RESEARCH ARTICLE

Reinforcement learning produces dominant strategies for the Iterated Prisoner’s Dilemma

Marc Harper1,*, Vincent Knight2*, Martin Jones3‡, Georgios Koutsovoulos4‡, Nikoleta E. Glynatsi2‡, Owen Campbell5‡

1 Google Inc., Mountain View, CA, United States of America, 2 Cardiff University, School of Mathematics, Cardiff, United Kingdom, 3 Independent Researcher, Edinburgh, Scotland, 4 INRA, Université Côte d’Azur, CNRS, ISA, Nice, France, 5 Independent Researcher, Chester, United Kingdom

☯ These authors contributed equally to this work.
‡ These authors also contributed equally to this work.
* knightva@cardiff.ac.uk

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 and classic strategies. All the trained strategies win standard tournaments against the total collection of other opponents. The trained strategies and one particular human made designed strategy are the top performers in noisy tournaments also.

Introduction

The Prisoner’s Dilemma (PD) is a two player game used to model a variety of strategic interactions. Each player chooses between cooperation (C) or defection (D). The payoffs of the game are defined by the matrix \( \begin{pmatrix} R & S \\ T & P \end{pmatrix} \), where \( T > R > P > S \) and \( 2R > T + S \). The PD is a one round game, but is commonly studied in a manner where the prior outcomes matter. This repeated form is called the Iterated Prisoner’s Dilemma (IPD). The IPD is frequently used to understand the evolution of cooperative behaviour from complex dynamics [1].

This manuscript uses the Axelrod library [2, 3], open source software for conducting IPD research with reproducibility as a principal goal. Written in the Python programming language, to date 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 all 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 [4] and previous tournament winners such as OmegaTFT [5], Adaptive Pavlov [6], and ZDGTFT2 [7].
Since Robert Axelrod’s seminal tournament [8], a number of IPD tournaments have been undertaken and are summarised in Table 1. Further to the work described in [2] a regular set of standard, noisy [9] and probabilistic ending [10] 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 tournaments with 176 strategies.

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 [13–22]. See also [23] 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 and diverse collection of strategies available from 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.

Materials and methods

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 the results Section. Training will be discussed in a later section. These strategies can encode a variety of other strategies, including classic strategies like Tit For Tat [25], handshake strategies, and grudging strategies, that always defect after an opponent defection.

LookerUp. The LookerUp strategy is based on a lookup table 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 Fig 1.

Training of this strategy corresponds to finding maps from partial histories to actions, 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.

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
<th>Number of Strategies</th>
<th>Type</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>[8]</td>
<td>13</td>
<td>Standard</td>
<td>Not immediately available</td>
</tr>
<tr>
<td>1979</td>
<td>[10]</td>
<td>64</td>
<td>Standard</td>
<td>Available in FORTRAN</td>
</tr>
<tr>
<td>2005</td>
<td>[12]</td>
<td>223</td>
<td>Varied</td>
<td>Not available</td>
</tr>
</tbody>
</table>

https://doi.org/10.1371/journal.pone.0188046.t001
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, known as Grim or Grudger.

Two strategies in the library, Winner12 and Winner21, from [26], 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.

**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 Fig 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 52 of the 64 probabilities being 0 or 1. This strategy type can be used to train arbitrary memory-\(n\) strategies. A memory one strategy called PSOGamblerMem1 was trained, with probabilities $(\Pr(C | CC), \Pr(C | CD), \Pr(C | DC), \Pr(C | DD)) = (1, 0.5217, 0, 0.121)$. Though it performs well in standard tournaments (see Table 2) it does not outperform 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].

**ANN: Single hidden 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

Table 2. Standard tournament: Mean score per turn of top 15 strategies (ranked by median over 50000 tournaments). The leaderboard is dominated by the trained strategies (indicated by a *).

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

https://doi.org/10.1371/journal.pone.0188046.t002
• 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 Fig 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 created.

