Reviving, reproducing and revisiting Axelrod's second tournamentt

October 23, 2018

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 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 [9].

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 of $R = 3, P = 1, S = 0, T = 5$ were used. Strategies were submitted written in either Fortran or Basic (in which case they were translated to Fortran). The very first tournament [1] involved 14 submissions (complemented by a fifteenth 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 work describes the process of reviving and using these strategies in a modern software framework ([12]): a Python library with over 200 strategies and a large number of analytical procedures which has been used in ongoing research [7, 8].

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 topics:

- Experimenting with adding one of over 196 other strategies from [12] to see how the results change.
- Investigating the overall performance of Zero determinant strategies [10].
- 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 [12] 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 [5].

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 [6].

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

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.

```fortran
      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.

```
      Do 20 Game = 1,5
            RowGameSc = 0
            ColGameSc = 0
            JA = 0          ! Row's previous move, reported to column
            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.

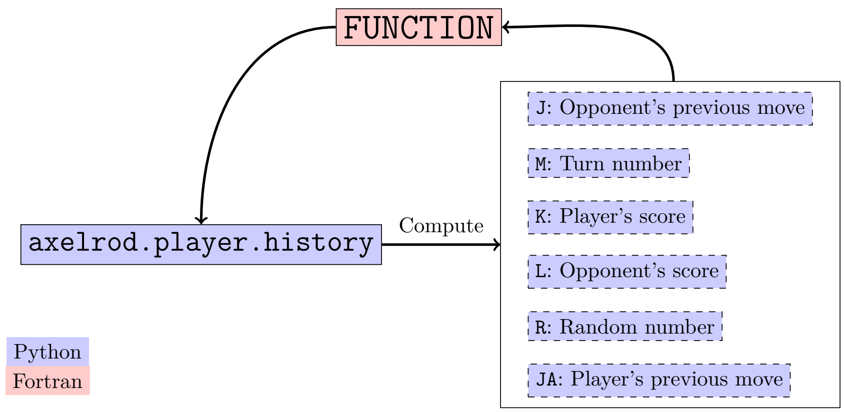


Figure 3: The interface between the Python axelrod library and the Fortran code

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 25000 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."*

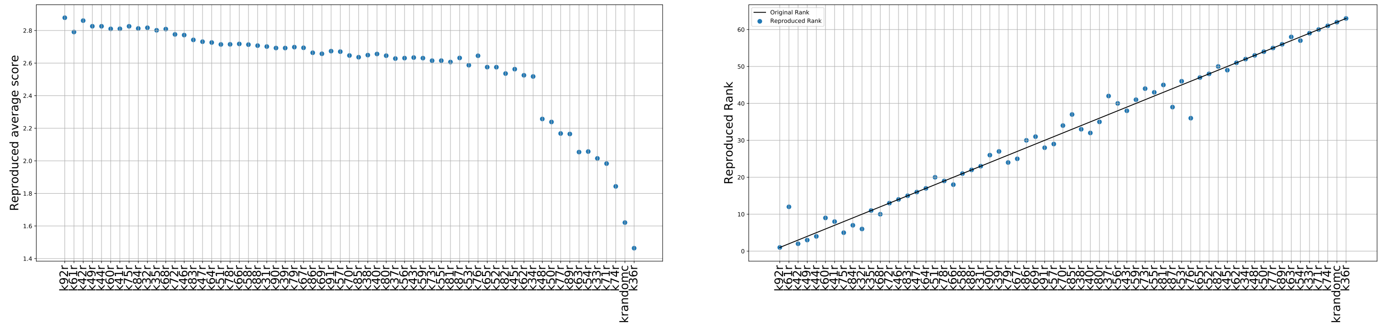
As only four match length samples were specified but the average was given, a fifth match length of 156 is assumed (giving the correct average of 151). It seems that the only stochastic smoothing used was these five repetitions, without a known seed it is not possible to replicate, thus the original tournament is repeated a total of 25000 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`: Is 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
k92r	Anatol Rapoport	2.8785	1	1
k61r	Danny C Champion	2.7909	12	2
k42r	Otto Borufsen	2.8607	2	3
k49r	Rob Cave	2.8262	3	4
k44r	William Adams	2.8261	4	5
k60r	Jim Graaskamp and Ken Katzen	2.8101	9	6
k41r	Herb Weiner	2.8105	8	7
k75r	Paul D Harrington	2.8260	5	8
k84r	T Nicolaus Tideman and Paula Chieruzz	2.8126	7	9
k32r	Charles Kluepfel	2.8170	6	10
k35r	Abraham Getzler	2.8015	11	11
k68r	Fransois Leyvraz	2.8088	10	12
k72r	Edward C White Jr	2.7761	13	13
k46r	Graham J Eatherley	2.7721	14	14
k83r	Paul E Black	2.7425	15	15

