0	157.0	160.0	169
HTfT	155.298	3.344	141	150.0	153.0	155.0	158.0	161.0	168
Retaliate: 0.1	155.370	3.314	139	150.0	153.0	155.0	158.0	161.0	168
Spiteful Tit For Tat	155.030	3.326	133	150.0	153.0	155.0	157.0	160.0	167
Punisher	153.281	3.375	140	148.0	151.0	153.0	156.0	159.0	167
2TfT	152.823	3.429	138	147.0	151.0	153.0	155.0	158.0	165
TF3	153.031	3.327	138	148.0	151.0	153.0	155.0	158.0	166
Fool Me Once	152.817	3.344	138	147.0	151.0	153.0	155.0	158.0	166
Predator	151.406	3.403	138	146.0	149.0	151.0	154.0	157.0	165

Table 7: Noisy (5%) Tournament: Number of wins per tournament of top 15 strategies (ranked by median wins over 50000 tournaments)

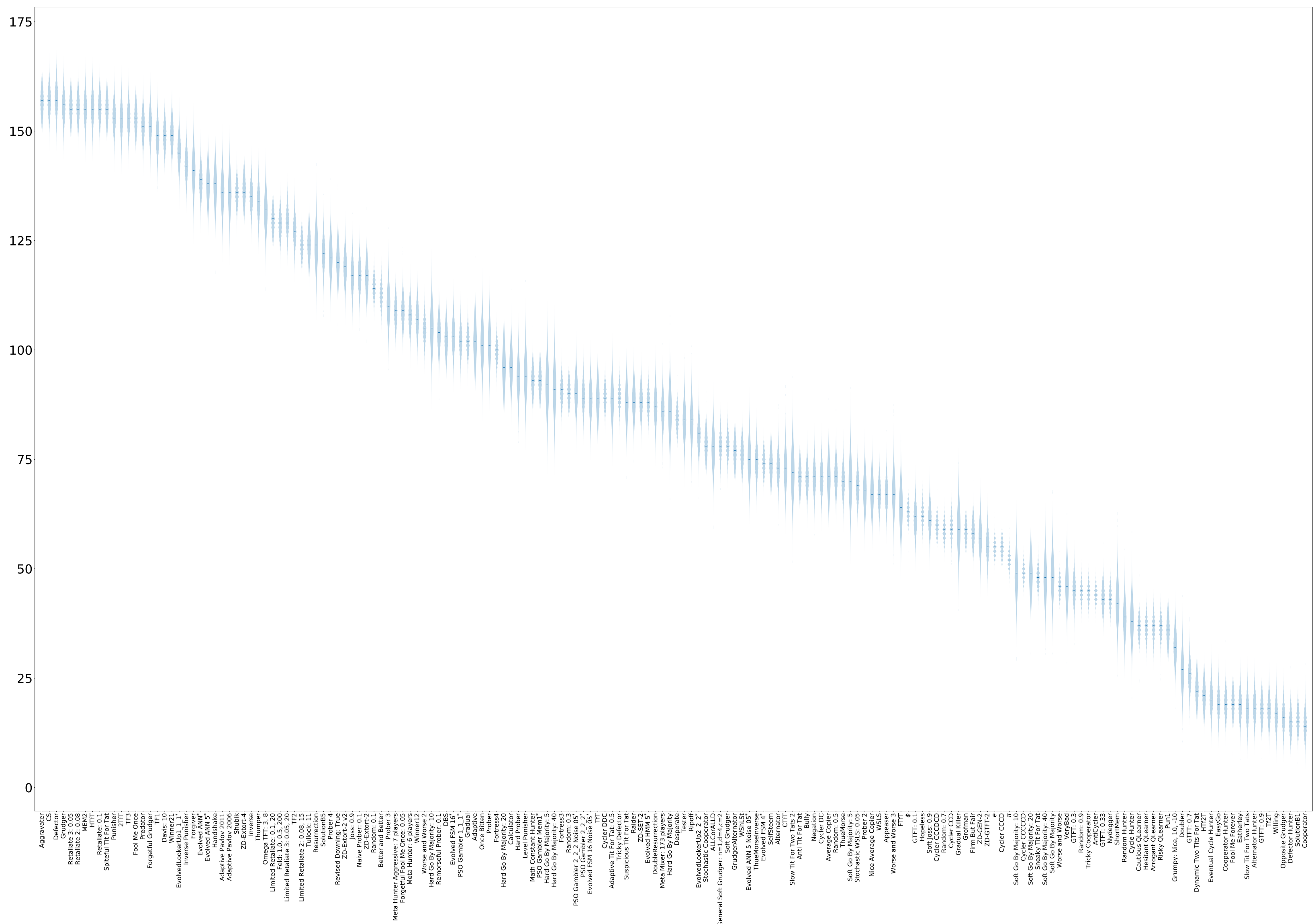


Figure 16: Noisy (5%) Tournament: number of wins per tournament (ranked by median over 50000 tournaments)

As shown in Table 8, the top ranking strategies win a larger number of matches in the presence of noise. For example Spiteful Tit For Tat in one tournament won almost all its matches (167).

	mean	std	min	5%	25%	50%	75%	95%	max
DBS	102.545	3.671	87	97.0	100.0	103.0	105.0	109.0	118
Evolved ANN 5 Noise 05*	75.026	4.226	57	68.0	72.0	75.0	78.0	82.0	93
Evolved FSM 16 Noise 05*	88.699	3.864	74	82.0	86.0	89.0	91.0	95.0	104
Evolved ANN 5*	137.878	4.350	118	131.0	135.0	138.0	141.0	145.0	156
Evolved FSM 4*	74.250	2.694	64	70.0	72.0	74.0	76.0	79.0	85
Evolved HMM 5*	88.189	2.774	77	84.0	86.0	88.0	90.0	93.0	99
Level Punisher	94.263	4.789	75	86.0	91.0	94.0	97.0	102.0	116
Omega TFT: 3, 8	131.655	4.302	112	125.0	129.0	132.0	135.0	139.0	150
Spiteful Tit For Tat	155.030	3.326	133	150.0	153.0	155.0	157.0	160.0	167
Evolved FSM 16*	103.288	3.631	89	97.0	101.0	103.0	106.0	109.0	118
PSO Gambler 2_2_2 Noise 05*	90.515	4.012	75	84.0	88.0	90.0	93.0	97.0	109
Adaptive	101.898	4.899	83	94.0	99.0	102.0	105.0	110.0	124
Evolved ANN*	138.514	3.401	125	133.0	136.0	139.0	141.0	144.0	153
Math Constant Hunter	93.010	3.254	79	88.0	91.0	93.0	95.0	98.0	107
Gradual	101.899	2.870	91	97.0	100.0	102.0	104.0	107.0	114