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. This is illustrated diagrammatically in Fig 4.
Training this strategy corresponds to finding mappings of states and histories to an action and a state. Figs 5 and 6 show two of the trained finite state machines. The layout of state nodes is kept the same between Figs 5 and 6 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.

**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, they 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 Fig 7. Training this strategy corresponds to finding mappings of states and histories to probabilities of cooperating as well as probabilities of the next internal state.
Meta strategies. 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 highest performing 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 knowledge of the match length to defect on the last round(s) of play, not all of these strategies were included in results of this paper. These strategies do not typically outperform the trained strategies described above.
Training 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 \[27\] 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:

- Particle Swarm Algorithm: \[28\].
- Evolutionary algorithm: \[29\].

The Particle Swarm Algorithm is implemented using the pyswarm library: \url{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 [30]. This software is (similarly to the Axelrod library) available on github \url{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.

<table>
<thead>
<tr>
<th>Player</th>
<th>Opponent</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig 7. Diagrammatic representation of the hidden markov model archetype.

\url{https://doi.org/10.1371/journal.pone.0188046.g007}
These can be used in noisy or standard environments. These objectives can be further modified to suit other purposes. New strategies could be trained with variations including spatial structure and probabilistically ending matches.

**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 [23]). 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. A standard tournament with 200 turns and a tournament with 5% noise is discussed. Due to the inherent stochasticity of these IPD tournaments, these tournaments were repeated 50000 times. This allows for a detailed and confident analysis of the performance of strategies. To illustrate the results considered, Fig 8 shows the distribution of the mean score per turn of Tit For Tat over all the repetitions.

![Fig 8. Scores for Tit for Tat over 50000 tournaments.](https://doi.org/10.1371/journal.pone.0188046.g008)
Similarly, Fig 9 shows the ranks of Tit For Tat for each repetition. (We note that it never wins a tournament). Finally Fig 10 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).
The utilities used are \((R, P, T, S) = (3, 1, 5, 0)\) thus the specific Prisoner’s Dilemma being played is:

\[
\begin{pmatrix}
3 & 0 \\
5 & 1
\end{pmatrix}
\] (1)

All data generated for this work is archived and available at [31].

**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) [32], which actively analyzes the opponent and responds accordingly. The next two strategies are Winner12, based on a lookup table, Fool Me Once [3], a grudging strategy that defects indefinitely on the second defection, and Omega Tit For Tat [12].

For completeness, violin plots showing the distribution of the scores of each strategy (again ranked by median score) are shown in Fig 11.

Pairwise payoff results are given as a heatmap (Fig 12) 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.

The strategies that win the most matches (Table 3) are Defector [1] and Aggravater [3], followed by handshaking and zero determinant strategies [33]. 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 Fig 13 where the distribution of the number of wins of each strategy is shown.
The number of wins of the top strategies of Table 2 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 = 34%) matches in a given tournament.

Finally, Table 5 and Fig 14 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 EvolvedLookerUp_2_2_2 ranks in any tournament is 8th.

Figs 15–17 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.

Noisy tournament

Results from noisy tournaments in which there is a 5% chance that an action is flipped are now described. As shown in Table 6 and Fig 18, the best performing strategies in median
payoff are DBS, designed to account 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 Omega TFT (also designed to handle noise). While DBS is the clear winner, it comes at a 6x increased run time over Evolved FSM 16 Noise 05.

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 Fig 19 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.

<table>
<thead>
<tr>
<th>Table 4. Standard tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 50000 tournaments) * indicates that the strategy was trained.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>EvolvedLookerUp2_2_2*</td>
</tr>
<tr>
<td>Evolved HMM 5*</td>
</tr>
<tr>
<td>Evolved FSM 16*</td>
</tr>
<tr>
<td>PSO Gambler 2_2_2*</td>
</tr>
<tr>
<td>Evolved FSM 16 Noise 05*</td>
</tr>
<tr>
<td>PSO Gambler 1_1_1*</td>
</tr>
<tr>
<td>Evolved ANN 5*</td>
</tr>
<tr>
<td>Evolved FSM 4*</td>
</tr>
<tr>
<td>Evolved ANN*</td>
</tr>
<tr>
<td>PSO Gambler Mem1*</td>
</tr>
<tr>
<td>Evolved ANN 5 Noise 05*</td>
</tr>
<tr>
<td>DBS</td>
</tr>
<tr>
<td>Winner12</td>
</tr>
<tr>
<td>Fool Me Once</td>
</tr>
<tr>
<td>Omega TFT: 3, 8</td>
</tr>
</tbody>
</table>

