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 100 distinct opponents, including many well-known strategies from the literature, and all the trained strategies win standard tournaments against the total collection of other opponents. We also trained variants to win noisy tournaments.

1 Introduction

* Update on Axelrod Library

The Axelrod library [32] now 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 [35], and ZDGTFT2 [50].

There are several previous publications that use evolutionary algorithms to evolve IPD strategies in various circumstances [2, 3, 10, 12, 13, 20, 25, 40, 51, 55]. See also [28] for a strategy trained to win against a collection of well-known IPD opponents and see [26] 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.

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.

The various archetypes will be described in the following sections.

2.1 LookerUp

The first strategy trained with reinforcement learning in the library is based on lookup tables. The strategy 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. Although we tried various combinations of n_1, m_1, and m_2, the best performance at the time of training was obtained for $n_1 = m_1 = m_2 = 2$ and generally for $n_1 > 0$. First impressions matter in the IPD. The library includes a strategy called EvolvedLookerUp2_2_2 which 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 [41], 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 PSO Gambler

PSO 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. The library includes a player 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.

This strategy type can be used to train arbitrary memory-n strategies and a memory one strategy was trained and is called PSO Gambler Mem 1, 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 1) 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: PSO Gamble 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 [26].

2.3 ANN: Single Layer Artificial Neural Network

Strategies based on artificial neural networks can also be trained with an evolutionary algorithm. A variety of features are computed from the history of play such as the opponents trailing moves, the total number of cooperations of the player and the opponent, and several others, which are then input into a feed forward neural network with one layer and user-supplied width. 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 and a stochastic variant that outputs probabilities rather than exact moves could be easily created.

2.4 Finite State Machines

We used strategies based on finite state machines to create a number of strategies. These strategies are deterministic and are efficient computational. At each state the machine transitions to a new state and plays a specified move depending on the opponent's last action. These strategies can encode a variety of other strategies, including classic strategies like TitForTat, encode handshakes, and grudging strategies that always defect after an opponent defection.

2.5 Hidden Markov Models

We also trained stochastic versions of finite state machine players called hidden Markov model players or HMMs. These strategies also encode an internal state with probabilistic transitions to other states and cooperate or defect with various probabilities at each state. These are the best performing stochastic strategies in the library but take longer to train due to their stochasticity.

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 and each is polled for its desired next move. The ensemble then selects the next move based on some 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, these strategies were omitted from the tournament results.

3 Results

3.1 Standard Tournament

We conducted a tournament with all strategies in the Axelrod library, including some parametrized strategies. These are listed in Appendix A. The top 11 performing strategies by median payoff are all strategies trained to maximize total payoff against a subset of the strategies (Table 1). The next strategy is Desired Belief Strategy (DBS) which actively analyzes the opponent 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. 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

	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	2.971
Evolved FSM 16 Noise 05	2.919	0.013	2.874	2.898	2.910	2.919	2.928	2.939	2.964
PSO Gambler 1_1_1	2.912	0.023	2.810	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	2.868	2.889	2.901	2.910	2.918	2.929	2.942
Evolved ANN	2.907	0.010	2.865	2.890	2.900	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	2.943	2.994
Evolved ANN 5 Noise 05	2.864	0.008	2.835	2.850	2.858	2.865	2.870	2.877	2.891
DBS: 0.75, 3, 4, 3, 5	2.857	0.009	2.823	2.843	2.851	2.857	2.863	2.872	2.899
Winner12	2.849	0.008	2.820	2.836	2.843	2.850	2.855	2.862	2.873
Fool Me Once	2.844	0.008	2.819	2.831	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 1: Standard Tournament: Mean score per turn of top 15 strategies (ranked by median over 34000 tournaments)

Violin plots showing the distribution of the scores of each strategy (again ranked by median score) are shown in Figure 1.

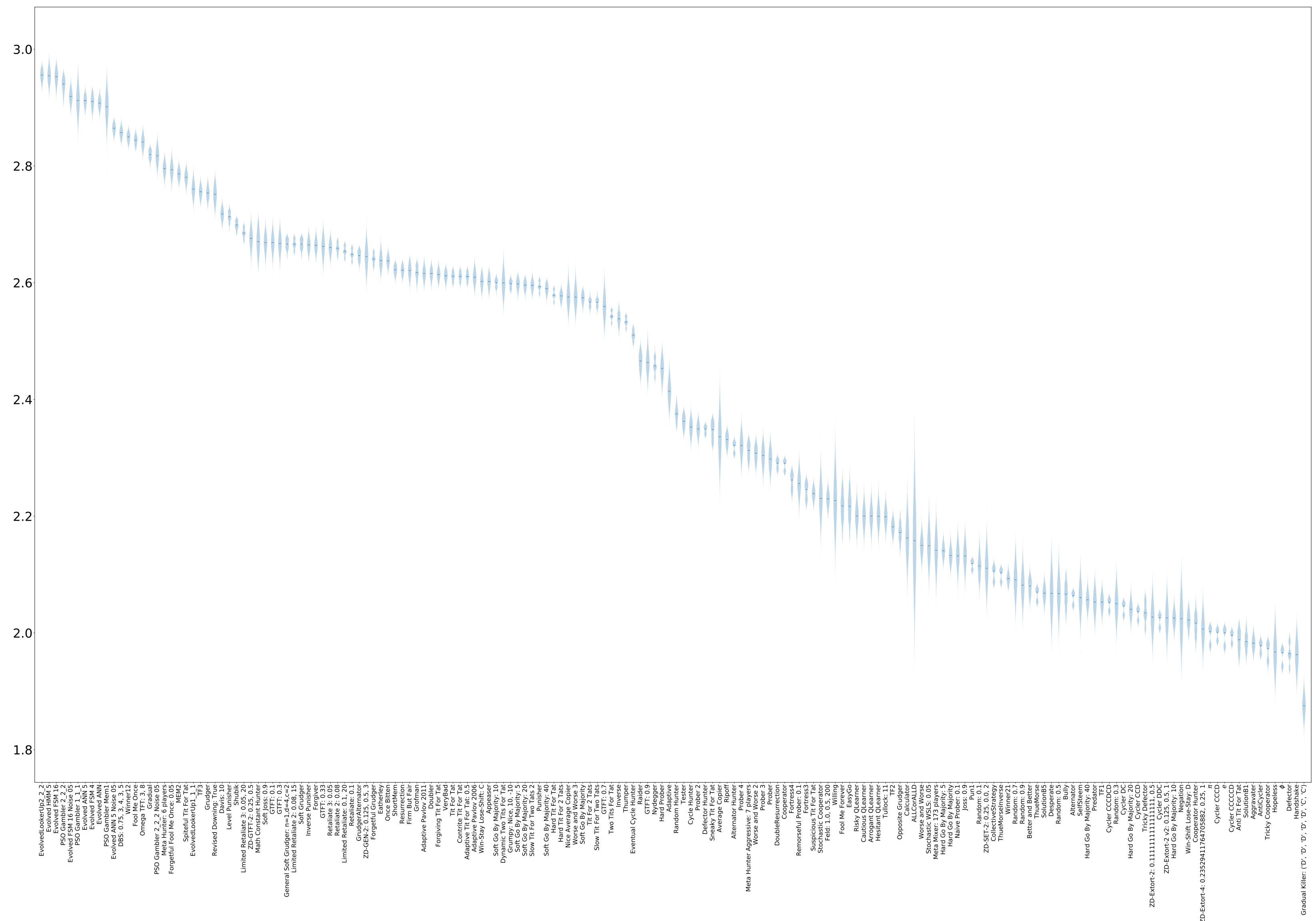


Figure 1: Standard Tournament: Mean score per turn (ranked by median over 34000 tournaments)

Pairwise payoff results are given as a heatmap (Figure 2) which shows that many strategies achieve mutual cooperation with each other and that the top performing strategies never defect first but are able to exploit weaker strategies that attempt to defect.