Table 8: Noisy (5%) Tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 50000 tournaments)

Finally, Table 9 and Figure 17 show the ranks (based on median score) of each strategy over the repeated tournaments. We see that the stochasticity of the ranks understandably increases the DBS strategy never ranks lower than second and wins 75% of the time. The two strategies trained for noisy tournaments rank in the top three 95% of the time.

	mean	std	min	5%	25%	50%	75%	95%	max
DBS	1.205	0.468	1	1.000	1.0	1.0	1.0	2.0	3
Evolved ANN 5 Noise 05*	2.184	0.629	1	1.000	2.0	2.0	3.0	3.0	5
Evolved FSM 16 Noise 05*	2.626	0.618	1	1.000	2.0	3.0	3.0	3.0	9
Evolved ANN 5*	6.371	2.786	2	4.000	4.0	5.0	8.0	12.0	31
Evolved FSM 4*	7.919	3.175	3	4.000	5.0	7.0	10.0	14.0	33
Evolved HMM 5*	7.996	3.110	3	4.000	6.0	7.0	10.0	14.0	26
Level Punisher	8.337	3.083	3	4.000	6.0	8.0	10.0	14.0	26
Omega TFT: 3, 8	8.510	3.249	3	4.000	6.0	8.0	11.0	14.0	32
Spiteful Tit For Tat	9.159	3.772	3	4.000	6.0	9.0	12.0	16.0	40
Evolved FSM 16*	10.218	4.099	3	4.975	7.0	10.0	13.0	17.0	56
PSO Gambler 2_2_2 Noise 05*	10.760	4.102	3	5.000	8.0	10.0	13.0	18.0	47
Evolved ANN*	11.346	3.252	3	6.000	9.0	11.0	13.0	17.0	32
Adaptive	11.420	5.739	3	4.000	7.0	11.0	14.0	21.0	63
Math Constant Hunter	14.668	3.788	3	9.000	12.0	15.0	17.0	21.0	43
Gradual	15.163	3.672	4	10.000	13.0	15.0	17.0	21.0	49

Table 9: Noisy (5%) Tournament: Rank in each tournament of top 15 strategies (ranked by median over 50000 tournaments)

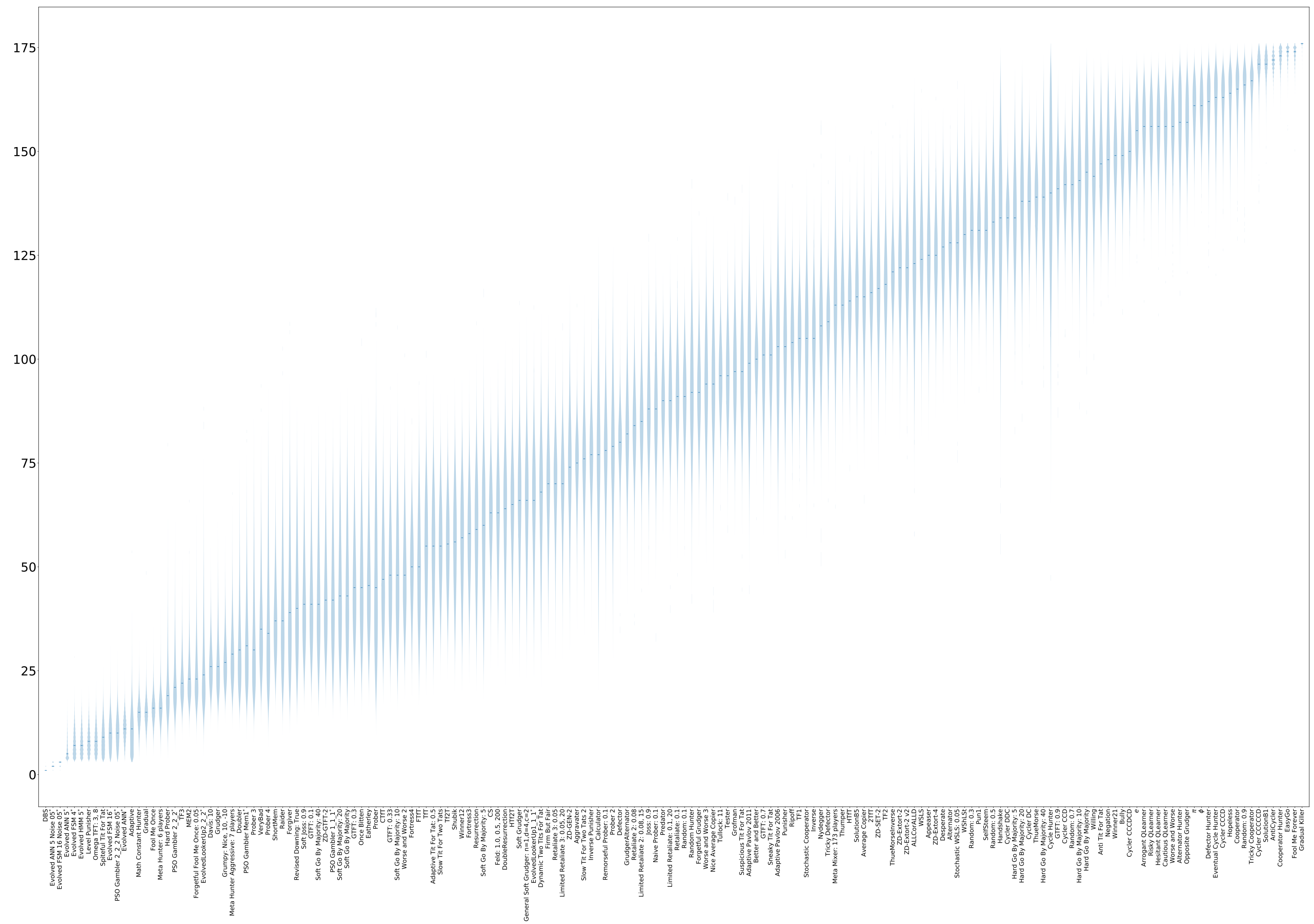
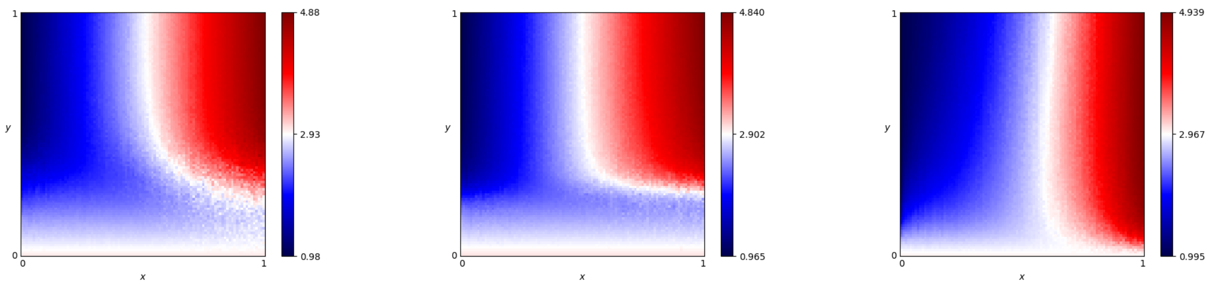


