Reviving, reproducing and revisiting Axelrod's second tournamentt

Vincent Knight Owen Campbell Marc Harper T. J. Gaffney Nikoleta Glynatsi

May 2, 2025

1 Introduction

The Prisoner's Dilemma has been the centre of the study of cooperative behaviour since Robert Axelrod's seminal work in the 1980s [1, 2, 4]. In this work, Robert Axelrod invited fellow academics and the wider public, through computer hobbyist magazines (one contribution was from an eleven year old), to write computer code to play the Iterated Prisoner's Dilemma.

This initial research has subsequently led to a huge area of research aiming to understand the emergence of cooperative behaviour. A good overview of the variety of research areas is given in [14].

In the Prisoner's Dilemma, each player chooses simultaneously and independently 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).

In Robert Axelrod's tournaments the particular values $R = 3, P = 1, S = 0, T = 5$ were used. Strategies were submitted written in either Fortran [16] or Basic [5] languages (in which case they were translated to Fortran). The very first tournament [1] involved 14 submissions (complemented by a 15th random strategy). Famously, the winner of this tournament was Tit For Tat: a strategy that mimics the opponent's previous action. The only sources for this work are the paper itself, all of the source code is apparently lost. In the second tournament [2] 64 strategies were submitted and again: Tit For Tat was declared the winner. From a computation archaeological point of view this tournament was far superior as the source code for all the strategies was kept and made available for use. It can be found at `http://www-personal.umich.edu/~axe/research/Software/CC/CC2/TourExec1.1.f.html`.

This manuscript describes the process of reviving and using these strategies in a modern software framework ([18]): a Python library with over 200 strategies and a large number of analytical procedures which has been used in ongoing research [11, 12, 9, 6, 13].

Importantly, reviving the strategies is not done through a manual exercise of reverse engineering the Fortran code which would be prone to mistakes. This is done by calling the original functions ensuring the **best possible reproduction and analysis of Robert Axelrod's original work**. This will be described in more detail in Section 2.

Results and further analyses pertaining to reproducing the tournament will be given in Section 3. Finally, Section 4 will extend the analysis to include contemporary research:

- Experimenting with adding one, two or three of over 200 other strategies from [18] to see how the results change.
- Running a tournament with all considered strategies (more than 250).

This work contributes to the game theoretic literature by providing the first exercise in reproducing reported results that have been at the core of the study of cooperation. Furthermore it also provides a contemporary lens which, amongst other things concludes with one of the largest Iterated Prisoner's Dilemma tournaments. Finally, by reviving the original code and making it available to use in [18] it is now possible to use the original strategies of Robert Axelrod's second tournament in a modern framework which for example allows for the study of topological and evolutionary variants.

2 Reviving the tournament

As described in Section 1, the original source code for Axelrod's second tournament was written in Fortan (some contributors submitted code in Basic), this was subsequently published at [3]. This website maintained by the University of Michigan Center for the Study of Complex Systems was last updated (at the time of writing) in 1996. The source code was originally a single file (TourExec1.1.f) and is published on the site in HTML format.

For the purposes of this work, the html formatting was removed to produce the original fortran file which was then minimally modified so that it would compile on a modern compiler.

Furthermore, each strategy was extracted in to a single modular file which follows modern best practice and makes analysis more readable.

Finally, a Makefile was created to control the compilation of the fortran strategy files into a single binary shared object file (named `libstrategies.so`) which is then installed into a standard location on a Posix compliant operating system.

This work can be found at `https://github.com/Axelrod-Python/TourExec` and has been archived at [7]. This archival is itself a major contribution of this paper as it ensures permanency of the original source code.

A Python library has been written that enables an interface to the Axelrod library described in the previous section. This library is referred to as the `Axelrod_fortran` library and is available at `https://github.com/Axelrod-Python/axelrod-fortran` and the specific version used for this work is [8].

This library has the binary file `libstrategies.so` described above as a dependency but otherwise offers a straightforward to install and use option for the study of the strategies of [2] (using standard scientific Python packages).

For this to work there are a variety of minor translations that need to take place. As documented at `http://www-personal.umich.edu/~axe/research/Software/CC/CC2/TourExec1.1.f.html` the Fortran code implies that each strategy is a Fortran function which takes the following inputs.

- `J`: The opponent's previous move: 0 corresponds to a defection and 1 to a cooperation.
- `M`: The current turn number (starting at 1).
- `K`: The player's current cumulative score.
- `L`: The opponent's current cumulative score.
- `R`: A random number between 0 and 1: used by stochastic strategies.
- `JA`: The player's previous move.

For example, Figure 1 shows the source code for `k92r.f` also known as Tit For Tat.

```
      FUNCTION K92R(J,M,K,L,R, JA)
C BY ANATOL RAPOPORT
C TYPED BY AX 3/27/79 (SAME AS ROUND ONE TIT FOR TAT)
c replaced by actual code, Ax 7/27/93
c  T=0
c   K92R=ITFTR(J,M,K,L,T,R)
      k92r=0
      k92r = j
c test 7/30
c   write(6,77) j, k92r
c77   format('␣test␣k92r.␣j,k92r:␣', 2i3)
      RETURN
      END
```

Figure 1: Fortran Source code for original Tit For Tat strategy submitted to Axelrod's second tournament.

The `Axelrod` library takes advantage of the modern Object Oriented framework in Python. Each strategy is a class with agent based behaviour. The input to each player is simply the opponent with both holding their respective history of plays. In the newly written `Axelrod_fortran` library a class inherited from the base `Axelrod` class is written that interfaces with the Fortran strategies and the Python requirements. This includes, for example passing an initial move required by the Fortran player: using the original Fortran code as reference (see Figure 2) it is assumed that the assumed prior move is a cooperation (which is in line with Figure 1).

Figure 3 shows a diagrammatic representation of the interface between the Python strategy and the Fortran function.

The major advantage of this approach is that at no point has any subjectivity been added to the process of replicating Axelrod's second tournament. Indeed for some strategies the only description available is the Fortran code itself, thus they are being run and used as available. To the authors' knowledge this is the best possible way to replicate Axelrod's work which is the subject of the next section.

```
67        Do 20 Game = 1,5
68              RowGameSc = 0
69              ColGameSc = 0
70              JA = 0          ! Row's previous move, reported to column
71              JB = 0          ! Col's previous move, reported to row
```

Figure 2: A portion of the code `https://github.com/Axelrod-Python/TourExec/blob/master/src/tournament/AxTest.f` setting the default previous move to a cooperation and score to 0.