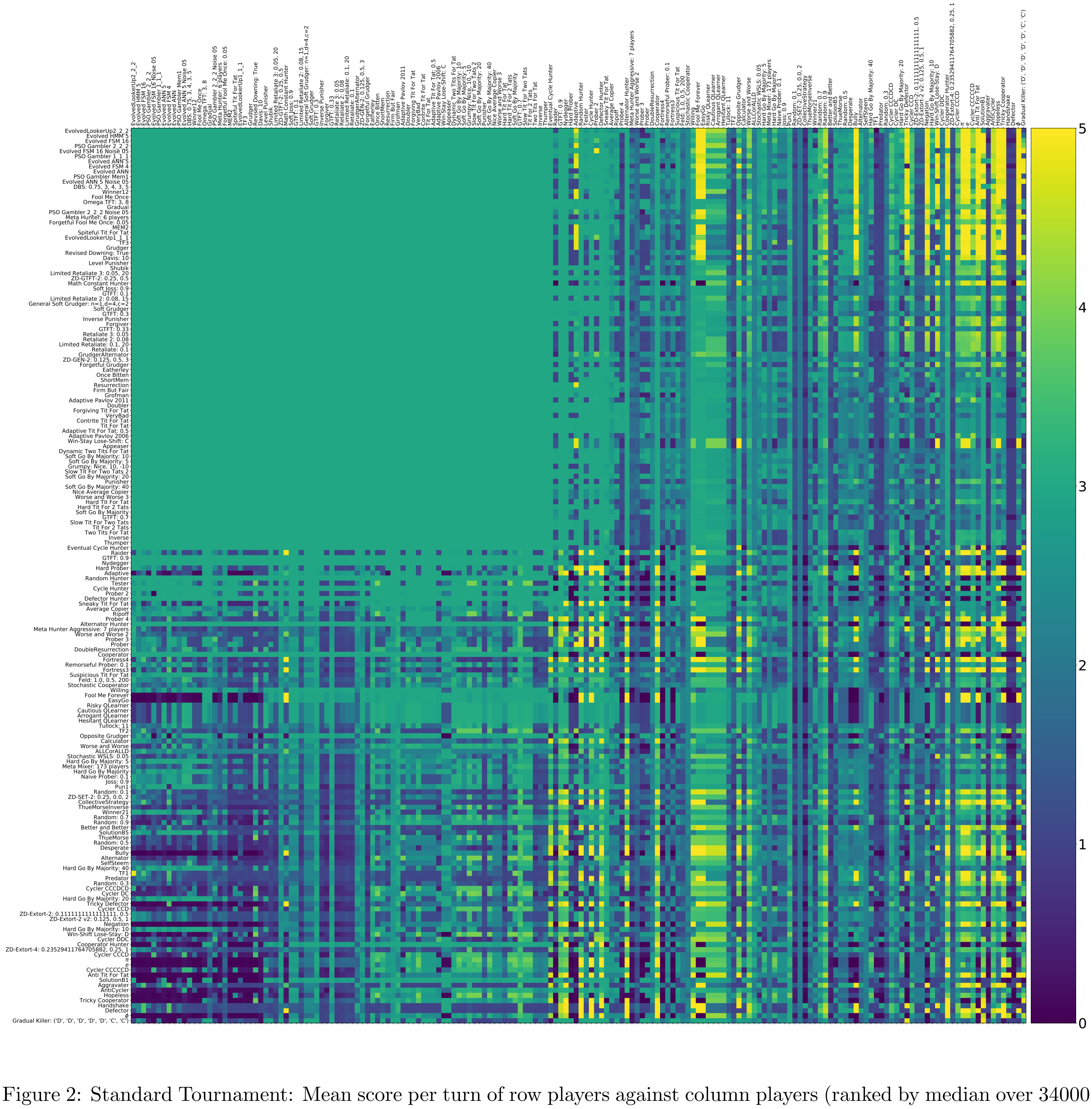


Figure 2: Standard Tournament: Mean score per turn of row players against column players (ranked by median over 34000 tournaments)

The strategies that win the most matches are Defector, Aggravater, followed by handshaking and zero determinant strategies. This includes two handshaking strategies that were the result of training to maximize Moran process fixation. No strategies were trained specifically to win matches. None of the top scoring strategies appear in the top 20 list of strategies ranked by match wins. This can be seen in Figure 3 where the distribution of the number of wins of each strategy is shown.