Figure 17: Noisy (5%) Tournament: rank in each tournament (ranked by median over 50000 tournaments)

Similarly to Section 3.1 the top performing noisy strategies (Figure 18) have a striking similarity in the fingerprints which indicates that the strategies may behave similarly in principle.



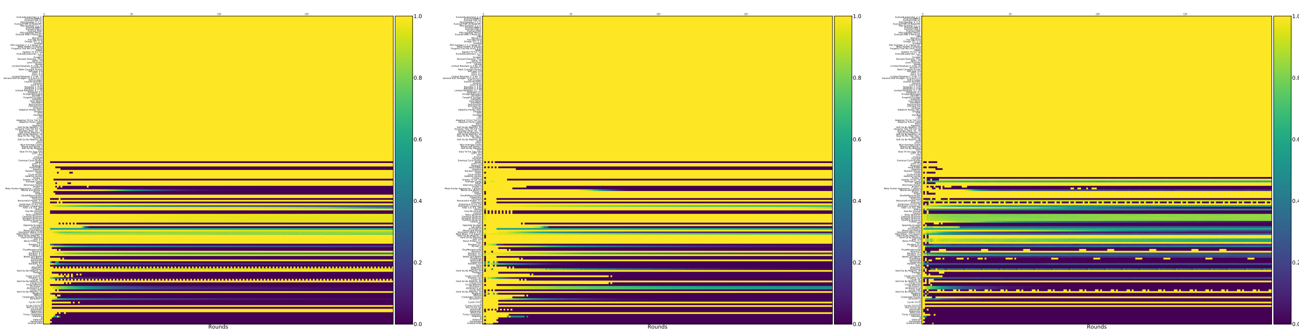
(a) Fingerprint for DBS

(b) Fingerprint for Evolved_ANN_5_Noise_05

(c) Fingerprint for Evolved_FSM_16_Noise_05

Figure 18: Comparison of Fingerprints for Noisy (5%) Tournament Top 3

Figure 19 shows the rate of cooperation in each round for the top three strategies and just as for the top performing strategies in the standard tournament (Figure 13) it is evident that the strategies never defect first and learn to quickly punish poorer strategies.



(a) Cooperation rate per round for DBS (over 1000 repetitions)

(b) Cooperation rate per round for Evolved_ANN_5_Noise_05 (over 1000 repetitions)

(c) Cooperation rate per round for Evolved_FSM_16_Noise_05 (over 1000 repetitions)

Figure 19: Comparison of collaboration rates for Noisy (5%) Tournament Top 3

4 Methods

The trained strategies (denoted by a * in Appendix A were trained using reinforcement learning algorithms. The ideas of reinforcement learning can be attributed to the original work of [54] in which the notion that computers would learn by taking random actions but according to a distribution that picked actions with high rewards more often. The two particular algorithms used here:

- Evolutionary algorithm: [42].
- Particle Swarm Algorithm: [29].

The Particle Swarm Algorithm is implemented using the Python library: `https://pypi.python.org/pypi/pyswarm`. This algorithm was used only to train the Gambler archetype.

All other strategies were trained using evolutionary algorithms. The evolutionary algorithms used standard techniques, varying strategies by mutation and crossover, and evaluating the performance against each opponent for many repetitions. The best performing strategies in each generation are persisted, variants created, and objective functions computed again.

The default parameters for this procedure:

- A population size of 40 individuals (kept constant across the generations);
- A mutation rate of 10%;
- 10 individuals kept from one generation to the next;
- A total of 500 generations.

All implementations of these algorithms are archived at [27]. This software is (similarly to the Axelrod library) available on github `https://github.com/Axelrod-Python/axelrod-dojo`. There are objective functions for:

- total or mean payoff
- total or mean payoff difference (unused in this work)
- total Moran process wins (fixation probability). This lead to the strategies named TF1, TF2, TF3 listed in Appendix A.

These can be used in Noisy or Standard environments (as evidenced by Sections 3.1 and **??**). These objectives can be further modified to suit other purposes. New strategies could be trained with variations including spatial structure, and probabilistically ending matches.

5 Discussion

The tournament results indicate that pre-trained strategies are generally better than human designed strategies at maximizing payoff against a diverse set of opponents. An evolutionary algorithm produces strategies based on multiple standard machine learning techniques that are able to achieve a higher average score than any other known opponent in a standard tournament. Most of the trained strategies use multiple rounds of the history of play (some using all of it) and outperform memory-one strategies (though the trained memory one strategy performs well). The generic structure of the trained strategies did not appear to be critical – strategies based on lookup tables, finite state machines, and stochastic variants all performed well for standard tournaments. Single layer neural networks (Section **??**) performed well in both noisy and standard tournaments though these had some aspect of human involvement in the selection of features. This is in line with the other strategies also where human decisions are made regarding the structure. For the LookerUp and Gambler archetypes (Sections 2.1 and 2.2) a decision has to be made regarding the number of turns of history and initial play that are to be used. Whilst, the Finite state machines and Hidden Markov models (Sections2.4 and **??**) also need input from a human as to the number of states to use they are much more of a black box and thus offer wider adaptability.