Table 1: Top 15 strategies in the reproduced tournament

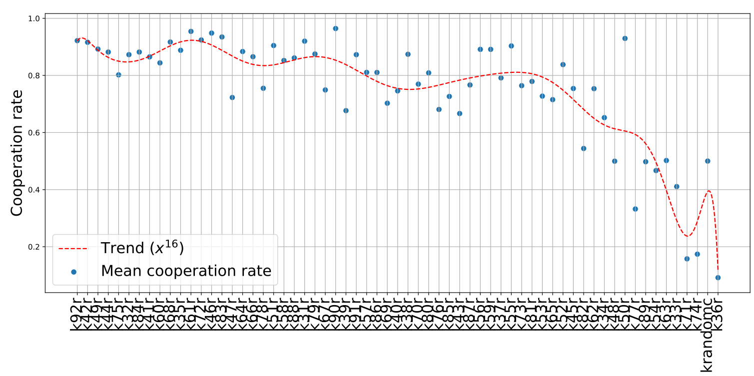
```
1        FUNCTION K61R(ISPICK,ITURN,K,L,R, JA)
2  C BY DANNY C. CHAMPION
3  C TYPED BY JM 3/27/79
4        k61r=ja     ! Added 7/27/93 to report own old value
5        IF (ITURN .EQ. 1) ICOOP = 0   ! Added 10/8/2017 to fix bug for multiple runs
6        IF (ITURN .EQ. 1) K61R = 0
7        IF (ISPICK .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.

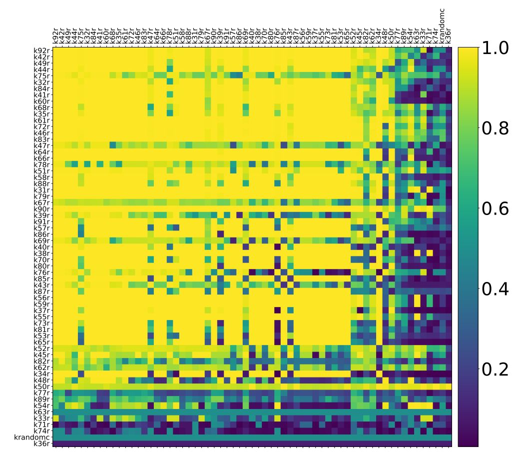
Despite fixing this bug and verifying all other strategies for potentially similar bugs there is still a discrepancy in the results. There is no immediate explanation for this in [2]. Potential explanations include:

- 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 this one outlier, the agreement between the original and the reproduced tournament is very 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 6 shows the cooperation rates of the tournament. It is clear that the high performing strategies cooperate overall more often (Figure **??**). Also, looking at the pairwise cooperation rates in Figure 6b 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.



(a) Mean cooperation rates



(b) Cooperation rates between each pair of players

Figure 6: Replicated tournament cooperation rates with strategies ordered by original rank

For completeness 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). The coefficients of this model are shown in Table 2.

Strategies	Coefficients
k69r	0.202
k91r	0.198
k40r	0.110
k76r	0.072
k67r	0.086
Intercept	0.795

Table 2: Linear model described in [2] with $R^2 = 0.7594$

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 $R^2 = 0.7594$.

Fitting a new model to the same 5 strategies gives the coefficients shown in Table 3 with $R^2 = 0.9440$