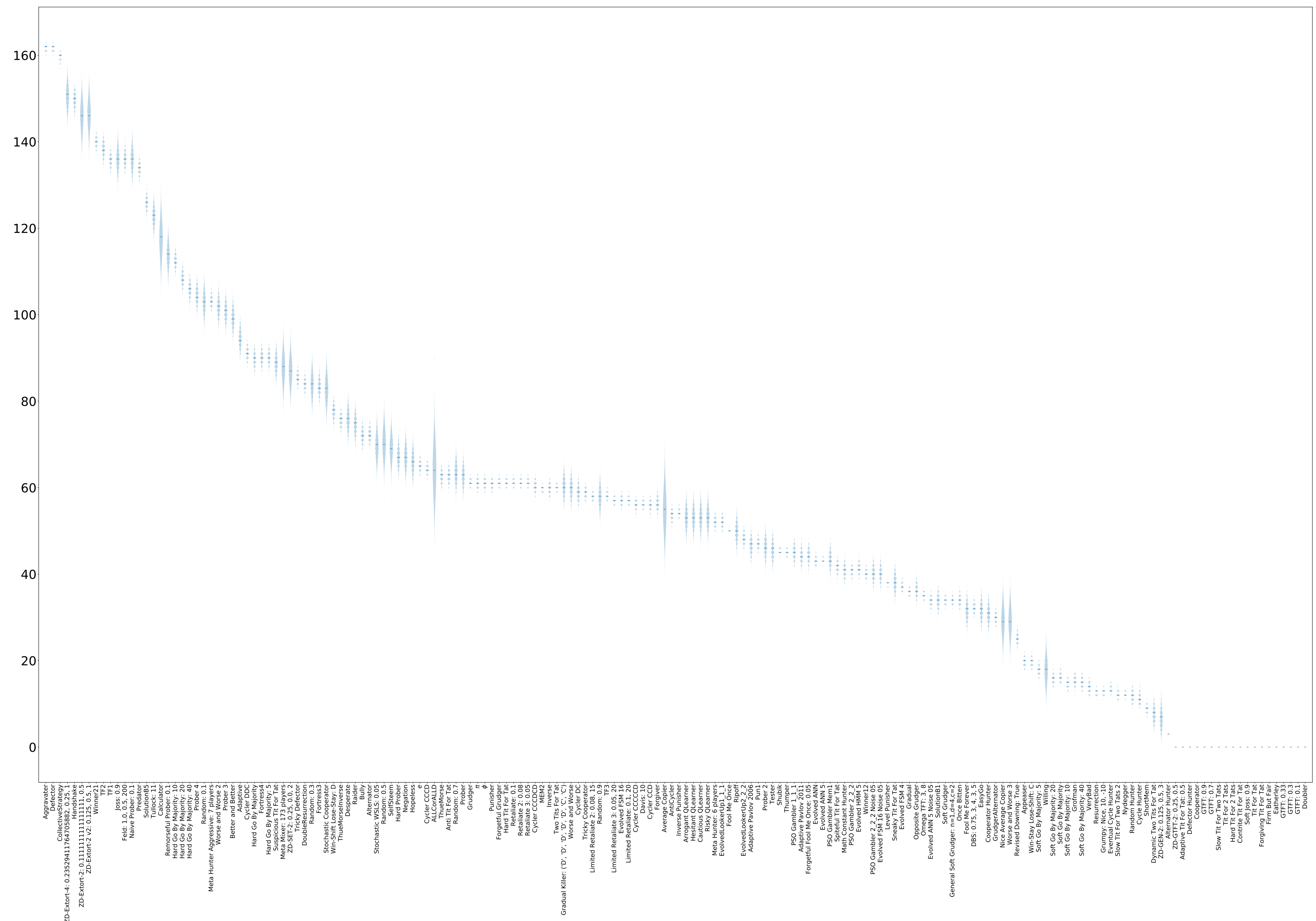


Figure 3: Standard Tournament: number of wins per tournament (ranked by median over 34000 tournaments)

The number of wins of the top strategies of Table 2 are shown in Table 2. 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	48.263	1.337	43	46.0	47.0	48.0	49.0	50.0	53
Evolved HMM 5	41.364	1.221	37	39.0	41.0	41.0	42.0	43.0	45
Evolved FSM 16	56.980	1.099	51	55.0	56.0	57.0	58.0	59.0	60
PSO Gambler 2_2_2	40.686	1.093	36	39.0	40.0	41.0	41.0	42.0	45
Evolved FSM 16 Noise 05	40.079	1.671	34	37.0	39.0	40.0	41.0	43.0	47
PSO Gambler 1_1_1	45.001	1.598	38	42.0	44.0	45.0	46.0	48.0	51
Evolved ANN 5	43.226	0.675	41	42.0	43.0	43.0	44.0	44.0	47
Evolved FSM 4	37.228	0.952	34	36.0	37.0	37.0	38.0	39.0	41
Evolved ANN	43.097	1.021	40	42.0	42.0	43.0	44.0	45.0	48
PSO Gambler Mem1	43.443	1.839	34	40.0	42.0	43.0	45.0	46.0	51
Evolved ANN 5 Noise 05	33.709	1.125	30	32.0	33.0	34.0	34.0	35.0	38
DBS: 0.75, 3, 4, 3, 5	32.327	1.197	28	30.0	32.0	32.0	33.0	34.0	37
Winner12	40.175	1.036	36	39.0	39.0	40.0	41.0	42.0	44
Fool Me Once	50.123	0.424	48	50.0	50.0	50.0	50.0	51.0	52
Omega TFT: 3, 8	35.158	0.862	32	34.0	35.0	35.0	36.0	37.0	39

Table 2: Standard Tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 34000 tournaments)

Finally, Table 3 and Figure 4 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 7th.

	mean	std	min	5%	25%	50%	75%	95%	max
EvolvedLookerUp2_2_2	2.172	1.068	1	1.0	1.0	2.0	3.0	4.0	8
Evolved HMM 5	2.326	1.276	1	1.0	1.0	2.0	3.0	5.0	10
Evolved FSM 16	2.487	1.303	1	1.0	1.0	2.0	3.0	5.0	10
PSO Gambler 2_2_2	3.960	1.523	1	2.0	3.0	4.0	5.0	7.0	10
Evolved FSM 16 Noise 05	6.299	1.686	1	4.0	5.0	6.0	7.0	9.0	11
PSO Gambler 1_1_1	7.092	2.504	1	3.0	5.0	7.0	9.0	10.0	17
Evolved ANN 5	7.284	1.525	2	5.0	6.0	7.0	8.0	10.0	11
Evolved FSM 4	7.529	1.631	2	5.0	6.0	8.0	9.0	10.0	12
Evolved ANN	7.895	1.453	2	5.0	7.0	8.0	9.0	10.0	12
PSO Gambler Mem1	8.223	2.535	1	4.0	6.0	9.0	10.0	12.0	20
Evolved ANN 5 Noise 05	11.364	0.877	8	10.0	11.0	11.0	12.0	13.0	16
DBS: 0.75, 3, 4, 3, 5	12.188	1.120	9	11.0	11.0	12.0	13.0	14.0	16
Winner12	13.223	1.141	9	11.0	12.0	13.0	14.0	15.0	17
Fool Me Once	13.964	1.079	9	12.0	13.0	14.0	15.0	15.0	17
Omega TFT: 3, 8	14.270	1.300	9	12.0	13.0	15.0	15.0	16.0	19

Table 3: Standard Tournament: Rank in each tournament of top 15 strategies (ranked by median over 34000 tournaments)