A lot of strategies could be represented by many archetypes, however some archetypes will be far more efficient in doing so than others. The fact that the Lookerup strategy does the best for the standard tournament indicates that it represents an efficient reduction of dimension which in turn makes its training more efficient. For the noisy tournament however the dimension reduction represented by some archetypes indicates that sufficient features are not captured by the Lookup tables whilst they are by the neural network and the finite state machine allowing them to adapt to the noisy environment.

In opposition to historical tournament results and community folklore, our results show that complex strategies can be very effective for the IPD. It is not the complexity of strategies that is disadvantageous; rather that directly designing a broadly effective strategy is no easy task. Of all the human-designed strategies in the library, only DBS consistently performs well, and it is substantially more complex than traditional tournament winners like TFT, OmegaTFT, and zero determinant strategies. Furthermore, dealing with noise is difficult for most strategies. Two strategies designed specifically to account for noise, DBS and OmegaTFT, perform well and only DBS performs better than the trained strategies and **only** in the noisy context.

Of the strategies trained to maximize their average score all are generally cooperative, not defecting until the opponent defects. Maximizing for individual performance across a collection of opponents leads to mutual cooperation despite the fact that mutual cooperation is an unstable evolutionary equilibrium for the prisoner's dilemma. Specifically it is noted that the reinforcement learning process for maximizing payoff does not lead to exploitative zero determinant strategies, which may also be a result of the collection of training strategies, of which several retaliate harshly.

We take the liberty of generalizing from the results of this study. For the trained strategies utilizing look up tables we generally found those that incorporate one or more of the initial rounds of play outperformed those that did not. The strategies based on neural networks and finite state machines also are able to condition throughout a match on the first rounds of play. Accordingly, we conclude that first impressions matter in the IPD. The best strategies are nice (never defecting first) and this property could be further investigated with the Axelrod library in future work by e.g. forcing all strategies to defect on the first round.

Finally, we note that as the library grows, the top performing strategies sometimes shuffle, and are not retrained regularly. Most of the strategies were trained on an earlier version of the library (v2.2.0: [53]) that did not include DBS and several other opponents. The precise parameters that are optimal will depend on the pool of opponents. Moreover we have not extensively trained strategies to determine the minimum parameters that are sufficient – neural networks with fewer nodes and features and finite state machines with fewer states may suffice. See [5] for discussion of resource availability for IPD strategies. It may be possible to train strategies more effective in noisy tournaments than DBS.

Future work:

Acknowledgements

This work was performed using the computational facilities of the Advanced Research Computing @ Cardiff (ARCCA) Division, Cardiff University.

References

[1] Christoph Adami and Arend Hintze. "Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything." In: *Nature communications* 4.1 (2013), p. 2193. ISSN: 2041-1723. DOI: 10.1038/ncomms3193. arXiv: arXiv:1208.2666v4. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3741637%7B%5C&%7Dtool=pmcentrez%7B%5C&%7Drendertype=abstract.

[2] Daniel Ashlock. "Training function stacks to play the iterated prisoner's dilemma". In: *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE. 2006, pp. 111–118.

[3] Daniel Ashlock, Joseph Alexander Brown, and Philip Hingston. "Multiple Opponent Optimization of Prisoners Dilemma Playing Agents". In: *IEEE Transactions on Computational Intelligence and AI in Games* 7.1 (2015), pp. 53–65.

[4] Daniel Ashlock and Eun-Youn Kim. "Fingerprinting: Visualization and automatic analysis of prisoner's dilemma strategies". In: *IEEE Transactions on Evolutionary Computation* 12.5 (2008), pp. 647–659.

[5] Daniel Ashlock and Eun-Youn Kim. "The impact of varying resources available to iterated prisoner's dilemma agents". In: *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium on*. IEEE. 2013, pp. 60–67.

[6] Daniel Ashlock, Eun-Youn Kim, and Wendy Ashlock. "Fingerprint analysis of the noisy prisoner's dilemma using a finite-state representation". In: *IEEE Transactions on Computational Intelligence and AI in Games* 1.2 (2009), pp. 154–167.

[7] Wendy Ashlock and Daniel Ashlock. "Changes in Prisoner ' s Dilemma Strategies Over Evolutionary Time With Different Population Sizes". In: (2006), pp. 1001–1008.

[8] Wendy Ashlock and Daniel Ashlock. "Changes in prisoners dilemma strategies over evolutionary time with different population sizes". In: *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE. 2006, pp. 297–304.

[9] Wendy Ashlock and Daniel Ashlock. "Shaped prisoner's dilemma automata". In: *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE. 2014, pp. 1–8.

[10] Wendy Ashlock, Jeffrey Tsang, and Daniel Ashlock. "The evolution of exploitation". In: *Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on*. IEEE. 2014, pp. 135–142.

[11] Tsz-Chiu Au and Dana Nau. "Accident or intention: that is the question (in the Noisy Iterated Prisoner's Dilemma)". In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM. 2006, pp. 561–568.

[12] R. Axelrod. "Effective Choice in the Prisoner's Dilemma". In: *Journal of Conflict Resolution* 24.1 (1980), pp. 3–25.

[13] R. Axelrod. "More Effective Choice in the Prisoner's Dilemma". In: *Journal of Conflict Resolution* 24.3 (1980), pp. 379–403. ISSN: 0022-0027. DOI: 10.1177/002200278002400301.

[14] Robert Axelrod. “Effective choice in the prisoner’s dilemma”. In: *Journal of conflict resolution* 24.1 (1980), pp. 3–25.

[15] Robert M Axelrod. *The evolution of cooperation*. Basic books, 2006.

[16] Jeffrey S Banks and Rangarajan K Sundaram. “Repeated games, finite automata, and complexity”. In: *Games and Economic Behavior* 2.2 (1990), pp. 97–117.