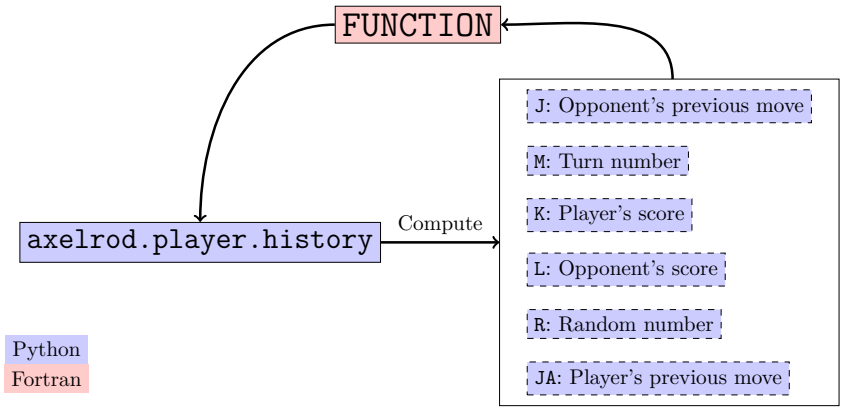


Figure 3: The interface between the Python axelrod library and the Fortran code

3 Reproducing the tournament

From [2], the following characteristics of the original tournament have been identified:

- Matches have length from $\{63, 77, 151, 308, 156\}$;
- Players do not know the number of rounds in a given match;
- A total of 25,000 repetitions across the various match lengths have been carried out.

Note that there is some lack of clarity in [2] as to the length of the matches:

> *"As announced in the rules, the length of the games was determined probabilistically with a .00346 chance of ending with each given move. This parameter was chosen so that the expected median length of a game would be 200 moves. In practice, each pair of players was matched five times, and the lengths of these five games were determined once and for all by drawing a random sample. The resulting random sample from the implied distribution specified that the five games for each pair of players would be of lengths 63, 77, 151, and 308 moves. Thus the average length of a game turned out to be somewhat shorter than expected at 151 moves."*

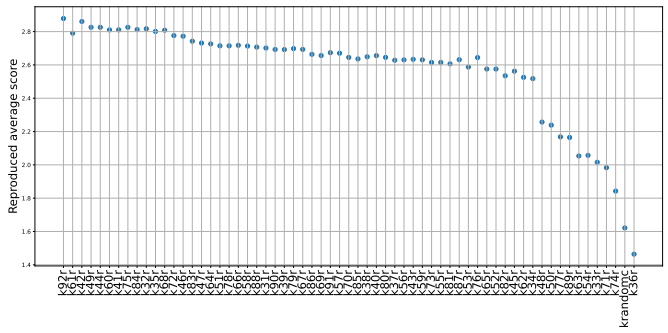
As only four match length samples were specified but the average was given, a fifth match length of 156 is assumed (giving the correct average of 151). It seems that the only stochastic smoothing used was these five repetitions, without a known seed it is not possible to replicate, thus the original tournament is repeated a total of 25,000 for each match length.

The replicated scores and corresponding rankings of each strategy across all repetitions are shown in Figure 4.

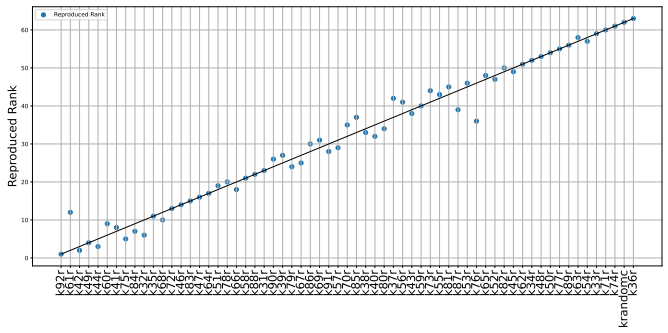
The top 15 strategies in the reproduced tournament are shown in Table 1.

Whilst the results show an overall agreement with the original reported results, a distinct outlier is `k61r`. `k61r`, referred to as Champion: cooperates for the first 10 moves, plays tit for tat for the next fifteen and then will cooperate unless: the other player defected on the previous move, the other player cooperated less than 60% and a random number between 0 and 1 is greater that the other player's cooperation rate. Upon closer investigation a bug was noted in the original code for `k61r`. One initial value was not being initialised, the modified version of this strategy is shown in Figure 5

It is not clear if this bug affected the tournament results reported in 1980. There is no source of the specific code used to run the tournaments and the bug only effects `k61r` on a second use of the function (the very first time: `ICOOP` is assumed to be 0). Thus, it is possible that every single match of the tournament was run in isolation.



(a) Average score per turn of each strategy.



(b) Ranking of each strategy.

Figure 4: Replicated tournament with strategies ordered by original rank

	Author	Scores	Rank	Original Rank	Mean Cooperation Rate
k92r	Anatol Rapoport	2.878	1	1	0.922
k61r	Danny C Champion	2.791	12	2	0.954
k42r	Otto Borufsen	2.861	2	3	0.916
k49r	Rob Cave	2.826	4	4	0.892
k44r	William Adams	2.826	3	5	0.882
k60r	Jim Graaskamp and Ken Katzen	2.810	9	6	0.844
k41r	Herb Weiner	2.811	8	7	0.865
k75r	Paul D Harrington	2.826	5	8	0.802
k84r	T Nicolaus Tideman and Paula Chieruzz	2.812	7	9	0.882
k32r	Charles Kluepfel	2.817	6	10	0.873
k35r	Abraham Getzler	2.801	11	11	0.888
k68r	Fransois Leyvraz	2.808	10	12	0.917
k72r	Edward C White Jr	2.776	13	13	0.924
k46r	Graham J Eatherley	2.772	14	14	0.948
k83r	Paul E Black	2.743	15	15	0.935

Table 1: Top 15 strategies in the reproduced tournament

```
1       FUNCTION K61R(ISPICK,ITURN,K,L,R, JA)
2 C BY DANNY C. CHAMPION
3 C TYPED BY JM 3/27/79
4       k61r=ja     ! Added 7/27/93 to report own old value
5       IF (ITURN .EQ. 1) ICOOP = 0  ! Added 10/8/2017 to fix bug for multiple runs
6       IF (ITURN .EQ. 1) K61R = 0
7       IF (ISPICK .EQ. 0) ICOOP = ICOOP + 1
8       IF (ITURN .LE. 10) RETURN
9       K61R = ISPICK
10      IF (ITURN .LE. 25) RETURN
11      K61R = 0
12      COPRAT = FLOAT(ICOOP) / FLOAT(ITURN)
13      IF (ISPICK .EQ. 1 .AND. COPRAT .LT. .6 .AND. R .GT. COPRAT)
14     +K61R = 1
15      RETURN
16      END
```

Figure 5: Original code for `k61r` with fixed bug on line 5.

Despite fixing this bug and verifying all other strategies for potentially similar bugs there is still a discrepancy in the results. Figure 6 shows the results of the tournament computed using a number of different approaches.