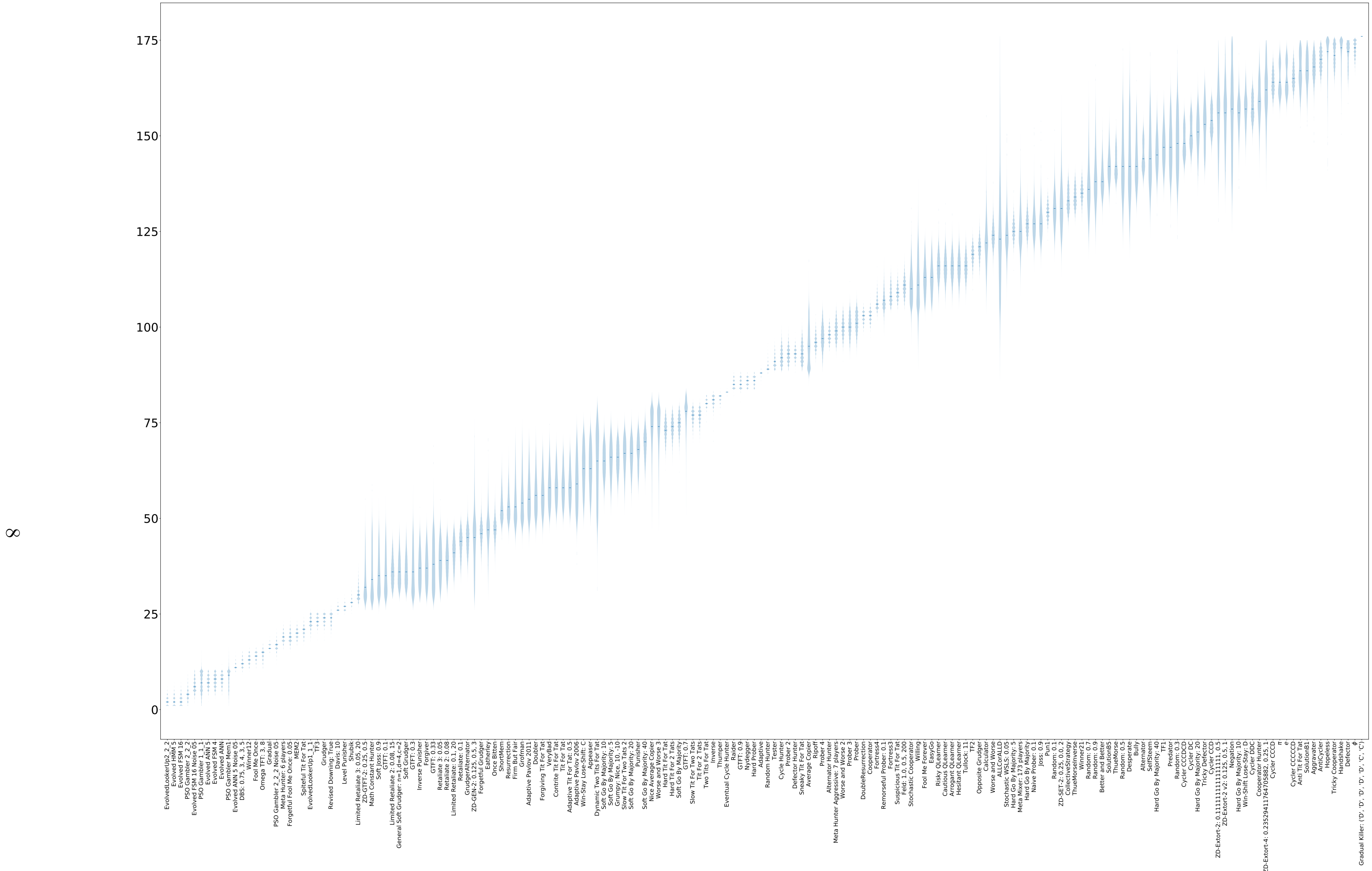
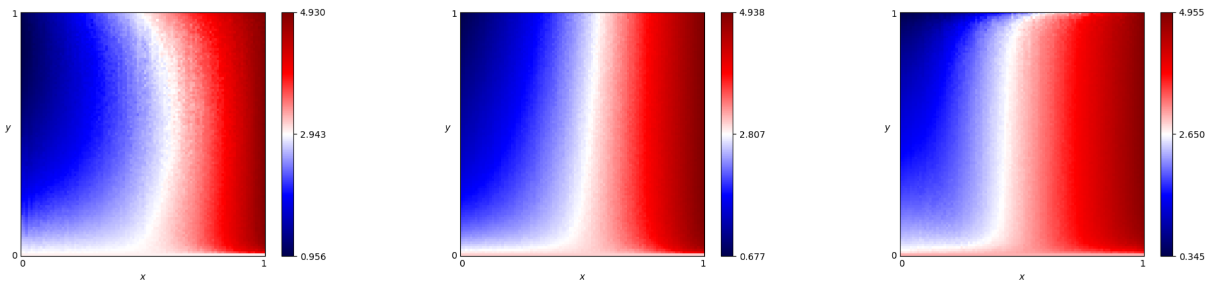


Figure 4: Standard Tournament: rank in each tournament (ranked by median over 34000 tournaments)

Using the method of fingerprinting described in [5] [8], we can compare strategies. For the top performing noisy strategies there is a striking similarity in the fingerprints.



(a) Fingerprint for Evolved-LookerUp_2_2_2

(b) Fingerprint for Evolved_HMM_5

(c) Fingerprint for Evolved_FSM_16

Figure 5: Comparison of Fingerprints for Noisy Tournament Top 3

3.2 Noisy Tournament

We also ran noisy tournaments in which there is a 5% chance that an action is flipped. As shown in Table 4 and Figure 6, 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: 0.75, 3, 4, 3, 5	2.573	0.025	2.474	2.533	2.556	2.573	2.589	2.614	2.667
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	2.292	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	2.496
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	2.487
Omega TFT: 3, 8	2.387	0.026	2.270	2.343	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	0.030	2.245	2.326	2.355	2.376	2.395	2.423	2.494
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	2.432	2.524
Evolved ANN	2.365	0.022	2.276	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	2.432
Gradual	2.341	0.021	2.248	2.306	2.327	2.341	2.355	2.376	2.429

Table 4: Noisy (5%) Tournament: Mean score per turn of top 15 strategies (ranked by median over 33000 tournaments)

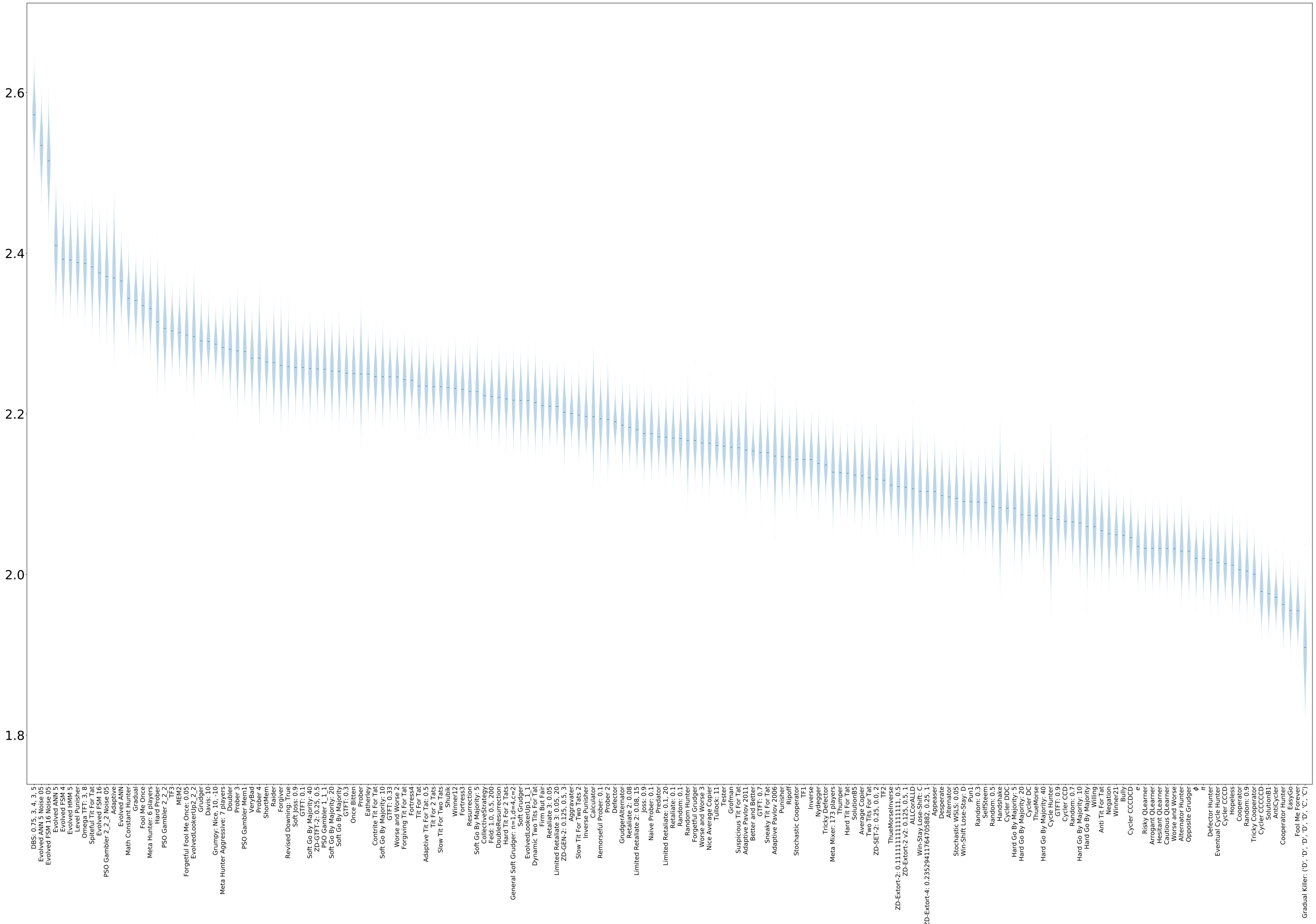


Figure 6: Noisy (5%) Tournament: Mean score per turn (ranked by median over 33000 tournaments)