[17] Lee-Ann Barlow and Daniel Ashlock. “Varying decision inputs in Prisoner’s Dilemma”. In: *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2015 IEEE Conference on*. IEEE. 2015, pp. 1–8.

[18] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. “Our meeting with gradual, a good strategy for the iterated prisoners dilemma”. In: *Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*. 1997, pp. 202–209.

[19] Jonathan Bendor, Roderick M Kramer, and Suzanne Stout. “When in doubt . . .: Cooperation in a noisy prisoner’s dilemma”. In: *Journal of Conflict Resolution* 35.4 (1991), pp. 691–719. ISSN: 0022-0027. DOI: 10.1177/0022002791035004007.

[20] Pieter van den Berg and Franz J Weissing. “The importance of mechanisms for the evolution of cooperation”. In: *Proc. R. Soc. B*. Vol. 282. 1813. The Royal Society. 2015, p. 20151382.

[21] Andre LC Carvalho et al. “Iterated Prisoners Dilemma-An extended analysis”. In: (2013).

[22] Eckhart Arnold. *CoopSim v0.9.9 beta 6*. 2015. URL: https://github.com/jecki/CoopSim/.

[23] David B Fogel. “Evolving behaviors in the iterated prisoner’s dilemma”. In: *Evolutionary Computation* 1.1 (1993), pp. 77–97.

[24] Nelis Franken and Andries Petrus Engelbrecht. “Particle swarm optimization approaches to coevolve strategies for the iterated prisoner’s dilemma”. In: *IEEE Transactions on Evolutionary Computation* 9.6 (2005), pp. 562–579.

[25] Marcus R Frean. “The prisoner’s dilemma without synchrony”. In: *Proceedings of the Royal Society of London B: Biological Sciences* 257.1348 (1994), pp. 75–79.

[26] Marco Gaudesi et al. “Exploiting evolutionary modeling to prevail in iterated prisoners dilemma tournaments”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 8.3 (2016), pp. 288–300.

[27] Marc Harper, Vince Knight, and Martin Jones. *Axelrod-Python/axelrod-dojo: v0.0.1*. July 2017. DOI: 10.5281/zenodo.824264. URL: https://doi.org/10.5281/zenodo.824264.

[28] Christian Hilbe, Martin A Nowak, and Arne Traulsen. “Adaptive dynamics of extortion and compliance”. In: *PloS one* 8.11 (2013), e77886.

[29] Muhammad Imran, Rathiah Hashim, and Noor Elaiza Abd Khalid. “An overview of particle swarm optimization variants”. In: *Procedia Engineering* 53 (2013), pp. 491–496.

[30] Graham Kendall, Xin Yao, and Siang Yew Chong. *The iterated prisoners’ dilemma: 20 years on*. Vol. 4. World Scientific, 2007.

[31] Vincent Knight et al. “An Open Framework for the Reproducible Study of the Iterated Prisoners Dilemma”. In: *Journal of Open Research Software* 4.1 (2016).

[32] David Kraines and Vivian Kraines. “Pavlov and the prisoner’s dilemma”. In: *Theory and decision* 26.1 (1989), pp. 47–79.

[33] Steven Kuhn. “Prisoner’s Dilemma”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2017. Metaphysics Research Lab, Stanford University, 2017.

[34] Jiawei Li. “How to design a strategy to win an IPD tournament”. In: *The iterated prisoners dilemma* 20 (2007), pp. 89–104.

[35] Jiawei Li and Graham Kendall. “A strategy with novel evolutionary features for the iterated prisoner’s dilemma.” In: *Evolutionary Computation* 17.2 (2009), pp. 257–274. ISSN: 1063-6560. DOI: 10.1162/evco.2009.17.2.257. URL: http://www.ncbi.nlm.nih.gov/pubmed/19413490.

[36] Jiawei Li, Graham Kendall, and Senior Member. “The effect of memory size on the evolutionary stability of strategies in iterated prisoner ’ s dilemma”. In: X.X (2014), pp. 1–8.

[37] Jiawei Li et al. “Engineering Design of Strategies for Winning Iterated Prisoner ’ s Dilemma Competitions”. In: 3.4 (2011), pp. 348–360.

[38] LIFL. *PRISON*. 2008. URL: http://www.lifl.fr/IPD/ipd.frame.html.

[39] Robert E Marks. “Niche strategies: the Prisoners Dilemma computer tournaments revisited”. In: *JOURNAL OF EVOLUTIONARY ECONOMICS*. Citeseer. 1989.

[40] Philippe Mathieu and Jean-Paul Delahaye. “New Winning Strategies for the Iterated Prisoner’s Dilemma (Extended Abstract)”. In: *14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)* (2015), pp. 1665–1666. ISSN: 15582914.

[41] Shashi Mittal and Kalyanmoy Deb. “Optimal strategies of the iterated prisoner’s dilemma problem for multiple conflicting objectives”. In: *IEEE Transactions on Evolutionary Computation* 13.3 (2009), pp. 554–565. ISSN: 1089778X. DOI: 10.1109/TEVC.2008.2009459.

[42] David E Moriarty, Alan C Schultz, and John J Grefenstette. “Evolutionary algorithms for reinforcement learning”. In: *J. Artif. Intell. Res.(JAIR)* 11 (1999), pp. 241–276.

[43] John H Nachbar. “Evolution in the finitely repeated prisoner’s dilemma”. In: *Journal of Economic Behavior & Organization* 19.3 (1992), pp. 307–326.

[44] M Nowak and K Sigmund. “A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner’s Dilemma game.” In: *Nature* 364.6432 (1993), pp. 56–58. ISSN: 0028-0836. DOI: 10.1038/364056a0.

[45] Martin Nowak and Karl Sigmund. “A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner’s Dilemma game.” In: *Nature* 364.6432 (1993), p. 56.

[46] William H Press and Freeman J Dyson. “Iterated Prisoner’s Dilemma contains strategies that dominate any evolutionary opponent.” In: *Proceedings of the National Academy of Sciences of the United States of America* 109.26 (2012), pp. 10409–13. ISSN: 1091-6490. DOI: 10.1073/pnas.1206569109. URL: http://www.pnas.org/content/109/26/10409.abstract.

[47] Arthur J Robson. “Efficiency in evolutionary games: Darwin, Nash and the secret handshake”. In: *Journal of theoretical Biology* 144.3 (1990), pp. 379–396.