https://doi.org/10.1371/journal.pone.0188046.t004

<table>
<thead>
<tr>
<th>Table 5. Standard tournament: Rank in each tournament of top 15 strategies (ranked by median over 50000 tournaments) * indicates that the strategy was trained.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>EvolvedLookerUp2_2_2*</td>
</tr>
<tr>
<td>Evolved HMM 5*</td>
</tr>
<tr>
<td>Evolved FSM 16*</td>
</tr>
<tr>
<td>PSO Gambler 2_2_2*</td>
</tr>
<tr>
<td>Evolved FSM 16 Noise 05*</td>
</tr>
<tr>
<td>PSO Gambler 1_1_1*</td>
</tr>
<tr>
<td>Evolved ANN 5*</td>
</tr>
<tr>
<td>Evolved FSM 4*</td>
</tr>
<tr>
<td>Evolved ANN*</td>
</tr>
<tr>
<td>PSO Gambler Mem1*</td>
</tr>
<tr>
<td>Evolved ANN 5 Noise 05*</td>
</tr>
<tr>
<td>DBS</td>
</tr>
<tr>
<td>Winner12</td>
</tr>
<tr>
<td>Omega TFT: 3, 8</td>
</tr>
</tbody>
</table>

https://doi.org/10.1371/journal.pone.0188046.t005
As shown in Table 7 and Fig 20 the strategies tallying the most wins are somewhat similar to the standard tournaments, with Defector, the handshaking CollectiveStrategy [34], and Aggravater appearing as the top three again.

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 [35] in one tournament won almost all its matches (167).

Finally, Table 9 and Fig 21 show the ranks (based on median score) of each strategy over the repeated tournaments. We see that the stochasticity of the ranks understandably increases relative to the standard tournament. An exception is the top three strategies, for example, 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.

Figs 22–24 shows the rate of cooperation in each round for the top three strategies (in the absence of noise) and just as for the top performing strategies in the standard tournament it is evident that the strategies never defect first and learn to quickly punish poorer strategies.

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 generic archetypes 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 from the literature. Interestingly, a trained memory one strategy produced by a particle swarm algorithm performs well, better than human designed strategies.
such as Win Stay Lose Shift and zero determinant strategies (which enforce a payoff difference rather than maximize total payoff).

In opposition to historical tournament results and community folklore, our results show that complex strategies can be effective for the IPD. 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.

The generic structure of the trained strategies did not appear to be critical for the standard tournament—strategies based on lookup tables, finite state machines, neural networks, and stochastic variants all performed well. Single layer neural networks 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 some human decisions are made regarding the structure. For the LookerUp and Gambler archetypes a decision has to be made regarding the number of rounds of history and initial play that are to be used. In contrast, the finite state machines and hidden Markov models required only a choice of the number of states, and the training algorithm can eliminate unneeded states in the case of finite state
machines (evidenced by the unconnected nodes in the diagrams for the included representations).

Many strategies can be represented by multiple archetypes, however some archetypes will be more efficient in encoding the patterns present in the data. 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. In particular the first rounds of play were valuable bits of information. For the noisy tournament however the dimension reduction represented by some archetypes indicates that some features of the data are not captured by the lookup tables while they are by the neural networks and the finite state machines, allowing the latter to adapt better to the noisy environment. Intuitively, a noisy environment can significantly affect a lookup table based on the last two rounds of play since these action pairs compete with probing defections, apologies, and retaliations. Accordingly, it is not surprising that additional parameter space is needed to adapt to a noisy environment.