Strategies	Coefficients	p-value	F-value
k69r	0.099	9.5178e-09	44.1166
k91r	0.206	3.07113e-10	56.3992
k40r	0.206	1.55822e-12	78.2667
k76r	0.060	0.0257417	5.22576
k67r	0.124	1.7738e-06	27.9476
Intercept	0.770	NA	NA

Table 3: Linear model fitted to the same 5 strategies described in [2] with $R^2 = 0.9440$

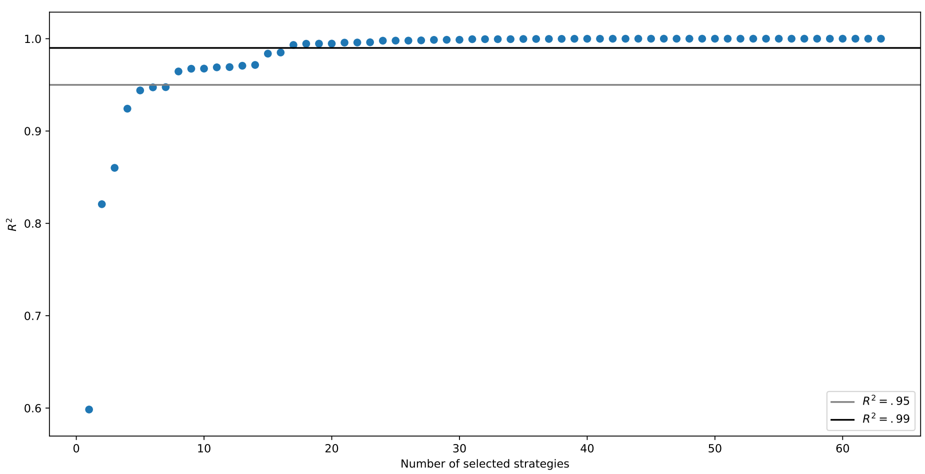


Figure 7: R^2 for models obtained using recursive feature elimination.

Using recursive feature elimination It is possible to select the features (strategies) that give the best prediction for a given number of features. The R^2 versus the number of features is shown in Figure 7.

Tables 4 and **??** show the coefficients for linear models fitted to 5 and 12 strategies with $R^2 = 0.9440$ and $R^2 = 0.9932$ respectively (17 strategies is the smallest number of strategies for which $R^2 > 99$).

Strategies	Coefficients	p-value	F-value
k64r	0.195	1.68964e-14	100.282
k70r	0.165	1.00776e-12	80.2471
k75r	0.123	0.168258	1.94432
k89r	0.159	0.0157351	6.17256
k92r	0.301	1.07093e-13	90.8974
Intercept	0.068	NA	NA

Table 4: Linear model best fitted to 5 strategies in the reproduced tournament with $R^2 = 0.9440$

Strategies	Coefficients	p-value	F-value
k35r	-0.145	1.37332e-13	89.6766
k36r	0.128	0.190269	1.75437
k38r	-0.102	3.3876e-09	47.6646
k44r	0.084	8.26862e-10	52.7105
k46r	-0.292	0.004666	8.6278
k56r	0.235	2.29213e-09	49.0386
k64r	0.179	1.68964e-14	100.282
k68r	0.190	4.52731e-06	25.3442
k70r	0.145	1.00776e-12	80.2471
k75r	0.095	0.168258	1.94432
k76r	0.054	0.0257417	5.22576
k78r	0.057	9.40283e-07	29.7599
k80r	0.099	9.90171e-16	115.859
k84r	-0.172	3.35039e-18	151.839
k89r	0.090	0.0157351	6.17256
k91r	-0.082	3.07113e-10	56.3992
k92r	0.407	1.07093e-13	90.8974
Intercept	0.207	NA	NA

Table 5: Linear model best fitted to 17 strategies in the reproduced tournament with $R^2 = 0.9932$