[48] Wolfgang Slany and Wolfgang Kienreich. “On some winning strategies for the Iterated Prisoners Dilemma, or, Mr. Nice Guy and the Cosa Nostra”. In: *The Iterated Prisoners’ Dilemma: 20 Years on* 4 (2007), p. 171.

[49] D W Stephens, C M McLinn, and J R Stevens. “Discounting and reciprocity in an Iterated Prisoner’s Dilemma.” In: *Science (New York, N.Y.)* 298.5601 (2002), pp. 2216–2218. ISSN: 00368075. DOI: 10.1126/science.1078498.

[50] Alexander J. Stewart and Joshua B. Plotkin. “Extortion and cooperation in the Prisoners Dilemma”. In: *Proceedings of the National Academy of Sciences* 109.26 (2012), pp. 10134–10135. DOI: 10.1073/pnas.1208087109. eprint: http://www.pnas.org/content/109/26/10134.full.pdf. URL: http://www.pnas.org/content/109/26/10134.short.

[51] Takahiko Sudo et al. “Effects of ensemble action selection with different usage of player’s memory resource on the evolution of cooperative strategies for iterated prisoner’s dilemma game”. In: *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE. 2015, pp. 1505–1512.

[52] The Axelrod project developers. *Axelrod-Python/Axelrod: v2.13.0*. June 2017. DOI: 10.5281/zenodo.801749. URL: https://doi.org/10.5281/zenodo.801749.

[53] The Axelrod project developers. *Axelrod-Python/Axelrod: v2.2.0*. Dec. 2016. DOI: 10.5281/zenodo.211828. URL: https://doi.org/10.5281/zenodo.211828.

[54] Alan M Turing. “Computing machinery and intelligence”. In: *Mind* 59.236 (1950), pp. 433–460.

[55] E Tzafestas. “Toward adaptive cooperative behavior”. In: *From Animals to animals: Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior (SAB-2000)* 2 (2000), pp. 334–340.

[56] Unkwown. *www.prisoners-dilemma.com*. 2017. URL: http://www.prisoners-dilemma.com/.

[57] Vassilis Vassiliades and Chris Christodoulou. “Multiagent reinforcement learning in the iterated prisoner’s dilemma: fast cooperation through evolved payoffs”. In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE. 2010, pp. 1–8.

[58] Jianzhong Wu and Robert Axelrod. “How to cope with noise in the iterated prisoner’s dilemma”. In: *Journal of Conflict resolution* 39.1 (1995), pp. 183–189.

A List of players

The players used for this study are from Axelrod version 2.13.0 [52].

1. ϕ - *Deterministic* - *Memory depth*: ∞. [52]
2. π - *Deterministic* - *Memory depth*: ∞. [52]
3. e - *Deterministic* - *Memory depth*: ∞. [52]
4. ALLCorALLD - *Stochastic* - *Memory depth*: 1. [52]
5. Adaptive - *Deterministic* - *Memory depth*: ∞. [37]
6. Adaptive Pavlov 2006 - *Deterministic* - *Memory depth*: ∞. [30]
7. Adaptive Pavlov 2011 - *Deterministic* - *Memory depth*: ∞. [37]
8. Adaptive Tit For Tat: 0.5 - *Deterministic* - *Memory depth*: ∞. [55]
9. Aggravater - *Deterministic* - *Memory depth*: ∞. [52]
10. Alternator - *Deterministic* - *Memory depth*: 1. [15, 41]
11. Alternator Hunter - *Deterministic* - *Memory depth*: ∞. [52]
12. Anti Tit For Tat - *Deterministic* - *Memory depth*: 1. [28]
13. AntiCycler - *Deterministic* - *Memory depth*: ∞. [52]
14. Appeaser - *Deterministic* - *Memory depth*: ∞. [52]
15. Arrogant QLearner - *Stochastic* - *Memory depth*: ∞. [52]
16. Average Copier - *Stochastic* - *Memory depth*: ∞. [52]
17. Better and Better - *Stochastic* - *Memory depth*: ∞. [38]
18. Bully - *Deterministic* - *Memory depth*: 1. [43]
19. Calculator - *Stochastic* - *Memory depth*: ∞. [38]
20. Cautious QLearner - *Stochastic* - *Memory depth*: ∞. [52]
21. CollectiveStrategy (**CS**) - *Deterministic* - *Memory depth*: ∞. [35]
22. Contrite Tit For Tat (**CTfT**) - *Deterministic* - *Memory depth*: 3. [58]
23. Cooperator - *Deterministic* - *Memory depth*: 0. [15, 41, 46]
24. Cooperator Hunter - *Deterministic* - *Memory depth*: ∞. [52]
25. Cycle Hunter - *Deterministic* - *Memory depth*: ∞. [52]
26. Cycler CCCCCD - *Deterministic* - *Memory depth*: 5. [52]
27. Cycler CCCD - *Deterministic* - *Memory depth*: 3. [52]
28. Cycler CCCDCD - *Deterministic* - *Memory depth*: 5. [52]
29. Cycler CCD - *Deterministic* - *Memory depth*: 2. [41]
30. Cycler DC - *Deterministic* - *Memory depth*: 1. [52]
31. Cycler DDC - *Deterministic* - *Memory depth*: 2. [41]
32. DBS: 0.75, 3, 4, 3, 5 - *Deterministic* - *Memory depth*: ∞. [11]
33. Davis: 10 - *Deterministic* - *Memory depth*: ∞. [14]
34. Defector - *Deterministic* - *Memory depth*: 0. [15, 41, 46]
35. Defector Hunter - *Deterministic* - *Memory depth*: ∞. [52]
36. Desperate - *Stochastic* - *Memory depth*: 1. [20]
37. DoubleResurrection - *Deterministic* - *Memory depth*: 5. [22]
38. Doubler - *Deterministic* - *Memory depth*: ∞. [38]
39. Dynamic Two Tits For Tat - *Stochastic* - *Memory depth*: 2. [52]
40. EasyGo - *Deterministic* - *Memory depth*: ∞. [37, 38]
41. Eatherley - *Stochastic* - *Memory depth*: ∞. [13]
42. Eventual Cycle Hunter - *Deterministic* - *Memory depth*: ∞. [52]
43. Evolved ANN - *Deterministic* - *Memory depth*: ∞. [52]
44. Evolved ANN 5 - *Deterministic* - *Memory depth*: ∞. [52]
45. Evolved ANN 5 Noise 05 - *Deterministic* - *Memory depth*: ∞. [52]
46. Evolved FSM 16 - *Deterministic* - *Memory depth*: 16. [52]
47. Evolved FSM 16 Noise 05 - *Deterministic* - *Memory depth*: 16. [52]
48. Evolved FSM 4 - *Deterministic* - *Memory depth*: 4. [52]
49. Evolved HMM 5 - *Stochastic* - *Memory depth*: 5. [52]
50. EvolvedLookerUp1_1_1 - *Deterministic* - *Memory depth*: ∞. [52]
51. EvolvedLookerUp2_2_2 - *Deterministic* - *Memory depth*: ∞. [52]
52. Feld: 1.0, 0.5, 200 - *Stochastic* - *Memory depth*: 200. [14]