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 noisy contexts.
Empirically we find that DBS (with its default parameters) does not win tournaments at 1% noise. However DBS has a parameter that accounts for the expected amount of noise and a followup study with various noise levels could make a more complete study of the performance of DBS and strategies trained at various noise levels.

The strategies trained to maximize their average score are generally cooperative and do not defect first. 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. Training with the objective of maximizing payoff difference may produce strategies more like zero determinant strategies.

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

Fig 17. Cooperation rates for Evolved_FSM_16 (strategies ordered by median score over 10000 tournaments).

https://doi.org/10.1371/journal.pone.0188046.g017
the first rounds of play. Accordingly, we conclude that first impressions matter in the IPD. The best strategies are nice (never defecting first) and the impact of the first rounds of play could be further investigated with the Axelrod library in future work by e.g. forcing all strategies to defect on the first round.

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.

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

Fig 18. Noisy (5%) tournament: Mean score per turn (strategies ordered by median score over 50000 tournaments).
We note that as the library grows, the top performing strategies sometimes shuffle, and are not retrained automatically. Most of the strategies were trained on an earlier version of the library (v2.2.0: [36]) 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 parameter spaces that are sufficient—neural networks with fewer nodes and features and finite state machines with fewer states may suffice. See [37] for discussion of resource availability for IPD strategies.

Finally, whilst we have considered the robustness of our claims and results with respect to noise it would also be of interest to train strategies for different versions of the stage game (also referred to as dilemma strength) [38, 39]. Our findings seems to indicate that obtaining strong strategies for other games through reinforcement learning would be possible.
Table 7. Noisy (5%) tournament: Number of wins per tournament of top 15 strategies (ranked by median wins over 50000 tournaments).

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

A Appendix A: List of players

The players used for this study are from Axelrod version 2.13.0 [3].