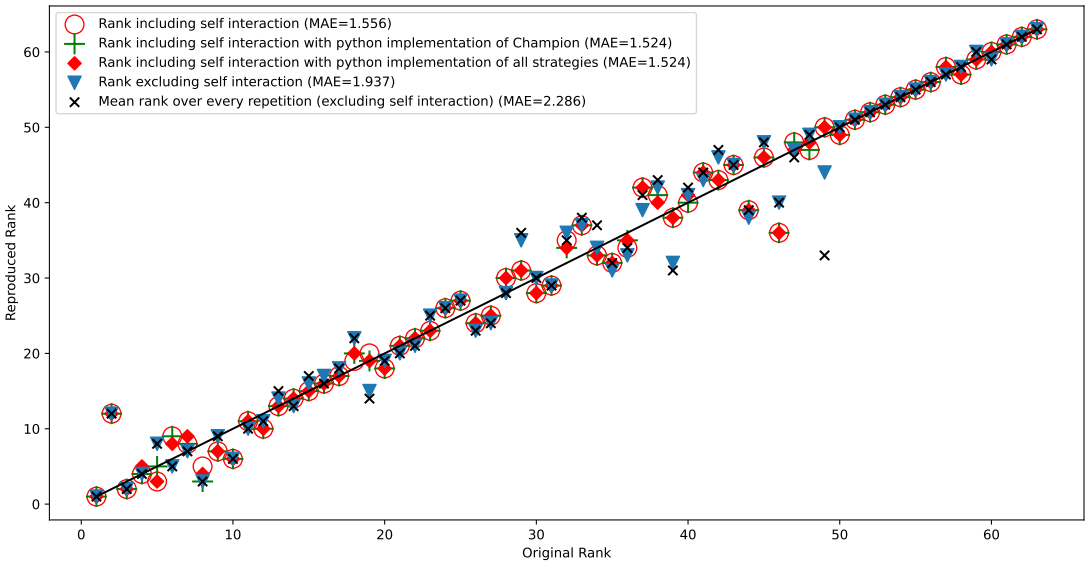


Figure 6: Using a variety of approaches to reproduce the original tournament. This includes using a Python implementation of Champion's strategy as well as using as many Python implementation of the Fortran strategies as possible.

There is no immediate explanation for the remaining discrepancy with the results reported in [2]. Potential explanations include:

- Stochastic variation not being sufficiently taken in to account in [2].
- A difference with how an older Fortran compiler would interpret the commands: this is not obvious though, the implemented version seems to interact as expected.
- An error in the reporting of [2] which could include a modification of the source code.

Apart from the strategy by Champion, the agreement between the original and the reproduced tournament is strong. The main conclusions included for example that Tit For Tat (`k92r`) once again wins the tournament. Furthermore, the fact that high performing strategies are "nice" is also evident although perhaps interestingly `k42r` which takes the second rank of `61r` is a strategy that cooperates with most strategies apart from itself. The overall cooperation rate of the tournament is 0.750 . Figure 7 shows the cooperation rates of the tournament. It is clear that the high performing strategies cooperate overall more often. Looking at the pairwise cooperation rates in Figure 7b shows that the high performing strategies generally seems to cooperate with high performing strategies. This underpins one of the main conclusions of [2] explaining the emergence of cooperation in competitive environments.

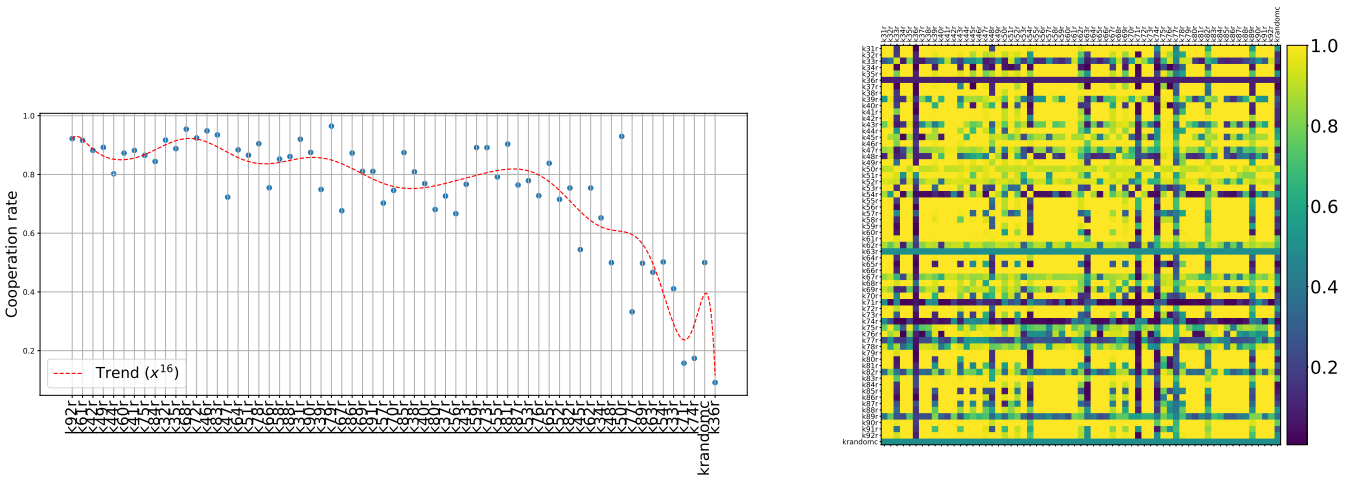
In [2], a linear regression model is used to identify 5 strategies, the scores against which are good predictors of the overall performance. The reported R^2 value is 0.979 (indicating 97% of variance accounted for by the model). For completeness, the coefficients of this model (reported in [2]) are shown in Table 2.

Given the discrepancy in results shown in Figure 4 and Table 1 it is not surprising to see that this model no longer performs as well with $R^2 = 0.7589$.

Fitting a new model to the same 5 strategies gives the coefficients shown in Table 3 with $R^2 = 0.924$

Using recursive feature elimination [10] it is possible to select the features (strategies) that give the best prediction for a given number of features. This *best* R^2 versus the number of features is shown in Figure 8.

Tables 4, 5 and 6 show the coefficients for linear models fitted to 5, 8 and 25 strategies with $R^2 = 0.924$, $R^2 = 0.967$ and $R^2 = 0.991$ respectively (8 strategies is the smallest number of strategies for which $R^2 > 96$ and 25 strategies is the smallest number of strategies for which $R^2 > 99$).



(a) Mean cooperation rates

(b) Cooperation rates between each pair of players

Figure 7: Replicated tournament cooperation rates with strategies ordered by original rank

Strategies	Coefficients
k69r	0.202000
k91r	0.198000
k40r	0.110000
k76r	0.072000
k67r	0.086000
Intercept	0.795000

Table 2: Linear model described in [2] with $R^2 = 0.7589$

Strategies	Coefficients	p-value	F-value
k69r	0.098640	0.000000	43.976036
k91r	0.207380	0.000000	56.667196
k40r	0.206440	0.000000	78.191831
k76r	0.059930	0.025992	5.207451
k67r	0.123740	0.000002	27.893148
Intercept	0.768410	NA	NA

Table 3: Linear model fitted to the same 5 strategies described in [2] with $R^2 = 0.885$

Strategies	Coefficients	p-value	F-value
k51r	0.458150	0.002438	10.001032
k56r	-24.586770	0.000000	48.729856
k59r	24.806190	0.000000	48.744926
k70r	0.261350	0.000000	79.384616
k89r	0.148550	0.015430	6.210811
Intercept	-0.399530	NA	NA

Table 4: Linear model best fitted to 5 strategies in the reproduced tournament with $R^2 = 0.924$