The predictions of these models are shown in Figure 8.

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 great than 2.5). To be able to obtain a good model even for high performing strategies 12 seem to provide a good predictive model.

4 Revisiting the tournament

In this section, the tournament of [2] will be revisited. Indeed, a large amount of research has gone on since Axelrod's original work which include for example training of strategies using reinforcement learning [7] but also the discovery of Zero Determinant strategies [10]. 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 with an extra invitation

4.1.1 Known strategies

The tournament is run with every strategy of the Axelrod library. Every tournament (corresponding to each strategy) was run for 12500 repetitions. Table 6 shows the top ranking strategies.

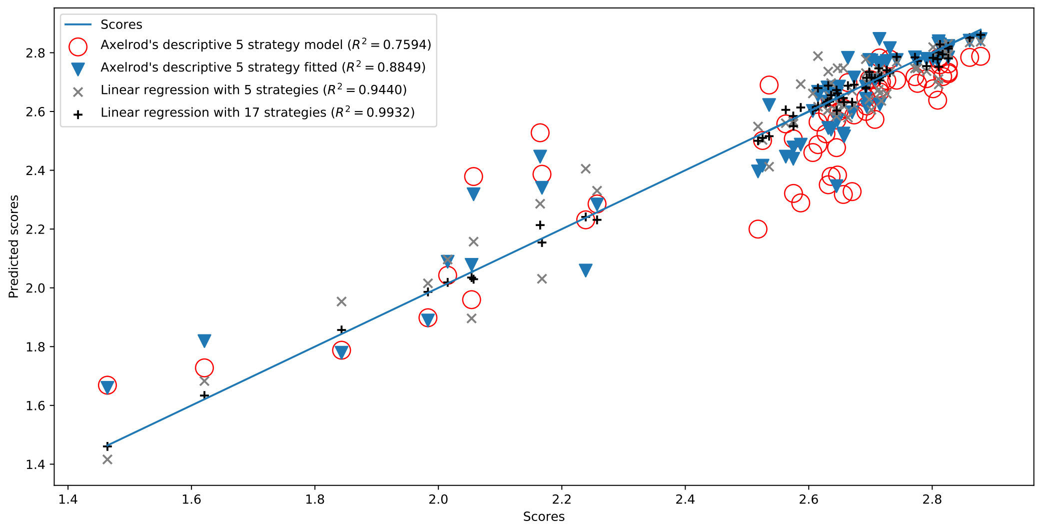


Figure 8: Predicting the performance of strategies using the 4 models discussed

Name	Cooperation Rate against opponents	Cooperation Rate from opponents	Library Rank	Rank	Score	Winner
Meta Majority Long Memory	0.897	0.910	80	2	2.866	k92r
Meta Majority	0.884	0.893	90	2	2.868	k92r
Firm But Fair	0.942	0.926	86	2	2.869	k92r
ZD-GTFT-2	0.941	0.929	52	2	2.876	k92r
Soft Joss	0.935	0.927	54	2	2.878	k92r
Adaptive Tit For Tat	0.923	0.922	84	2	2.880	k92r
PSO Gambler 2_2_2 Noise 05	0.866	0.880	16	3	2.833	k92r
Omega TFT	0.888	0.904	13	3	2.853	k92r
GTFT	0.946	0.928	59	3	2.855	k92r
Meta Majority Finite Memory	0.909	0.911	107	3	2.857	k92r
Forgiving Tit For Tat	0.925	0.916	81	3	2.858	k92r
Gradual	0.744	0.698	17	3	2.859	k92r
Resurrection	0.755	0.786	76	3	2.861	k92r
Spiteful Tit For Tat	0.857	0.877	22	7	2.816	k92r
Stein and Rapoport	0.864	0.882	50	7	2.816	k92r
Champion	0.954	0.911	89	12	2.794	k92r