1. $\phi$—Deterministic—Memory depth: $\infty$, [3]
2. $\pi$—Deterministic—Memory depth: $\infty$, [3]
Table 8. Noisy (5%) tournament: Number of wins per tournament of top 15 strategies (ranked by median score over 50000 tournaments) * indicates that the strategy was trained.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>95%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS</td>
<td>102.545</td>
<td>3.671</td>
<td>87</td>
<td>97.0</td>
<td>100.0</td>
<td>103.0</td>
<td>105.0</td>
<td>109.0</td>
<td>118</td>
</tr>
<tr>
<td>Evolved ANN 5 Noise 05*</td>
<td>75.026</td>
<td>4.226</td>
<td>57</td>
<td>68.0</td>
<td>72.0</td>
<td>75.0</td>
<td>78.0</td>
<td>82.0</td>
<td>93</td>
</tr>
<tr>
<td>Evolved FSM 16 Noise 05*</td>
<td>88.699</td>
<td>3.864</td>
<td>74</td>
<td>82.0</td>
<td>86.0</td>
<td>89.0</td>
<td>91.0</td>
<td>95.0</td>
<td>104</td>
</tr>
<tr>
<td>Evolved ANN 5*</td>
<td>137.878</td>
<td>4.350</td>
<td>118</td>
<td>131.0</td>
<td>135.0</td>
<td>138.0</td>
<td>141.0</td>
<td>145.0</td>
<td>156</td>
</tr>
<tr>
<td>Evolved FSM 4*</td>
<td>74.250</td>
<td>2.694</td>
<td>64</td>
<td>70.0</td>
<td>72.0</td>
<td>74.0</td>
<td>76.0</td>
<td>79.0</td>
<td>85</td>
</tr>
<tr>
<td>Evolved HMM 5*</td>
<td>88.189</td>
<td>2.774</td>
<td>77</td>
<td>84.0</td>
<td>86.0</td>
<td>88.0</td>
<td>90.0</td>
<td>93.0</td>
<td>99</td>
</tr>
<tr>
<td>Level Punisher</td>
<td>94.263</td>
<td>4.789</td>
<td>75</td>
<td>86.0</td>
<td>91.0</td>
<td>94.0</td>
<td>97.0</td>
<td>102.0</td>
<td>116</td>
</tr>
<tr>
<td>Omega TFT: 3, 8</td>
<td>131.655</td>
<td>4.302</td>
<td>112</td>
<td>125.0</td>
<td>129.0</td>
<td>132.0</td>
<td>135.0</td>
<td>139.0</td>
<td>150</td>
</tr>
<tr>
<td>Spiteful Tit For Tat</td>
<td>155.030</td>
<td>3.326</td>
<td>89</td>
<td>97.0</td>
<td>101.0</td>
<td>103.0</td>
<td>106.0</td>
<td>109.0</td>
<td>118</td>
</tr>
<tr>
<td>Evolved FSM 16*</td>
<td>103.288</td>
<td>3.631</td>
<td>89</td>
<td>97.0</td>
<td>101.0</td>
<td>103.0</td>
<td>106.0</td>
<td>109.0</td>
<td>118</td>
</tr>
<tr>
<td>PSO Gambler 2_2_2 Noise 05*</td>
<td>90.515</td>
<td>4.012</td>
<td>75</td>
<td>84.0</td>
<td>88.0</td>
<td>90.0</td>
<td>93.0</td>
<td>97.0</td>
<td>109</td>
</tr>
<tr>
<td>Adaptive</td>
<td>101.898</td>
<td>4.899</td>
<td>83</td>
<td>94.0</td>
<td>99.0</td>
<td>102.0</td>
<td>105.0</td>
<td>110.0</td>
<td>124</td>
</tr>
<tr>
<td>Evolved ANN*</td>
<td>138.514</td>
<td>3.401</td>
<td>2</td>
<td>4.000</td>
<td>4.0</td>
<td>5.0</td>
<td>8.0</td>
<td>12.0</td>
<td>31</td>
</tr>
<tr>
<td>Math Constant Hunter</td>
<td>93.010</td>
<td>3.254</td>
<td>79</td>
<td>88.0</td>
<td>91.0</td>
<td>93.0</td>
<td>95.0</td>
<td>98.0</td>
<td>107</td>
</tr>
<tr>
<td>Gradual</td>
<td>101.899</td>
<td>2.870</td>
<td>91</td>
<td>97.0</td>
<td>100.0</td>
<td>102.0</td>
<td>104.0</td>
<td>107.0</td>
<td>114</td>
</tr>
</tbody>
</table>

Table 9. Noisy (5%) tournament: Rank in each tournament of top 15 strategies (ranked by median over 50000 tournaments) * indicates that the strategy was trained.

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

https://doi.org/10.1371/journal.pone.0188046.t008

3. $e$—Deterministic—Memory depth: $\infty$. [3]
4. ALLCorALLD—Stochastic—Memory depth: 1. [3]
5. Adaptive—Deterministic—Memory depth: $\infty$. [43]
7. Adaptive Pavlov 2011—Deterministic—Memory depth: $\infty$. [43]
8. Adaptive Tit For Tat: 0.5—Deterministic—Memory depth: $\infty$. [44]
10. Alternator—Deterministic—Memory depth: 1. [1, 45]
12. Anti Tit For Tat—Deterministic—Memory depth: 1. [46]
15. Arrogant QLearner—Stochastic—Memory depth: ∞. [3]
21. CollectiveStrategy (CS)—Deterministic—Memory depth: ∞. [34]
22. Contrite Tit For Tat (CTFT)—Deterministic—Memory depth: 3. [48]
23. Cooperator—Deterministic—Memory depth: 0. [1, 33, 45]
27. Cycler CCCD—Deterministic—Memory depth: 3. [3]
29. Cycler CCD—Deterministic—Memory depth: 2. [45]
31. Cycler DDC—Deterministic—Memory depth: 2. [45]
32. DBS: 0.75, 3, 4, 3, 5—Deterministic—Memory depth: ∞. [32]
33. Davis: 10—Deterministic—Memory depth: ∞. [25]
34. Defector—Deterministic—Memory depth: 0. [1, 33, 45]
36. Desperate—Stochastic—Memory depth: 1. [49]
37. DoubleResurrection—Deterministic—Memory depth: 5. [50]
Fig 23. Cooperation rates for Evolved_ANN_5_Noise_05 (strategies ordered by median score over 10000 tournaments).