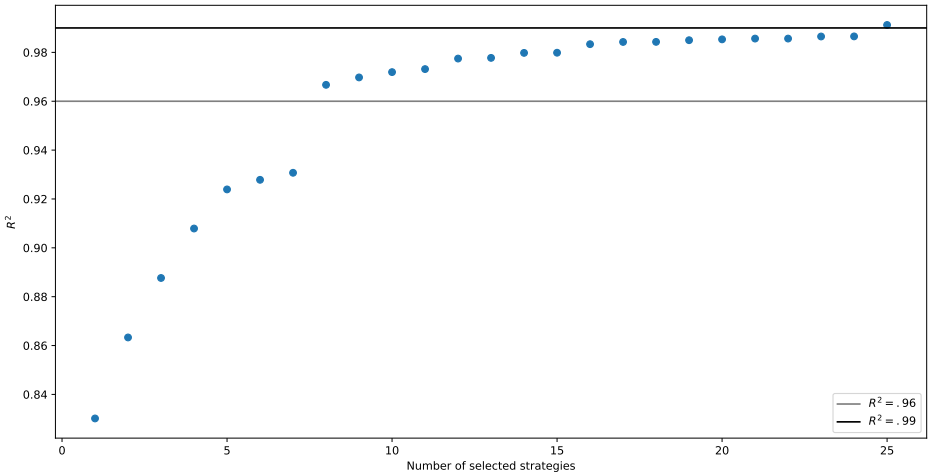


Figure 8: R^2 for models obtained using recursive feature elimination.

Strategies	Coefficients	p-value	F-value
k36r	0.133320	0.189994	1.756582
k51r	0.187270	0.002438	10.001032
k56r	-11.100380	0.000000	48.729856
k59r	11.231820	0.000000	48.744926
k62r	0.125620	0.703116	0.146621
k66r	0.151600	0.000000	135.614172
k70r	0.221780	0.000000	79.384616
k89r	0.172540	0.015430	6.210811
Intercept	-0.208980	NA	NA

Table 5: Linear model best fitted to 8 strategies in the reproduced tournament with $R^2 = 0.967$

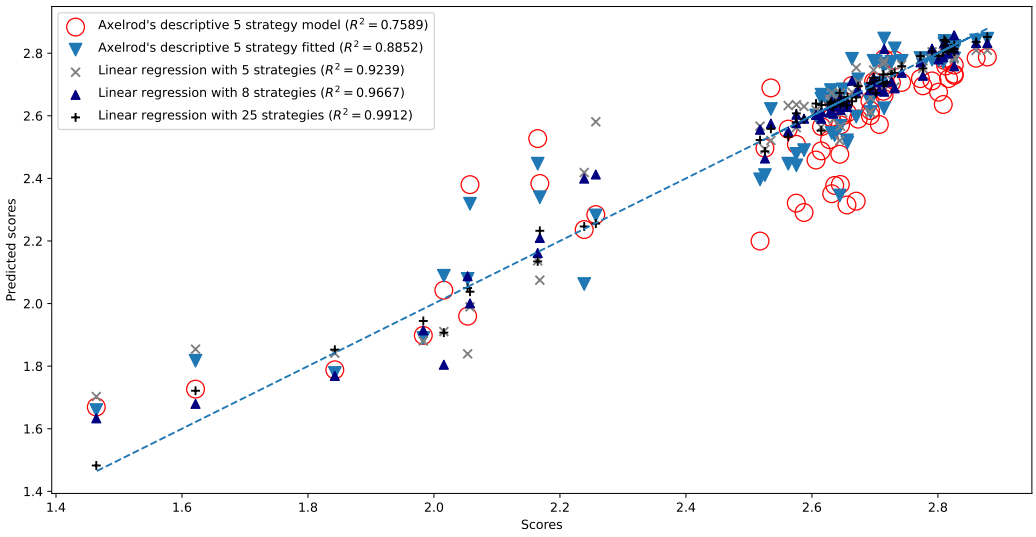


Figure 9: Predicting the performance of strategies using the 4 models discussed

Strategies	Coefficients	p-value	F-value
k32r	-0.060330	0.000000	186.247539
k34r	0.036030	0.000000	42.866846
k36r	0.044880	0.189994	1.756582
k41r	0.073800	0.000000	177.937911
k42r	0.044080	0.000000	79.351204
k43r	0.014170	0.000000	97.073236
k46r	0.044870	0.004805	8.566798
k47r	0.028380	0.000000	43.355694
k48r	0.029490	0.007426	7.670719
k51r	0.074000	0.002438	10.001032
k56r	-3.499880	0.000000	48.729856
k59r	3.551580	0.000000	48.744926
k60r	0.042590	0.000000	298.163936
k62r	0.042710	0.703116	0.146621
k66r	0.086580	0.000000	135.614172
k70r	0.067690	0.000000	79.384616
k73r	0.075500	0.000000	67.006588
k74r	0.023020	0.244116	1.383261
k75r	0.024290	0.169382	1.933926
k76r	0.031590	0.025992	5.207451
k81r	0.000130	0.000000	84.798727
k82r	0.032720	0.000001	29.385307
k89r	0.022020	0.015430	6.210811
k90r	0.041170	0.960298	0.002498
k91r	0.071910	0.000000	56.667196
Intercept	0.157940	NA	NA

Table 6: Linear model best fitted to 25 strategies in the reproduced tournament with $R^2 = 0.991$

The predictions of these models are shown in Figure 9.

It is clear that the effectiveness of the predictive models with 5 strategies is low for the cluster of highly performing strategies (with a score greater than 2.5). To be able to obtain a good model even for high performing strategies 8 seem to provide a good predictive model.

This is the first known re run of the work of [2] (which lead to [4] which has over 33000 citations). Despite some misalignment with the results the overall principles are well aligned: Tit For Tat wins, offering scope for the effectiveness of cooperation. [2] used these results to give guidelines for effective behaviour:

1. Do not be envious by striving for a payoff larger than the opponent's payoff.
2. Be "nice" by not being the first to defect.
3. Reciprocate both cooperation and defection; Be provocable to retaliation and forgiveness.
4. Do not be too clever by scheming to exploit the opponent.

Whilst these certainly apply to Axelrod's particular tournament, in the next section we revisit the tournament and aim to see how correct the Game Theoretic community was to take this guidelines as a generalization.

4 Revisiting the tournament

In this section, the tournament of [2] will be expanded. Indeed, a large amount of research has gone on since Axelrod's original work which include for example training of strategies using reinforcement learning [11] but also the discovery of Zero Determinant strategies [15]. This section aims to measure how well these strategies would have faired and **if** any of the original insights and conclusions would differ.

4.1 Running a large tournament

The Axelrod Python library has access to more than 200 strategies and regularly runs the largest iterated prisoners Dilemma tournament ever run (everytime a new strategy is contributed to the project the results are updated). We here expand this tournament further by including all of the original fortran strategies (omitting duplication). This gives a tournament with 272 strategies. As for [7] the archival of all of the interactions and results from this large tournament are another contribution of this paper.