The strategies trained in the presence of noise are also among the best performers in the absence of noise. As shown in Figure 7 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.

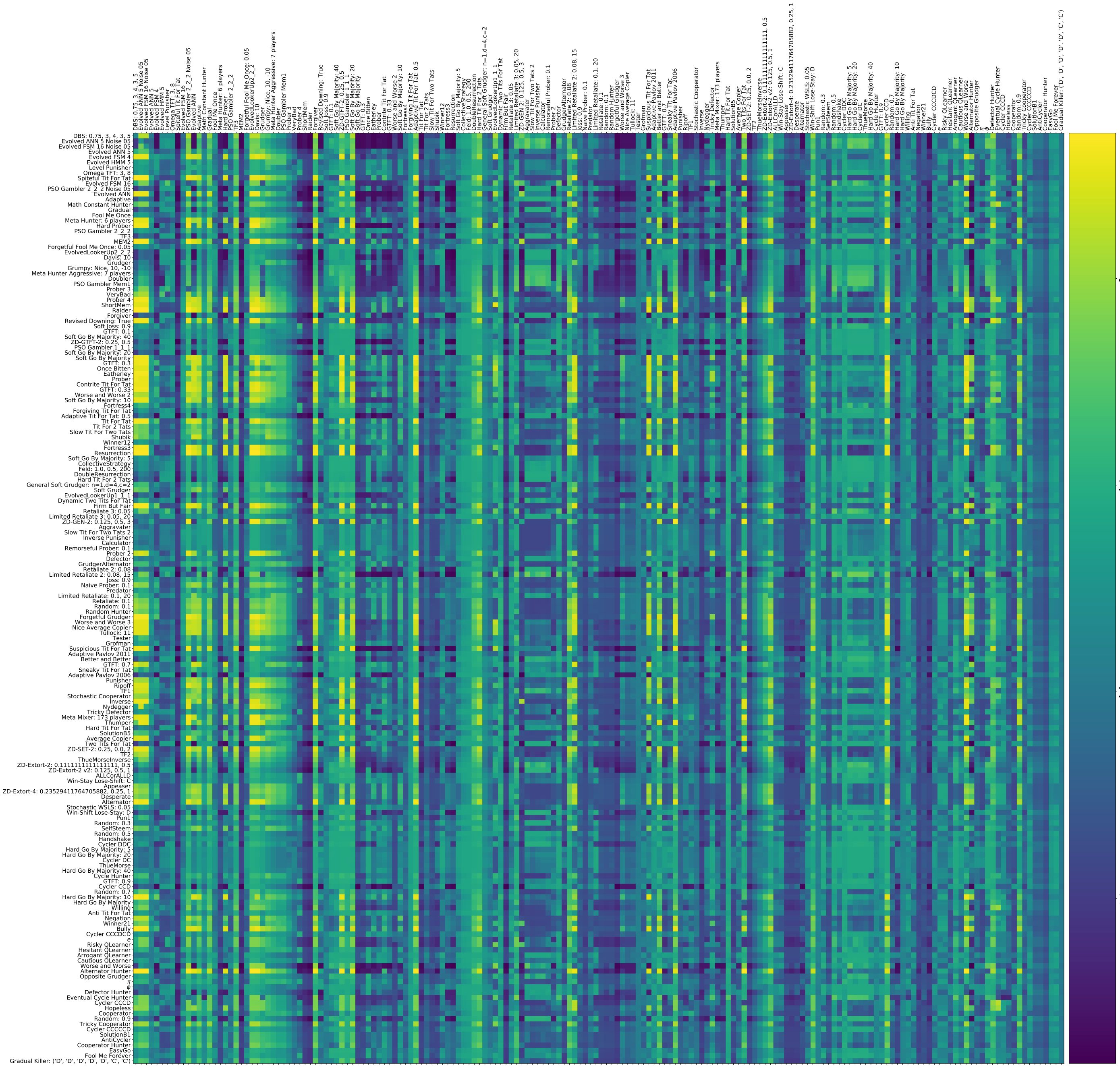


Figure 7: Noisy (5%) Tournament: Mean score per turn of row players against column players (ranked by median over 33000 tournaments)

As shown in Figure 8 the strategies tallying the most wins are somewhat similar, with Defector, the handshaking CollectiveStrategy, and Aggravate appearing as the top three again.