https://doi.org/10.1371/journal.pone.0188046.g023

38. Doubler—Deterministic—Memory depth: ∞. [35]
39. Dynamic Two Tits For Tat—Stochastic—Memory depth: 2. [3]
40. EasyGo—Deterministic—Memory depth: ∞. [35, 43]
41. Eatherley—Stochastic—Memory depth: ∞. [10]
42. Eventual Cycle Hunter—Deterministic—Memory depth: ∞. [3]
43. Evolved ANN—Deterministic—Memory depth: ∞. [3]
44. Evolved ANN 5—Deterministic—Memory depth: ∞. [3]
45. Evolved ANN 5 Noise 05—Deterministic—Memory depth: ∞. [3]
Fig 24. Cooperation rates for Evolved_FSM_16_Noise_05 (strategies ordered by median score over 10000 tournaments).

https://doi.org/10.1371/journal.pone.0188046.g024

49. Evolved HMM 5—Stochastic—Memory depth: 5. [3]
50. EvolvedLookerUp1_1_1—Deterministic—Memory depth: ∞. [3]
51. EvolvedLookerUp2_2_2—Deterministic—Memory depth: ∞. [3]
52. Feld: 1.0, 0.5, 200—Stochastic—Memory depth: 200. [25]
53. Firm But Fair—Stochastic—Memory depth: 1. [51]
55. Fool Me Once—Deterministic—Memory depth: ∞. [3]
56. Forgetful Fool Me Once: 0.05—Stochastic—Memory depth: ∞. [3]
57. Forgetful Grudger—Deterministic—Memory depth: 10. [3]
58. Forgiver—Deterministic—Memory depth: ∞. [3]
59. Forgiving Tit For Tat (FTFT)—Deterministic—Memory depth: ∞. [3]
Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma

60. Fortress3—Deterministic—Memory depth: 3. [14]
62. GTFT: 0.1—Stochastic—Memory depth: 1.
63. GTFT: 0.3—Stochastic—Memory depth: 1.
64. GTFT: 0.33—Stochastic—Memory depth: 1. [23, 52]
65. GTFT: 0.7—Stochastic—Memory depth: 1.
66. GTFT: 0.9—Stochastic—Memory depth: 1.
68. Gradual—Deterministic—Memory depth: ∞. [53]
70. Grofman—Stochastic—Memory depth: ∞. [25]
71. Grudger—Deterministic—Memory depth: 1. [25, 43, 49, 53, 54]
72. GrudgerAlternator—Deterministic—Memory depth: ∞. [35]
73. Grumpy: Nice, 10, −10—Deterministic—Memory depth: ∞. [3]
74. Handshake—Deterministic—Memory depth: ∞. [55]
75. Hard Go By Majority—Deterministic—Memory depth: ∞. [45]
78. Hard Go By Majority: 40—Deterministic—Memory depth: 40. [3]
79. Hard Go By Majority: 5—Deterministic—Memory depth: 5. [3]
80. Hard Prober—Deterministic—Memory depth: ∞. [35]
81. Hard Tit For 2 Tats (HTf2T)—Deterministic—Memory depth: 3. [7]
82. Hard Tit For Tat (HTfT)—Deterministic—Memory depth: 3. [56]
83. Hesitant QLearner—Stochastic—Memory depth: ∞. [3]
84. Hopeless—Stochastic—Memory depth: 1. [49]
85. Inverse—Stochastic—Memory depth: ∞. [3]
86. Inverse Punisher—Deterministic—Memory depth: ∞. [3]
87. Joss: 0.9—Stochastic—Memory depth: 1. [7, 25]
88. Level Punisher—Deterministic—Memory depth: ∞. [50]
89. Limited Retaliate 2: 0.08, 15—Deterministic—Memory depth: ∞. [3]
90. Limited Retaliate 3: 0.05, 20—Deterministic—Memory depth: ∞. [3]
91. Limited Retaliate: 0.1, 20—Deterministic—Memory depth: ∞. [3]
92. MEM2—Deterministic—Memory depth: ∞. [57]
93. Math Constant Hunter—Deterministic—Memory depth: $\infty$. [3]
94. Meta Hunter Aggressive: 7 players—Deterministic—Memory depth: $\infty$. [3]
95. Meta Hunter: 6 players—Deterministic—Memory depth: $\infty$. [3]
96. Meta Mixer: 173 players—Stochastic—Memory depth: $\infty$. [3]
97. Naive Prober: 0.1—Stochastic—Memory depth: 1. [43]
98. Negation—Stochastic—Memory depth: 1. [56]
99. Nice Average Copier—Stochastic—Memory depth: $\infty$. [3]
100. Nydegger—Deterministic—Memory depth: 3. [25]
101. Omega TFT: 3, 8—Deterministic—Memory depth: $\infty$. [12]
102. Once Bitten—Deterministic—Memory depth: 12. [3]
103. Opposite Grudger—Deterministic—Memory depth: $\infty$. [3]
104. PSO Gambler 1_1_1—Stochastic—Memory depth: $\infty$. [3]
105. PSO Gambler 2_2_2—Stochastic—Memory depth: $\infty$. [3]
106. PSO Gambler 2_2_2 Noise 05—Stochastic—Memory depth: $\infty$. [3]
107. PSO Gambler Mem1—Stochastic—Memory depth: 1. [3]
109. Prober—Deterministic—Memory depth: $\infty$. [43]
110. Prober 2—Deterministic—Memory depth: $\infty$. [35]
111. Prober 3—Deterministic—Memory depth: $\infty$. [35]
112. Prober 4—Deterministic—Memory depth: $\infty$. [35]
113. Pun1—Deterministic—Memory depth: 2. [14]
114. Punisher—Deterministic—Memory depth: $\infty$. [3]
115. Raider—Deterministic—Memory depth: 3. [17]
117. Random: 0.1—Stochastic—Memory depth: 0.
118. Random: 0.3—Stochastic—Memory depth: 0.
119. Random: 0.5—Stochastic—Memory depth: 0. [25, 44]
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. [43]
123. Resurrection—Deterministic—Memory depth: 5. [50]
124. Retaliate 2: 0.08—Deterministic—Memory depth: $\infty$. [3]
125. Retaliate 3: 0.05—Deterministic—Memory depth: $\infty$. [3]
126. Retaliate: 0.1 — *Deterministic* — *Memory depth*: ∞. [3]
128. Ripoff — *Deterministic* — *Memory depth*: 2. [58]
129. Risky QLearner — *Stochastic* — *Memory depth*: ∞. [3]
130. SelfSteem — *Stochastic* — *Memory depth*: ∞. [59]
131. ShortMem — *Deterministic* — *Memory depth*: 10. [59]