Table 7 shows the rankings of the top 20 strategies when including all strategies over 25,000 repetitions. The cooperation rate for the tournament that pits all the library strategies against each other (without the Fortran ones) is 0.619.

	Mean Score	Rank	Coop. Rate	Original Rank	Original Coop. Rate	Library Rank	Library Coop. Rate
EvolvedLookerUp2_2_2	2.802	1	0.756	NA	NA	1	0.736
Evolved HMM 5	2.788	2	0.751	NA	NA	2	0.742
Omega TFT: 3, 8	2.786	3	0.750	NA	NA	12	0.725
Evolved ANN 5	2.784	4	0.720	NA	NA	3	0.713
Evolved ANN	2.782	5	0.739	NA	NA	5	0.727
Evolved FSM 16	2.780	6	0.731	NA	NA	4	0.713
Evolved FSM 16 Noise 05	2.776	7	0.726	NA	NA	6	0.717
PSO Gambler 2_2_2	2.760	8	0.702	NA	NA	7	0.692
Original Gradual	2.757	9	0.808	NA	NA	NA	NA
PSO Gambler 2_2_2 Noise 05	2.756	10	0.740	NA	NA	14	0.723
PSO Gambler Mem1	2.756	11	0.734	NA	NA	9	0.728
Evolved FSM 4	2.752	12	0.793	NA	NA	10	0.777
PSO Gambler 1_1_1	2.752	13	0.704	NA	NA	8	0.702
Gradual	2.746	14	0.763	NA	NA	15	0.793
DBS: 0.75, 3, 4, 3, 5	2.739	15	0.750	NA	NA	11	0.739
k42r	2.735	16	0.820	3	0.916	NA	NA
Winner12	2.733	17	0.688	NA	NA	16	0.682
Spiteful Tit For Tat	2.723	18	0.691	NA	NA	19	0.664
k60r	2.723	19	0.691	6	0.844	NA	NA
k85r	2.716	20	0.627	33	0.726	NA	NA
EugineNier: (D,)	2.713	21	0.673	NA	NA	22	0.645
k80r	2.706	22	0.723	36	0.809	NA	NA
k32r	2.704	23	0.758	10	0.873	NA	NA
DoubleCrosser: (D, D)	2.703	24	0.693	NA	NA	13	0.677
k58r	2.703	25	0.746	21	0.852	NA	NA

Table 7: Top 25 strategies in the tournament when using all available strategies

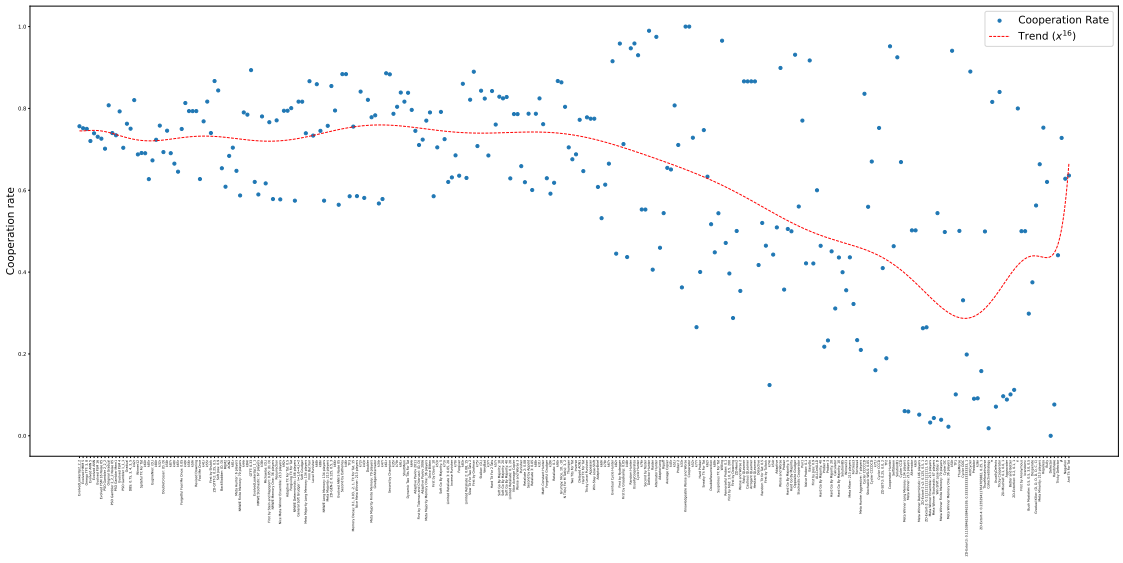


Figure 10: Cooperation rate versus rank for tournament with all available strategies

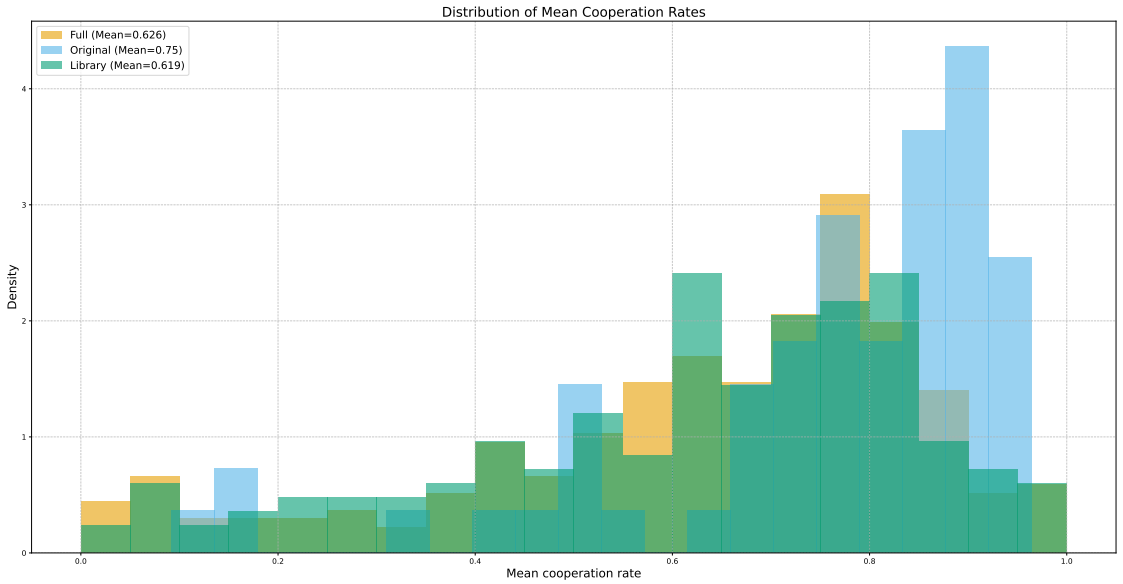


Figure 11: Distribution of cooperation rates for the full tournament.

The overall cooperation rate of this tournament is 0.626 and the various cooperation rates are shown in Figure 10 shows the cooperation rates of each strategy (ordered by rank).