53. Firm But Fair - *Stochastic* - *Memory depth*: 1. [25]
54. Fool Me Forever - *Deterministic* - *Memory depth*: ∞. [52]
55. Fool Me Once - *Deterministic* - *Memory depth*: ∞. [52]
56. Forgetful Fool Me Once: 0.05 - *Stochastic* - *Memory depth*: ∞. [52]
57. Forgetful Grudger - *Deterministic* - *Memory depth*: 10. [52]
58. Forgiver - *Deterministic* - *Memory depth*: ∞. [52]
59. Forgiving Tit For Tat (**FTfT**) - *Deterministic* - *Memory depth*: ∞. [52]
60. Fortress3 - *Deterministic* - *Memory depth*: 3. [8]
61. Fortress4 - *Deterministic* - *Memory depth*: 4. [8]
62. GTFT: 0.1 - *Stochastic* - *Memory depth*: 1.
63. GTFT: 0.3 - *Stochastic* - *Memory depth*: 1.
64. GTFT: 0.33 - *Stochastic* - *Memory depth*: 1. [26, 44]
65. GTFT: 0.7 - *Stochastic* - *Memory depth*: 1.
66. GTFT: 0.9 - *Stochastic* - *Memory depth*: 1.
67. General Soft Grudger: n=1,d=4,c=2 - *Deterministic* - *Memory depth*: ∞. [52]
68. Gradual - *Deterministic* - *Memory depth*: ∞. [18]
69. Gradual Killer: ('D', 'D', 'D', 'D', 'D', 'C', 'C') - *Deterministic* - *Memory depth*: ∞. [38]
70. Grofman - *Stochastic* - *Memory depth*: ∞. [14]
71. Grudger - *Deterministic* - *Memory depth*: ∞. [14, 16, 18, 20, 37]
72. GrudgerAlternator - *Deterministic* - *Memory depth*: ∞. [38]
73. Grumpy: Nice, 10, -10 - *Deterministic* - *Memory depth*: ∞. [52]
74. Handshake - *Deterministic* - *Memory depth*: ∞. [47]
75. Hard Go By Majority - *Deterministic* - *Memory depth*: ∞. [41]
76. Hard Go By Majority: 10 - *Deterministic* - *Memory depth*: 10. [52]
77. Hard Go By Majority: 20 - *Deterministic* - *Memory depth*: 20. [52]
78. Hard Go By Majority: 40 - *Deterministic* - *Memory depth*: 40. [52]
79. Hard Go By Majority: 5 - *Deterministic* - *Memory depth*: 5. [52]
80. Hard Prober - *Deterministic* - *Memory depth*: ∞. [38]
81. Hard Tit For 2 Tats (**HTf2T**) - *Deterministic* - *Memory depth*: 3. [50]
82. Hard Tit For Tat (**HTfT**) - *Deterministic* - *Memory depth*: 3. [56]
83. Hesitant QLearner - *Stochastic* - *Memory depth*: ∞. [52]
84. Hopeless - *Stochastic* - *Memory depth*: 1. [20]
85. Inverse - *Stochastic* - *Memory depth*: ∞. [52]
86. Inverse Punisher - *Deterministic* - *Memory depth*: ∞. [52]
87. Joss: 0.9 - *Stochastic* - *Memory depth*: 1. [14, 50]
88. Level Punisher - *Deterministic* - *Memory depth*: ∞. [22]
89. Limited Retaliate 2: 0.08, 15 - *Deterministic* - *Memory depth*: ∞. [52]
90. Limited Retaliate 3: 0.05, 20 - *Deterministic* - *Memory depth*: ∞. [52]
91. Limited Retaliate: 0.1, 20 - *Deterministic* - *Memory depth*: ∞. [52]
92. MEM2 - *Deterministic* - *Memory depth*: ∞. [36]
93. Math Constant Hunter - *Deterministic* - *Memory depth*: ∞. [52]
94. Meta Hunter Aggressive: 7 players - *Deterministic* - *Memory depth*: ∞. [52]
95. Meta Hunter: 6 players - *Deterministic* - *Memory depth*: ∞. [52]
96. Meta Mixer: 173 players - *Stochastic* - *Memory depth*: ∞. [52]
97. Naive Prober: 0.1 - *Stochastic* - *Memory depth*: 1. [37]
98. Negation - *Stochastic* - *Memory depth*: 1. [56]
99. Nice Average Copier - *Stochastic* - *Memory depth*: ∞. [52]
100. Nydegger - *Deterministic* - *Memory depth*: 3. [14]
101. Omega TFT: 3, 8 - *Deterministic* - *Memory depth*: ∞. [30]
102. Once Bitten - *Deterministic* - *Memory depth*: 12. [52]
103. Opposite Grudger - *Deterministic* - *Memory depth*: ∞. [52]
104. PSO Gambler 1_1_1 - *Stochastic* - *Memory depth*: ∞. [52]