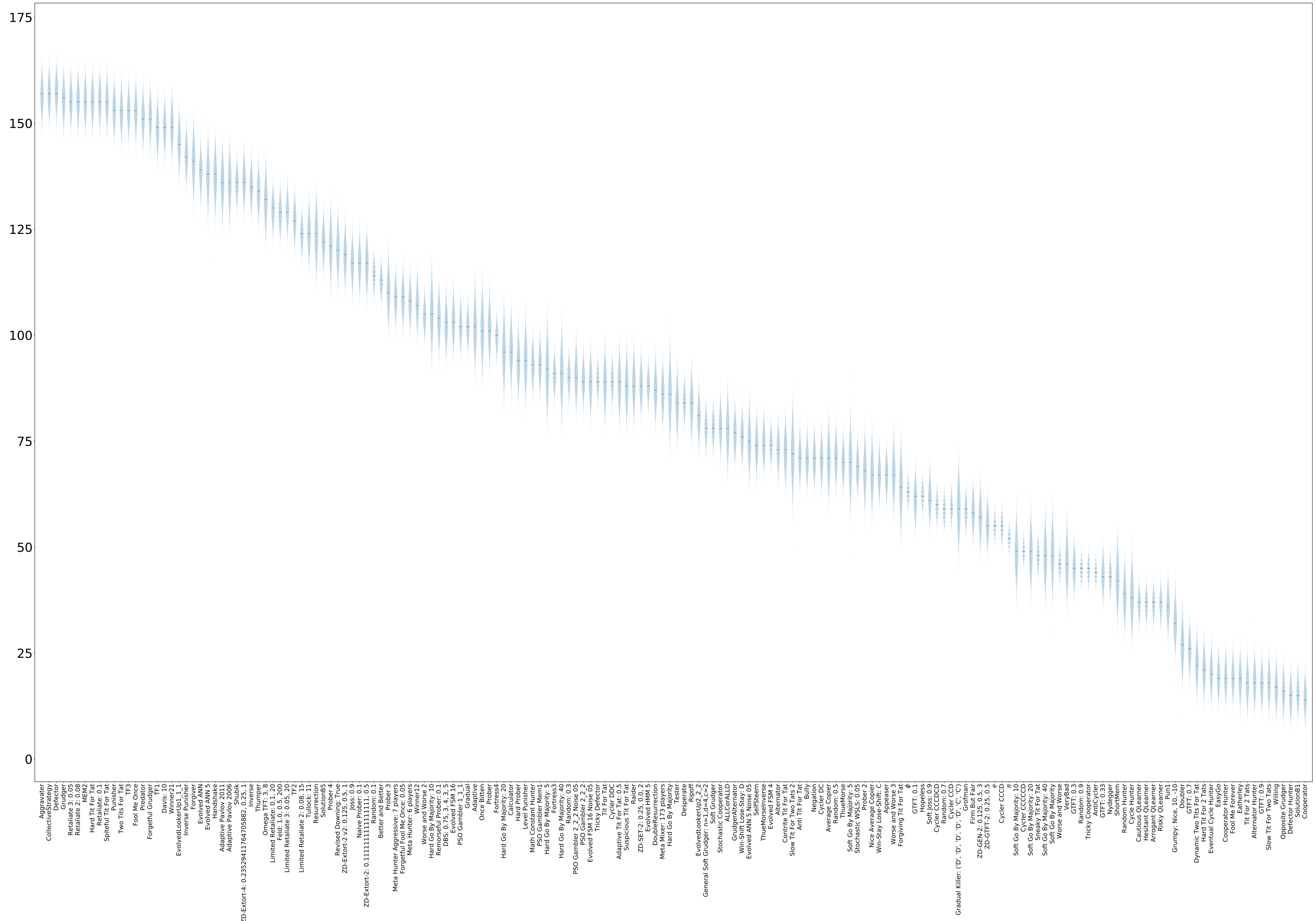


Figure 8: Noisy (5%) Tournament: number of wins per tournament (ranked by median over 33000 tournaments)

As shown in Table 5, 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 it's matches (167).

	mean	std	min	5%	25%	50%	75%	95%	max
DBS: 0.75, 3, 4, 3, 5	102.573	3.678	87	97.0	100.0	103.0	105.0	109.0	118
Evolved ANN 5 Noise 05	75.031	4.228	59	68.0	72.0	75.0	78.0	82.0	93
Evolved FSM 16 Noise 05	88.732	3.873	74	82.0	86.0	89.0	91.0	95.0	104
Evolved ANN 5	137.864	4.359	118	131.0	135.0	138.0	141.0	145.0	156
Evolved FSM 4	74.246	2.681	64	70.0	72.0	74.0	76.0	79.0	85
Evolved HMM 5	88.190	2.777	77	84.0	86.0	88.0	90.0	93.0	99
Level Punisher	94.278	4.771	77	86.0	91.0	94.0	97.0	102.0	116
Omega TFT: 3, 8	131.677	4.301	112	125.0	129.0	132.0	135.0	139.0	150
Spiteful Tit For Tat	155.035	3.329	133	150.0	153.0	155.0	157.0	160.0	167
Evolved FSM 16	103.293	3.641	89	97.0	101.0	103.0	106.0	109.0	118
PSO Gambler 2_2_2 Noise 05	90.516	4.013	75	84.0	88.0	90.0	93.0	97.0	107
Adaptive	101.874	4.902	84	94.0	99.0	102.0	105.0	110.0	122
Evolved ANN	138.511	3.390	125	133.0	136.0	139.0	141.0	144.0	152
Math Constant Hunter	93.007	3.273	79	88.0	91.0	93.0	95.0	98.0	107
Gradual	101.888	2.857	91	97.0	100.0	102.0	104.0	107.0	114

Table 5: Noisy (5%) Tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 33000 tournaments)

Finally, Table 6 and Figure 9 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: 0.75, 3, 4, 3, 5	1.206	0.469	1	1.0	1.0	1.0	1.0	2.0	3
Evolved ANN 5 Noise 05	2.185	0.628	1	1.0	2.0	2.0	3.0	3.0	5
Evolved FSM 16 Noise 05	2.625	0.621	1	1.0	2.0	3.0	3.0	3.0	9
Evolved ANN 5	6.370	2.782	2	4.0	4.0	5.0	8.0	12.0	24
Evolved FSM 4	7.925	3.186	3	4.0	5.0	7.0	10.0	14.0	33
Evolved HMM 5	8.016	3.120	3	4.0	6.0	7.0	10.0	14.0	26
Level Punisher	8.338	3.090	3	4.0	6.0	8.0	10.0	14.0	26
Omega TFT: 3, 8	8.518	3.263	3	4.0	6.0	8.0	11.0	14.0	32
Spiteful Tit For Tat	9.166	3.773	3	4.0	6.0	9.0	12.0	16.0	40
Evolved FSM 16	10.199	4.093	3	4.0	7.0	10.0	13.0	17.0	51
PSO Gambler 2_2_2 Noise 05	10.762	4.099	3	5.0	8.0	10.0	13.0	18.0	47
Evolved ANN	11.343	3.240	3	6.0	9.0	11.0	13.0	17.0	32
Adaptive	11.402	5.743	3	4.0	7.0	11.0	14.0	21.0	63
Math Constant Hunter	14.667	3.802	3	9.0	12.0	15.0	17.0	21.0	43
Gradual	15.167	3.678	4	10.0	13.0	15.0	17.0	21.0	49

Table 6: Noisy (5%) Tournament: Rank in each tournament of top 15 strategies (ranked by median over 33000 tournaments)