Figure 12 shows the change of behaviour between tournaments. It is clear that the original fortran strategies cooperate less than they did in the original tournament. Furthermore, looking at Table 7 the strategies that rank highly are sophisticated strategies that have been trained using reinforcement learning. The "Evolved" strategies have been trained using evolutionary algorithms and the "PSO" strategies have been trained using Particle Swarm Algorithms, these are described in [11]. Notably these high performing strategies do not necessarily follow the Axelrod's guidelines. In fact they are more in line with the guidelines identified in [9] in which not just one tournament was used but thousands to identify the following properties:

1. Be a little bit envious
2. Be "nice" in non-noisy environments or when game lengths are longer
3. Reciprocate both cooperation and defection appropriately; Be provocable in tournaments with short matches, and generous in tournaments with noise
4. It is ok to be clever
5. Adapt to the environment; Adjust to the mean population cooperation

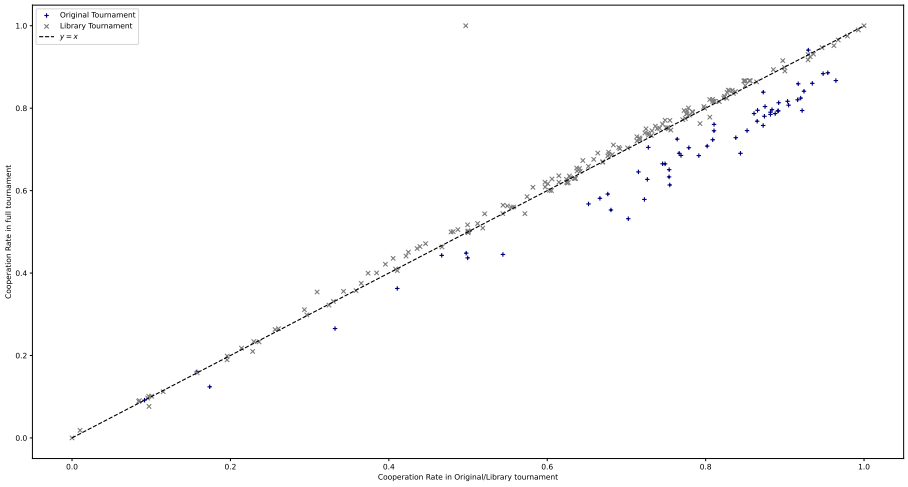


Figure 12: The relationship between cooperation rates in each tournament

In the next section we investigate if a small modification to Axelrod's original tournament would have changed the conclusions. Specifically: would a small set of extra submissions changed the outcome?

4.2 Running with extra invitations: Adding 1, 2 and 3 strategies

Using the full tournament results it is possible to extract results for adding new strategies.

Table 8 shows the top ranking strategies when adding a single strategy to the original tournament.

Adding a single strategy from Table 8 would in fact not modify the outcome: Tit For Tat wins. In fact adding a strategy either gives the same winner or gives either k42r or k60r as the winner however, adding 2 or 3 strategies does change the outcome. Table 9 shows the proportion of times this happens.

Clearly the more strategies added the less likely Tit For Tat is to win. Although, interestingly the environment does not let the high performing strategies of Table 7 to thrive. In the next section we will investigate adding a particularly aggressive subset of strategies.

Name	Mean Score	Rank	Winner
Adaptive Tit For Tat: 0.5	2.876	2	k92r
GTFT: 0.33	2.846	3	k92r
Omega TFT: 3, 8	2.859	3	k92r
Firm But Fair	2.861	3	k92r
ZD-GTFT-2: 0.25, 0.5	2.868	3	k92r
Resurrection	2.867	3	k92r
Meta Majority: 213 players	2.860	3	k92r
Meta Majority Finite Memory: 79 players	2.856	3	k92r
Meta Majority Long Memory: 134 players	2.856	3	k92r
Soft Joss: 0.9	2.871	3	k92r
Forgiving Tit For Tat	2.854	3	k92r
Original Gradual	2.857	3	k92r
Meta Majority Memory One: 36 players	2.838	4	k92r
PSO Gambler 2_2_2 Noise 05	2.840	4	k92r
Gradual	2.831	7	k92r
First by Stein and Rapoport: 0.05: (D, D)	2.827	8	k92r
Spiteful Tit For Tat	2.826	8	k92r
GrudgerAlternator	2.825	8	k92r
EugineNier: (D,)	2.815	10	k92r
First by Shubik	2.791	13	k92r

Table 8: Performance of extra strategy in Axelrod's original tournament

Winner	1 New (N = 209)	2 New (N = 21,736)	3 New (N = 1,499,784)
Adaptive Tit For Tat: 0.5	NaN	0.000092	0.000232
EugineNier: (D,)	NaN	NaN	0.000001
Firm But Fair	NaN	0.000046	0.000123
First by Stein and Rapoport: 0.05: (D, D)	NaN	NaN	0.000001
GTFT: 0.33	NaN	NaN	0.000020
Gradual	NaN	NaN	0.000001
Omega TFT: 3, 8	NaN	0.000506	0.001457
Original Gradual	NaN	0.000046	0.000218
PSO Gambler 2_2_2 Noise 05	NaN	0.000046	0.000227
Resurrection	NaN	0.000092	0.000256
Soft Joss: 0.9	NaN	0.000092	0.000285
Spiteful Tit For Tat	NaN	NaN	0.000049
ZD-GTFT-2: 0.25, 0.5	NaN	0.000138	0.000387
k32r	NaN	NaN	0.000007
k41r	NaN	NaN	0.000013
k42r	0.148325	0.269415	0.366403
k44r	NaN	0.000230	0.000568
k49r	NaN	0.000138	0.000347
k60r	0.004785	0.011180	0.018820
k75r	NaN	0.000506	0.002448
k92r	0.846890	0.717473	0.608137

Table 9: Proportion of winners of the original tournament with 1, 2 or 3 new strategies

4.3 Running with extortion

Since the work of [15] a lot of interest has been shown to Zero Determinant strategies. In [17] a small tournament is presented pitting these against each other. Table 10 shows the rankings of the top 15 strategies when including all the Zero Determinant strategies from [17].

Name	Mean Score	Rank	Mean Cooperation Rate	Original Mean Cooperation Rate	S. and P. Mean Cooperation Rate
ZD-GTFT-2: 0.25, 0.5	2.834	1	0.926	NaN	0.855
k42r	2.825	2	0.903	0.916	NaN
GTFT: 0.33	2.819	3	0.941	NaN	0.893
k92r	2.812	4	0.888	0.922	0.731
k49r	2.787	5	0.879	0.892	NaN
k75r	2.786	6	0.782	0.802	NaN
k44r	2.781	7	0.866	0.882	NaN
k61r	2.773	8	0.945	0.954	NaN
k68r	2.768	9	0.904	0.917	NaN
k41r	2.761	10	0.847	0.865	NaN
k46r	2.758	11	0.939	0.948	NaN
k32r	2.758	12	0.849	0.873	NaN
k72r	2.756	13	0.915	0.924	NaN
k35r	2.747	14	0.863	0.888	NaN
k84r	2.745	15	0.861	0.882	NaN
k60r	2.735	16	0.814	0.844	NaN
k83r	2.725	17	0.922	0.935	NaN
k64r	2.710	18	0.876	0.884	NaN
k66r	2.702	19	0.861	0.866	NaN
Hard Tit For 2 Tats	2.701	20	0.926	NaN	0.807
k39r	2.700	21	0.671	0.676	NaN
k47r	2.696	22	0.709	0.723	NaN
k58r	2.694	23	0.846	0.852	NaN
k31r	2.690	24	0.908	0.920	NaN
k88r	2.684	25	0.851	0.861	NaN

Table 10: Top 15 strategies in the tournament composed of the original strategies and the Zero Determinant strategies from [17]

The overall cooperation rate of this tournament is 0.732 and the various cooperation rates are shown in Figure 13 (ordered by rank). The distribution is shown in Figure 14.