105. PSO Gambler 2_2_2 - *Stochastic* - *Memory depth*: ∞. [52]

106. PSO Gambler 2_2_2 Noise 05 - *Stochastic* - *Memory depth*: ∞. [52]

107. PSO Gambler Mem1 - *Stochastic* - *Memory depth*: 1. [52]

108. Predator - *Deterministic* - *Memory depth*: 9. [8]

109. Prober - *Deterministic* - *Memory depth*: ∞. [37]

110. Prober 2 - *Deterministic* - *Memory depth*: ∞. [38]

111. Prober 3 - *Deterministic* - *Memory depth*: ∞. [38]

112. Prober 4 - *Deterministic* - *Memory depth*: ∞. [38]

113. Pun1 - *Deterministic* - *Memory depth*: 2. [7]

114. Punisher - *Deterministic* - *Memory depth*: ∞. [52]

115. Raider - *Deterministic* - *Memory depth*: 3. [10]

116. Random Hunter - *Deterministic* - *Memory depth*: ∞. [52]

117. Random: 0.1 - *Stochastic* - *Memory depth*: 0.

118. Random: 0.3 - *Stochastic* - *Memory depth*: 0.

119. Random: 0.5 - *Stochastic* - *Memory depth*: 0. [14, 55]

120. Random: 0.7 - *Stochastic* - *Memory depth*: 0.

121. Random: 0.9 - *Stochastic* - *Memory depth*: 0.

122. Remorseful Prober: 0.1 - *Stochastic* - *Memory depth*: 2. [37]

123. Resurrection - *Deterministic* - *Memory depth*: 5. [22]

124. Retaliate 2: 0.08 - *Deterministic* - *Memory depth*: ∞. [52]

125. Retaliate 3: 0.05 - *Deterministic* - *Memory depth*: ∞. [52]

126. Retaliate: 0.1 - *Deterministic* - *Memory depth*: ∞. [52]

127. Revised Downing: True - *Deterministic* - *Memory depth*: ∞. [14]

128. Ripoff - *Deterministic* - *Memory depth*: 2. [4]

129. Risky QLearner - *Stochastic* - *Memory depth*: ∞. [52]

130. SelfSteem - *Stochastic* - *Memory depth*: ∞. [21]

131. ShortMem - *Deterministic* - *Memory depth*: 10. [21]

132. Shubik - *Deterministic* - *Memory depth*: ∞. [14]

133. Slow Tit For Two Tats - *Deterministic* - *Memory depth*: 2. [52]

134. Slow Tit For Two Tats 2 - *Deterministic* - *Memory depth*: 2. [38]

135. Sneaky Tit For Tat - *Deterministic* - *Memory depth*: ∞. [52]

136. Soft Go By Majority - *Deterministic* - *Memory depth*: ∞. [15, 41]

137. Soft Go By Majority: 10 - *Deterministic* - *Memory depth*: 10. [52]

138. Soft Go By Majority: 20 - *Deterministic* - *Memory depth*: 20. [52]

139. Soft Go By Majority: 40 - *Deterministic* - *Memory depth*: 40. [52]

140. Soft Go By Majority: 5 - *Deterministic* - *Memory depth*: 5. [52]

141. Soft Grudger - *Deterministic* - *Memory depth*: 6. [37]

142. Soft Joss: 0.9 - *Stochastic* - *Memory depth*: 1. [38]

143. SolutionB1 - *Deterministic* - *Memory depth*: 3. [3]

144. SolutionB5 - *Deterministic* - *Memory depth*: 5. [3]

145. Spiteful Tit For Tat - *Deterministic* - *Memory depth*: ∞. [38]

146. Stochastic Cooperator - *Stochastic* - *Memory depth*: 1. [1]

147. Stochastic WSLS: 0.05 - *Stochastic* - *Memory depth*: 1. [52]

148. Suspicious Tit For Tat - *Deterministic* - *Memory depth*: 1. [18, 28]

149. TF1 - *Deterministic* - *Memory depth*: ∞. [52]

150. TF2 - *Deterministic* - *Memory depth*: ∞. [52]

151. TF3 - *Deterministic* - *Memory depth*: ∞. [52]

152. Tester - *Deterministic* - *Memory depth*: ∞. [13]

153. ThueMorse - *Deterministic* - *Memory depth*: ∞. [52]

154. ThueMorseInverse - *Deterministic* - *Memory depth*: ∞. [52]

155. Thumper - *Deterministic* - *Memory depth*: 2. [4]

156. Tit For 2 Tats (**Tf2T**) - *Deterministic* - *Memory depth*: 2. [15]

157. Tit For Tat (**TfT**) - *Deterministic* - *Memory depth*: 1. [14]

158. Tricky Cooperator - *Deterministic* - *Memory depth*: 10. [52]

159. Tricky Defector - *Deterministic* - *Memory depth*: ∞. [52]

160. Tullock: 11 - *Stochastic* - *Memory depth*: 11. [14]
161. Two Tits For Tat (**2TfT**) - *Deterministic* - *Memory depth*: 2. [15]
162. VeryBad - *Deterministic* - *Memory depth*: ∞. [21]
163. Willing - *Stochastic* - *Memory depth*: 1. [20]
164. Win-Shift Lose-Stay: D (**WShLSt**) - *Deterministic* - *Memory depth*: 1. [37]
165. Win-Stay Lose-Shift: C (**WSLS**) - *Deterministic* - *Memory depth*: 1. [32, 44, 50]
166. Winner12 - *Deterministic* - *Memory depth*: 2. [40]
167. Winner21 - *Deterministic* - *Memory depth*: 2. [40]
168. Worse and Worse - *Stochastic* - *Memory depth*: ∞. [38]
169. Worse and Worse 2 - *Stochastic* - *Memory depth*: ∞. [38]
170. Worse and Worse 3 - *Stochastic* - *Memory depth*: ∞. [38]
171. ZD-Extort-2 v2: 0.125, 0.5, 1 - *Stochastic* - *Memory depth*: 1. [33]
172. ZD-Extort-2: 0.1111111111111111, 0.5 - *Stochastic* - *Memory depth*: 1. [50]
173. ZD-Extort-4: 0.23529411764705882, 0.25, 1 - *Stochastic* - *Memory depth*: 1. [52]
174. ZD-GEN-2: 0.125, 0.5, 3 - *Stochastic* - *Memory depth*: 1. [33]
175. ZD-GTFT-2: 0.25, 0.5 - *Stochastic* - *Memory depth*: 1. [50]
176. ZD-SET-2: 0.25, 0.0, 2 - *Stochastic* - *Memory depth*: 1. [33]