132. Shubik — *Deterministic* — *Memory depth*: ∞. [25]
133. Slow Tit For Two Tats — *Deterministic* — *Memory depth*: 2. [3]
134. Slow Tit For Two Tats 2 — *Deterministic* — *Memory depth*: 2. [35]
135. Sneaky Tit For Tat — *Deterministic* — *Memory depth*: ∞. [3]
136. Soft Go By Majority — *Deterministic* — *Memory depth*: ∞. [1, 45]
139. Soft Go By Majority: 40 — *Deterministic* — *Memory depth*: 40. [3]
140. Soft Go By Majority: 5 — *Deterministic* — *Memory depth*: 5. [3]
142. Soft Joss: 0.9 — *Stochastic* — *Memory depth*: 1. [35]
144. SolutionB5 — *Deterministic* — *Memory depth*: 5. [15]
145. Spiteful Tit For Tat — *Deterministic* — *Memory depth*: ∞. [35]
146. Stochastic Cooperator — *Stochastic* — *Memory depth*: 1. [60]
147. Stochastic WSLS: 0.05 — *Stochastic* — *Memory depth*: 1. [3]
148. Suspicious Tit For Tat — *Deterministic* — *Memory depth*: 1. [46, 53]
149. TF1 — *Deterministic* — *Memory depth*: ∞. [3]
150. TF2 — *Deterministic* — *Memory depth*: ∞. [3]
151. TF3 — *Deterministic* — *Memory depth*: ∞. [3]
152. Tester — *Deterministic* — *Memory depth*: ∞. [10]
153. ThueMorse — *Deterministic* — *Memory depth*: ∞. [3]
154. ThueMorseInverse — *Deterministic* — *Memory depth*: ∞. [3]
155. Thumper — *Deterministic* — *Memory depth*: 2. [58]
156. Tit For 2 Tats (Tf2T) — *Deterministic* — *Memory depth*: 2. [1]
157. Tit For Tat (TfT) — *Deterministic* — *Memory depth*: 1. [25]
158. Tricky Cooperator — *Deterministic* — *Memory depth*: 10. [3]
159. Tricky Defector—Deterministic—Memory depth: ∞. [3]
161. Two Tīts For Tāt (2TT)—Deterministic—Memory depth: 2. [1]
162. VeryBad—Deterministic—Memory depth: ∞. [59]
163. Willing—Stochastic—Memory depth: 1. [49]
164. Win-Shift Lose-Stay: D (WShLSt)—Deterministic—Memory depth: 1. [43]
165. Win-Stay Lose-Shift: C (WSLS)—Deterministic—Memory depth: 1. [7, 52, 61]
166. Winner12—Deterministic—Memory depth: 2. [26]
167. Winner21—Deterministic—Memory depth: 2. [26]
168. Worse and Worse—Stochastic—Memory depth: ∞. [35]
169. Worse and Worse 2—Stochastic—Memory depth: ∞. [35]
170. Worse and Worse 3—Stochastic—Memory depth: ∞. [35]
171. ZD-Extort-2 v2: 0.125, 0.5, 1—Stochastic—Memory depth: 1. [62]
172. ZD-Extort-2: 0.1111111111111111, 0.5—Stochastic—Memory depth: 1. [7]
173. ZD-Extort-4: 0.23529411764705882, 0.25, 1—Stochastic—Memory depth: 1. [3]
174. ZD-GEN-2: 0.125, 0.5, 3—Stochastic—Memory depth: 1. [62]
175. ZD-GTFT-2: 0.25, 0.5—Stochastic—Memory depth: 1. [7]
176. ZD-SET-2: 0.25, 0.0, 2—Stochastic—Memory depth: 1. [62]

**Acknowledgments**

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

A variety of software libraries have been used in this work:

- The Axelrod library (IPD strategies and Tournaments) [3].
- The matplotlib library (visualisation) [40].
- The pandas and numpy libraries (data manipulation) [41, 42].

**Author Contributions**

**Conceptualization:** Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos.

**Data curation:** Vincent Knight.

**Formal analysis:** Vincent Knight.

**Methodology:** Marc Harper, Vincent Knight.

**Software:** Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Owen Campbell.
Visualization: Vincent Knight, Nikoleta E. Glynatsi.

Writing – original draft: Marc Harper, Vincent Knight, Nikoleta E. Glynatsi.

Writing – review & editing: Marc Harper, Vincent Knight, Owen Campbell.

References


6. Li J. How to design a strategy to win an IPD tournament. The iterated prisoner’s dilemma. 2007; 20:89–104. https://doi.org/10.1142/9789812770684_0004