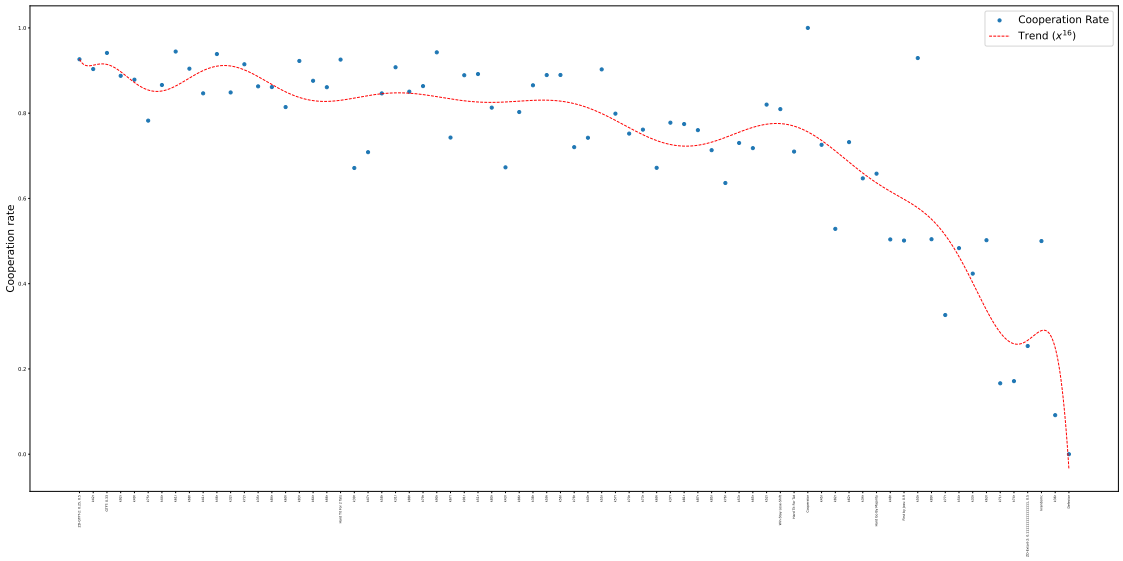


Figure 13: Cooperation rate versus rank for the Stewart and Poltkin tournament

Figure 15 shows the relative cooperation rates and here it identifies that the Zero Determinant strategies in fact adapt their cooperation rate to the original tournament. They cooperate more than.

These results seem to highlight that adaptability is an important guideline for cooperation...

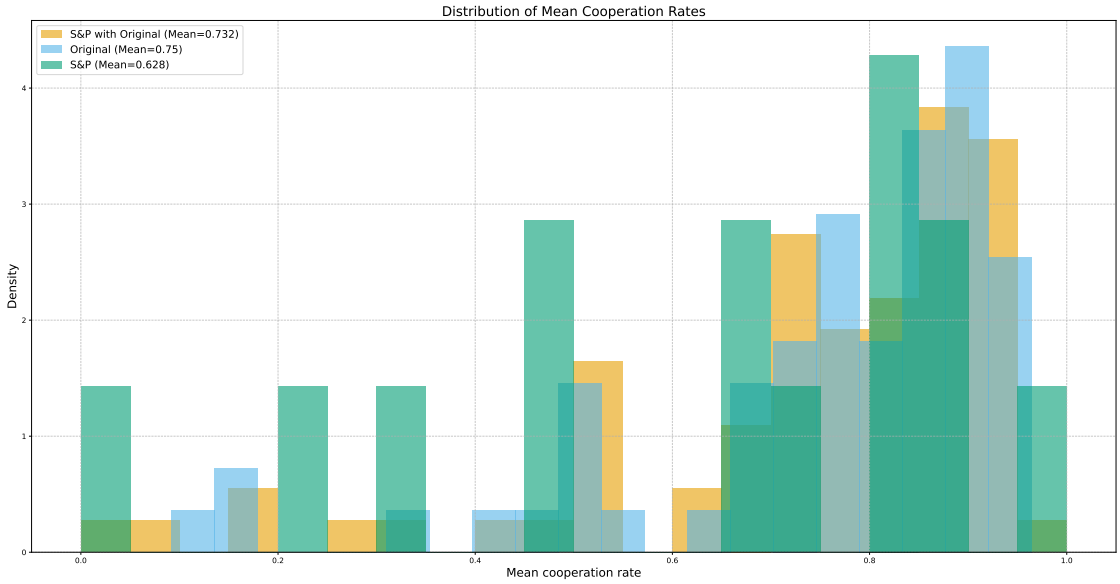


Figure 14: Distribution of cooperation rates for the Stewart and Plotkin tournament

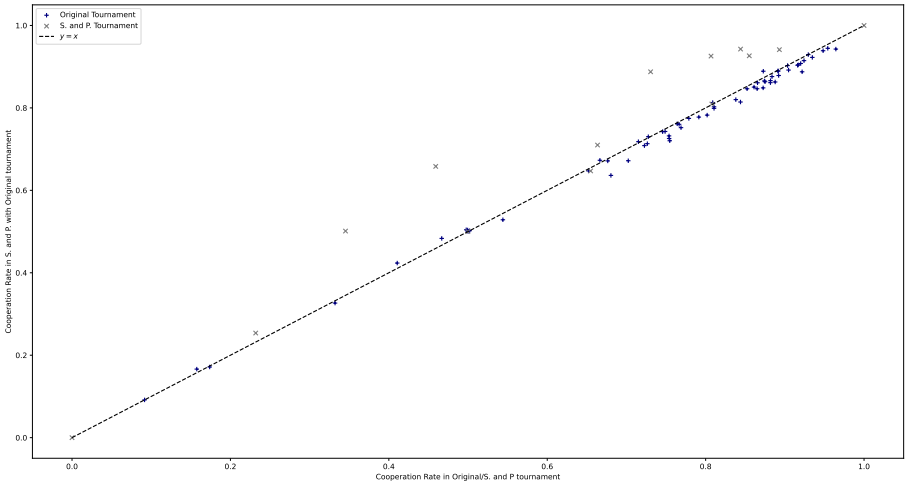


Figure 15: The relationship between cooperation rates in each tournament

5 Conclusion

Acknowledgements

References

[1] R. Axelrod. Effective Choice in the Prisoner's Dilemma. *Journal of Conflict Resolution*, 24(1):3–25, 1980.