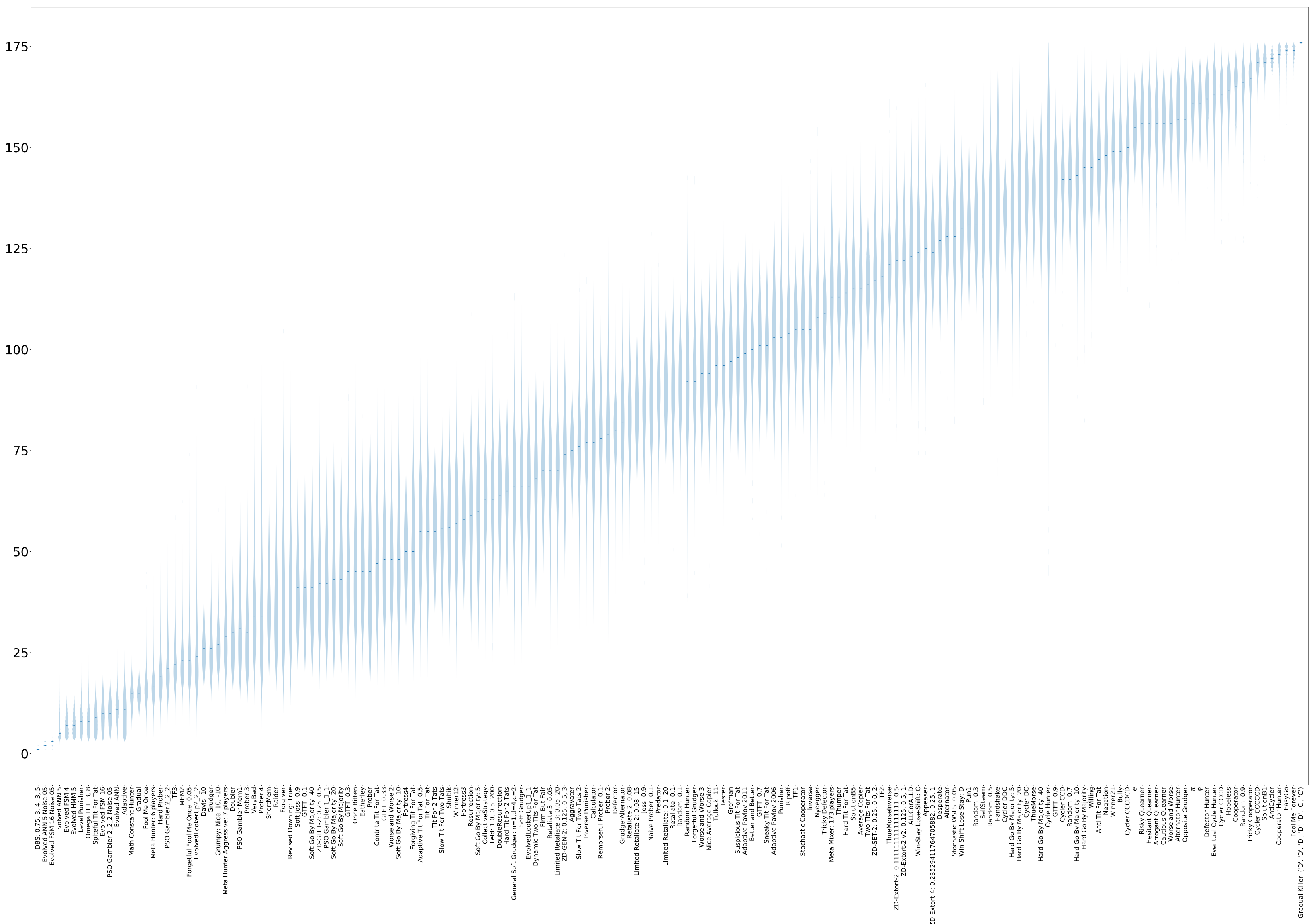
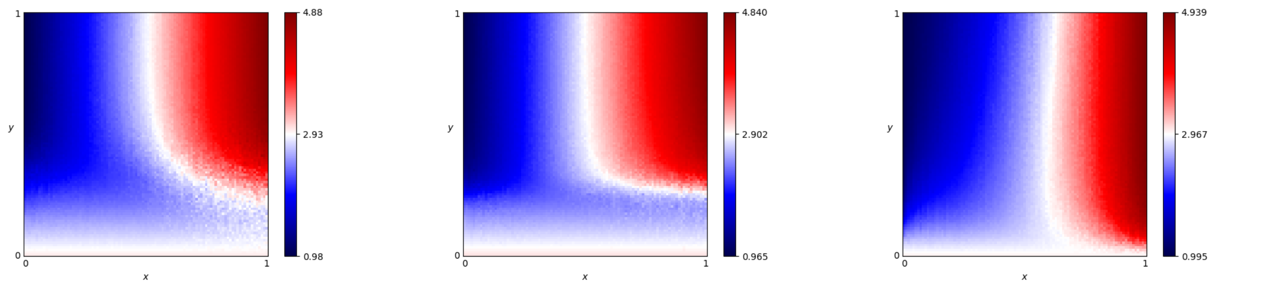


Figure 9: Noisy (5%) Tournament: rank in each tournament (ranked by median over 33000 tournaments)

Using the method of fingerprinting described in [5] [8], we can compare strategies. For the top performing noisy strategies there is a striking similarity in the fingerprints.



(a) Fingerprint for DBS (b) Fingerprint for Evolved_ANN_5_Noise_05 (c) Fingerprint for Evolved_FSM_16_Noise_05

Figure 10: Comparison of Fingerprints for Noisy Tournament Top 3

4 Methods

We trained a variety of strategies using evolutionary algorithms, and in the case of PSO Gambler particle swarm 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. This process continues for approximately 200 generations or until strategies no longer improve significantly.

All training code is available on github. There are objective functions for * total payoff * total payoff difference * total Moran process wins (fixation probability)

New strategies can be easily trained with variations including noise, 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. A simple evolutionary algorithm produces strategies based on multiple standard machine learning techniques that are able to achieve a higher median 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 strategy did not appear to be critical – strategies based on lookup tables, finite state machines, and stochastic variants all performed well. Single layer neural networks also performed well though these had some aspect of human involvement in the selection of features. The success of the other strategy types suggests that a deeper network that incorporates feature engineering would likely also perform well.

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 our trained strategies.

Of the strategies trained to maximize their median 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 equilibrium for the prisoner's dilemma. Specifically we note that the reinforcement learning process for maximizing payout does not lead to exploitative zero determinant strategies, which may be a result of the collection of training strategies, many of which retaliate harshly.

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) 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 [7] for discussion of resource availability for IPD strategies.

Future work: * spatial tournaments and other variants * Additional strategy archetypes by the Ashlocks, e.g. function stacks, binary decision players * further refine features and training parameters

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, 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] 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.

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

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

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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).

[33] David Kraines and Vivian Kraines. "Pavlov and the prisoner's dilemma". In: *Theory and decision* 26.1 (1989), pp. 47–79.

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

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

[36] 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.

[37] 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.

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

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

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

[41] 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.

[42] 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.

[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] 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.

[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.

[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: v2.9.0*. Apr. 2016. DOI: 499122. URL: http://dx.doi.org/10.5281/zenodo.499122.

[53] 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.

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

[55] 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.

[56] 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

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*: ∞. [38]
6. Adaptive Pavlov 2006 - *Deterministic* - *Memory depth*: ∞. [31]
7. Adaptive Pavlov 2011 - *Deterministic* - *Memory depth*: ∞. [38]
8. Adaptive Tit For Tat: 0.5 - *Deterministic* - *Memory depth*: ∞. [53]
9. Aggravater - *Deterministic* - *Memory depth*: ∞. [52]

10. Alternator - *Deterministic* - *Memory depth*: 1. [18, 42]
11. Alternator Hunter - *Deterministic* - *Memory depth*: ∞. [52]
12. Anti Tit For Tat - *Deterministic* - *Memory depth*: 1. [30]
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*: ∞. [39]
18. Bully - *Deterministic* - *Memory depth*: 1. [43]
19. Calculator - *Stochastic* - *Memory depth*: ∞. [39]
20. Cautious QLearner - *Stochastic* - *Memory depth*: ∞. [52]
21. CollectiveStrategy (**CS**) - *Deterministic* - *Memory depth*: ∞. [36]
22. Contrite Tit For Tat (**CTfT**) - *Deterministic* - *Memory depth*: 3. [56]
23. Cooperator - *Deterministic* - *Memory depth*: 0. [18, 42, 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. [42]
30. Cycler DC - *Deterministic* - *Memory depth*: 1. [52]
31. Cycler DDC - *Deterministic* - *Memory depth*: 2. [42]
32. DBS: 0.75, 3, 4, 3, 5 - *Deterministic* - *Memory depth*: ∞. [15]
33. Davis: 10 - *Deterministic* - *Memory depth*: ∞. [17]
34. Defector - *Deterministic* - *Memory depth*: 0. [18, 42, 46]
35. Defector Hunter - *Deterministic* - *Memory depth*: ∞. [52]
36. Desperate - *Stochastic* - *Memory depth*: 1. [22]
37. DoubleResurrection - *Deterministic* - *Memory depth*: 5. [24]
38. Doubler - *Deterministic* - *Memory depth*: ∞. [39]
39. Dynamic Two Tits For Tat - *Stochastic* - *Memory depth*: 2. [52]
40. EasyGo - *Deterministic* - *Memory depth*: ∞. [38, 39]
41. Eatherley - *Stochastic* - *Memory depth*: ∞. [16]
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. [17]
53. Firm But Fair - *Stochastic* - *Memory depth*: 1. [27]
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. [11]

61. Fortress4 - *Deterministic* - *Memory depth*: 4. [11]
62. GTFT: 0.1 - *Stochastic* - *Memory depth*: 1.
63. GTFT: 0.3 - *Stochastic* - *Memory depth*: 1.
64. GTFT: 0.33 - *Stochastic* - *Memory depth*: 1. [29, 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*: ∞. [21]
69. Gradual Killer: ('D', 'D', 'D', 'D', 'D', 'C', 'C') - *Deterministic* - *Memory depth*: ∞. [39]
70. Grofman - *Stochastic* - *Memory depth*: ∞. [17]
71. Grudger - *Deterministic* - *Memory depth*: ∞. [17, 19, 21, 22, 38]
72. GrudgerAlternator - *Deterministic* - *Memory depth*: ∞. [39]
73. Grumpy: Nice, 10, -10 - *Deterministic* - *Memory depth*: ∞. [52]
74. Handshake - *Deterministic* - *Memory depth*: ∞. [47]
75. Hard Go By Majority - *Deterministic* - *Memory depth*: ∞. [42]
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*: ∞. [39]
81. Hard Tit For 2 Tats (**HTf2T**) - *Deterministic* - *Memory depth*: 3. [49]
82. Hard Tit For Tat (**HTfT**) - *Deterministic* - *Memory depth*: 3. [54]
83. Hesitant QLearner - *Stochastic* - *Memory depth*: ∞. [52]
84. Hopeless - *Stochastic* - *Memory depth*: 1. [22]
85. Inverse - *Stochastic* - *Memory depth*: ∞. [52]
86. Inverse Punisher - *Deterministic* - *Memory depth*: ∞. [52]
87. Joss: 0.9 - *Stochastic* - *Memory depth*: 1. [17, 49]
88. Level Punisher - *Deterministic* - *Memory depth*: ∞. [24]
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*: ∞. [37]
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. [38]
98. Negation - *Stochastic* - *Memory depth*: 1. [54]
99. Nice Average Copier - *Stochastic* - *Memory depth*: ∞. [52]
100. Nydegger - *Deterministic* - *Memory depth*: 3. [17]
101. Omega TFT: 3, 8 - *Deterministic* - *Memory depth*: ∞. [31]
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. [11]
109. Prober - *Deterministic* - *Memory depth*: ∞. [38]
110. Prober 2 - *Deterministic* - *Memory depth*: ∞. [39]
111. Prober 3 - *Deterministic* - *Memory depth*: ∞. [39]
112. Prober 4 - *Deterministic* - *Memory depth*: ∞. [39]
113. Pun1 - *Deterministic* - *Memory depth*: 2. [9]

114. Punisher - *Deterministic* - *Memory depth*: ∞. [52]
115. Raider - *Deterministic* - *Memory depth*: 3. [14]
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. [17, 53]
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. [38]
123. Resurrection - *Deterministic* - *Memory depth*: 5. [24]
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*: ∞. [17]
128. Ripoff - *Deterministic* - *Memory depth*: 2. [6]
129. Risky QLearner - *Stochastic* - *Memory depth*: ∞. [52]
130. SelfSteem - *Stochastic* - *Memory depth*: ∞. [23]
131. ShortMem - *Deterministic* - *Memory depth*: 10. [23]
132. Shubik - *Deterministic* - *Memory depth*: ∞. [17]
133. Slow Tit For Two Tats - *Deterministic* - *Memory depth*: 2. [52]
134. Slow Tit For Two Tats 2 - *Deterministic* - *Memory depth*: 2. [39]
135. Sneaky Tit For Tat - *Deterministic* - *Memory depth*: ∞. [52]
136. Soft Go By Majority - *Deterministic* - *Memory depth*: ∞. [18, 42]
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. [38]
142. Soft Joss: 0.9 - *Stochastic* - *Memory depth*: 1. [39]
143. SolutionB1 - *Deterministic* - *Memory depth*: 3. [4]
144. SolutionB5 - *Deterministic* - *Memory depth*: 5. [4]
145. Spiteful Tit For Tat - *Deterministic* - *Memory depth*: ∞. [39]
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. [21, 30]
149. TF1 - *Deterministic* - *Memory depth*: ∞. [52]
150. TF2 - *Deterministic* - *Memory depth*: ∞. [52]
151. TF3 - *Deterministic* - *Memory depth*: ∞. [52]
152. Tester - *Deterministic* - *Memory depth*: ∞. [16]
153. ThueMorse - *Deterministic* - *Memory depth*: ∞. [52]
154. ThueMorseInverse - *Deterministic* - *Memory depth*: ∞. [52]
155. Thumper - *Deterministic* - *Memory depth*: 2. [6]
156. Tit For 2 Tats (**Tf2T**) - *Deterministic* - *Memory depth*: 2. [18]
157. Tit For Tat (**TfT**) - *Deterministic* - *Memory depth*: 1. [17]
158. Tricky Cooperator - *Deterministic* - *Memory depth*: 10. [52]
159. Tricky Defector - *Deterministic* - *Memory depth*: ∞. [52]
160. Tullock: 11 - *Stochastic* - *Memory depth*: 11. [17]
161. Two Tits For Tat (**2TfT**) - *Deterministic* - *Memory depth*: 2. [18]
162. VeryBad - *Deterministic* - *Memory depth*: ∞. [23]
163. Willing - *Stochastic* - *Memory depth*: 1. [22]
164. Win-Shift Lose-Stay: D (**WShLSt**) - *Deterministic* - *Memory depth*: 1. [38]
165. Win-Stay Lose-Shift: C (**WSLS**) - *Deterministic* - *Memory depth*: 1. [33, 44, 49]
166. Winner12 - *Deterministic* - *Memory depth*: 2. [41]
167. Winner21 - *Deterministic* - *Memory depth*: 2. [41]

168. Worse and Worse - *Stochastic* - *Memory depth*: ∞. [39]

169. Worse and Worse 2 - *Stochastic* - *Memory depth*: ∞. [39]

170. Worse and Worse 3 - *Stochastic* - *Memory depth*: ∞. [39]

171. ZD-Extort-2 v2: 0.125, 0.5, 1 - *Stochastic* - *Memory depth*: 1. [34]

172. ZD-Extort-2: 0.1111111111111111, 0.5 - *Stochastic* - *Memory depth*: 1. [49]

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. [34]

175. ZD-GTFT-2: 0.25, 0.5 - *Stochastic* - *Memory depth*: 1. [49]

176. ZD-SET-2: 0.25, 0.0, 2 - *Stochastic* - *Memory depth*: 1. [34]