[2] R. Axelrod. More Effective Choice in the Prisoner's Dilemma. *Journal of Conflict Resolution*, 24(3):379–403, 1980.

[3] R. Axelrod. Complexity of cooperation web site. http://www-personal.umich.edu/ axe/research/Software/C-C/CC2.html, 1996.

[4] Robert M Axelrod. The evolution of cooperation: revised edition. 2006.

[5] Peter Bishop. *Computer Programming in Basic Students' Book*. Thomas Nelson Publishers, 2004.

[6] Edgardo Bucciarelli, Shu-Heng Chen, Aurora Ascatigno, and Alfredo Colantonio. Studying the distribution of strategies in the two-scenario snowdrift game. In *Digital (Eco) Systems and Societal Challenges*, pages 407–428. Springer, 2024.

[7] Owen Campbell and Vince Knight. Axelrod-python/tourexec: v0.3.1 (2017-09-19). https://doi.org/10.5281/zenodo.896461, September 2017.

[8] Owen Campbell, Vince Knight, and Marc Harper. Axelrod-Python/axelrod-fortran: v0.3.1 (2017-08-04). https://doi.org/10.5281/zenodo.838980, August 2017.

[9] Nikoleta E Glynatsi, Vincent Knight, and Marc Harper. Properties of winning iterated prisoners dilemma strategies. *PLOS Computational Biology*, 20(12):e1012644, 2024.

[10] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[11] Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E. Glynatsi, and Owen Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *CoRR*, abs/1707.06307, 2017.

[12] Vincent Knight, Marc Harper, Nikoleta E. Glynatsi, and Owen Campbell. Evolution reinforces cooperation with the emergence of self-recognition mechanisms: an empirical study of the moran process for the iterated prisoner's dilemma. *CoRR*, abs/1707.06920, 2017.

[13] Olivia Macmillan-Scott, Akin Ünver, and Mirco Musolesi. Game-theoretic agent-based modelling of micro-level conflict: Evidence from the isis-kurdish war. *Plos one*, 19(6):e0297483, 2024.

[14] Martin A Nowak. *Evolutionary dynamics*. Harvard University Press, 2006.

[15] William H Press and Freeman J Dyson. Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences of the United States of America*, 109(26):10409–13, 2012.

[16] Ian Moffat Smith. *Programming in Fortran 90: a first course for engineers and scientists*. John Wiley & Sons, Inc., 1994.

[17] a. J. Stewart and J. B. Plotkin. Extortion and cooperation in the Prisoner's Dilemma. *Proceedings of the National Academy of Sciences*, 109(26):10134–10135, 2012.

[18] The Axelrod project developers. Axelrod-python/axelrod: v3.3.0. https://doi.org/10.5281/zenodo.836439, July 2017.

A List of original players

1. k31r - Original rank: 23. Authored by Gail Grisell
2. k32r - Original rank: 10. Authored by Charles Kluepfel
3. k33r - Original rank: 59. Authored by Harold Rabbie
4. k34r - Original rank: 52. Authored by James W Friedman
5. k35r - Original rank: 11. Authored by Abraham Getzler
6. k36r - Original rank: 63. Authored by Roger Hotz
7. k37r - Original rank: 37. Authored by George Lefevre
8. k38r - Original rank: 34. Authored by Nelson Weiderman
9. k39r - Original rank: 25. Authored by Tom Almy
10. k40r - Original rank: 35. Authored by Robert Adams
11. k41r - Original rank: 7. Authored by Herb Weiner
12. k42r - Original rank: 3. Authored by Otto Borufsen
13. k43r - Original rank: 39. Authored by R D Anderson
14. k44r - Original rank: 5. Authored by William Adams
15. k45r - Original rank: 50. Authored by Michael F McGurrin
16. k46r - Original rank: 14. Authored by Graham J Eatherley
17. k47r - Original rank: 16. Authored by Richard Hufford
18. k48r - Original rank: 53. Authored by George Hufford
19. k49r - Original rank: 4. Authored by Rob Cave
20. k50r - Original rank: 54. Authored by Rik Smoody
21. k51r - Original rank: 18. Authored by John William Colbert
22. k52r - Original rank: 48. Authored by David A Smith
23. k53r - Original rank: 45. Authored by Henry Nussbacher
24. k54r - Original rank: 58. Authored by William H Robertson
25. k55r - Original rank: 42. Authored by Steve Newman
26. k56r - Original rank: 38. Authored by Stanley F Quayle
27. k57r - Original rank: 31. Authored by Rudy Nydegger
28. k58r - Original rank: 21. Authored by Glen Rowsam
29. k59r - Original rank: 40. Authored by Leslie Downing
30. k60r - Original rank: 6. Authored by Jim Graaskamp and Ken Katzen
31. k61r - Original rank: 2. Authored by Danny C Champion
32. k62r - Original rank: 51. Authored by Howard R Hollander
33. k63r - Original rank: 57. Authored by George Duisman
34. k64r - Original rank: 17. Authored by Brian Yamachi
35. k65r - Original rank: 47. Authored by Mark F Batell
36. k66r - Original rank: 20. Authored by Ray Mikkelson
37. k67r - Original rank: 27. Authored by Craig Feathers
38. k68r - Original rank: 12. Authored by Fransois Leyvraz
39. k69r - Original rank: 29. Authored by Johann Joss
40. k70r - Original rank: 32. Authored by Robert Pebly
41. k71r - Original rank: 60. Authored by James E Hall
42. k72r - Original rank: 13. Authored by Edward C White Jr
43. k73r - Original rank: 41. Authored by George Zimmerman
44. k74r - Original rank: 61. Authored by Edward Friedland
45. k75r - Original rank: 8. Authored by Paul D Harrington
46. k76r - Original rank: 46. Authored by David Gladstein
47. k77r - Original rank: 55. Authored by Scott Feld
48. k78r - Original rank: 19. Authored by Fred Mauk
49. k79r - Original rank: 26. Authored by Dennis Ambuehl and Kevin Hickey
50. k80r - Original rank: 36. Authored by Robyn M Dawes and Mark Batell
51. k81r - Original rank: 43. Authored by Martyn Jones
52. k82r - Original rank: 49. Authored by Robert A Leyland
53. k83r - Original rank: 15. Authored by Paul E Black
54. k84r - Original rank: 9. Authored by T Nicolaus Tideman and Paula Chieruzz
55. k85r - Original rank: 33. Authored by Robert B Falk and James M Langsted

56. k86r - Original rank: 28. Authored by Bernard Grofman
57. k87r - Original rank: 44. Authored by E E H Schurmann
58. k88r - Original rank: 22. Authored by Scott Appold
59. k89r - Original rank: 56. Authored by Gene Snodgrass
60. k90r - Original rank: 24. Authored by John Maynard Smith
61. k91r - Original rank: 30. Authored by Jonathan Pinkley
62. k92r - Original rank: 1. Authored by Anatol Rapoport
63. krandomc - Original rank: 62. Authored by None