ZD-GEN-2	0.918	0.897	63	12	2.799	k92r
Doubler	0.922	0.892	85	13	2.785	k92r
EugineNier	0.824	0.857	28	13	2.787	k92r
GrudgerAlternator	0.828	0.835	69	13	2.793	k92r

Table 6: Performance of extra strategy in Axelrod's original tournament

Figure 9 shows the rank of the extra strategy against it's rank in the library tournament.

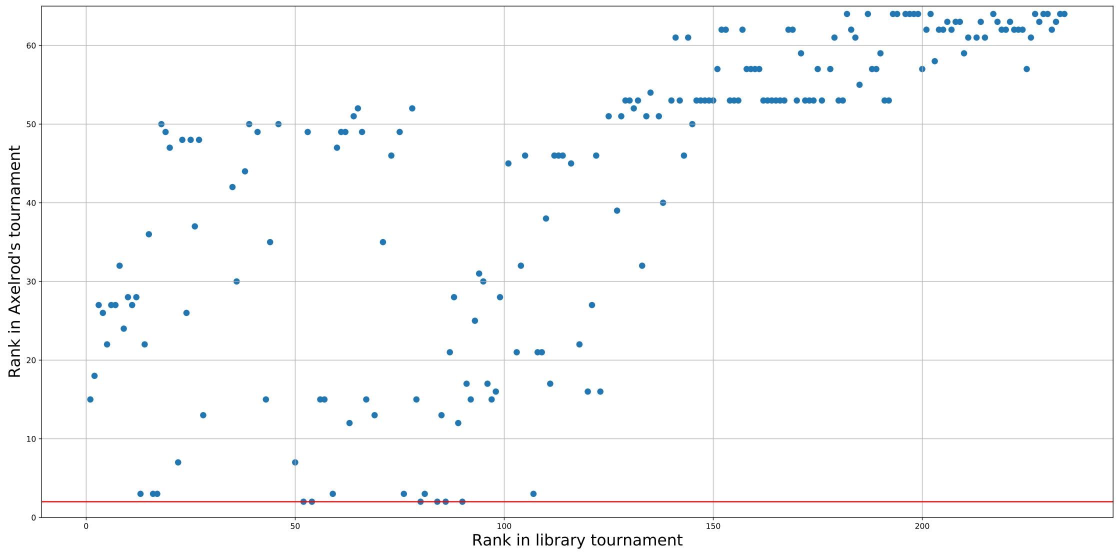


Figure 9: Ranks of extra strategy

4.2 Further tournaments

4.2.1 Running with extortion

Since the work of [10] a lot of interest has been shown to Zero Determinant strategies. In [11] a small tournament is presented pitting these against each other. Table 7 shows the rankings of the top 15 strategies when including all the Zero Determinant strategies from [11] over 30000 repetitions.

	Original Author	Scores	Rank	Original Rank	Reproduced Rank
ZD-GTFT-2	NA	2.8339	1	NA	NA
k42r	Otto Borufsen	2.8251	2	3	2
GTFT	NA	2.8185	3	NA	NA
k92r	Anatol Rapoport	2.8121	4	1	1
k49r	Rob Cave	2.7874	5	4	3
k75r	Paul D Harrington	2.7870	6	8	5
k44r	William Adams	2.7811	7	5	4
k61r	Danny C Champion	2.7722	8	2	12
k68r	Fransois Leyvraz	2.7679	9	12	10
k41r	Herb Weiner	2.7620	10	7	8
k32r	Charles Kluepfel	2.7582	11	10	6
k46r	Graham J Eatherley	2.7570	12	14	14
k72r	Edward C White Jr	2.7544	13	13	13
k35r	Abraham Getzler	2.7470	14	11	11
k84r	T Nicolaus Tideman and Paula Chieruzz	2.7449	15	9	7

Table 7: Top 15 strategies in the tournament composed of the original strategies and the Zero Determinant strategies from [11]

The overall cooperation rate of this tournament is 0.731 and the various cooperation rates are shown in Figure 10 shows the cooperation rates of each strategy (ordered by rank).

Figure 12 shows the pair wise cooperation rates.

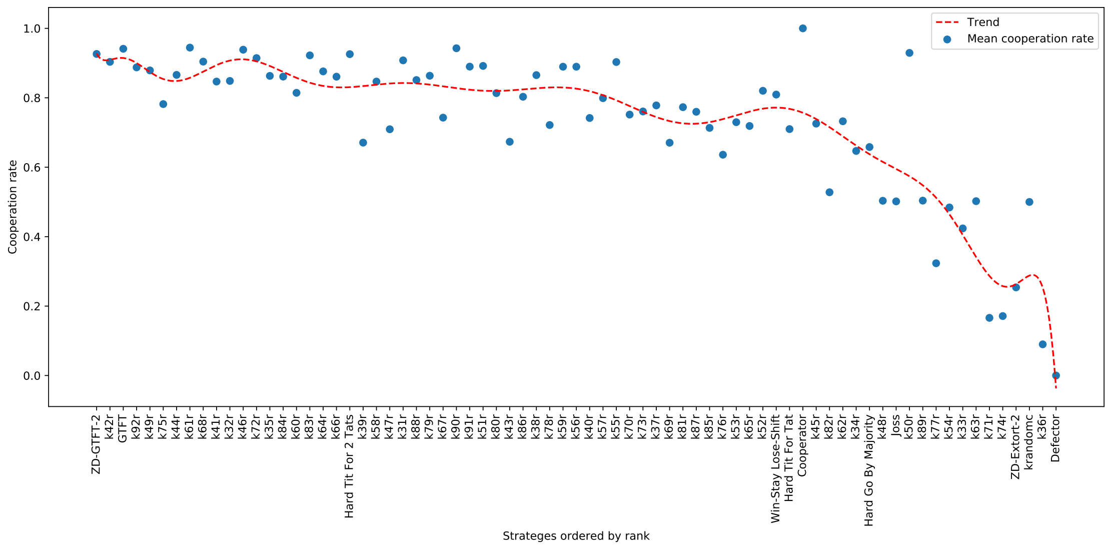


Figure 10: Cooperation rate versus rank for the Stewart and Poltkin tournament

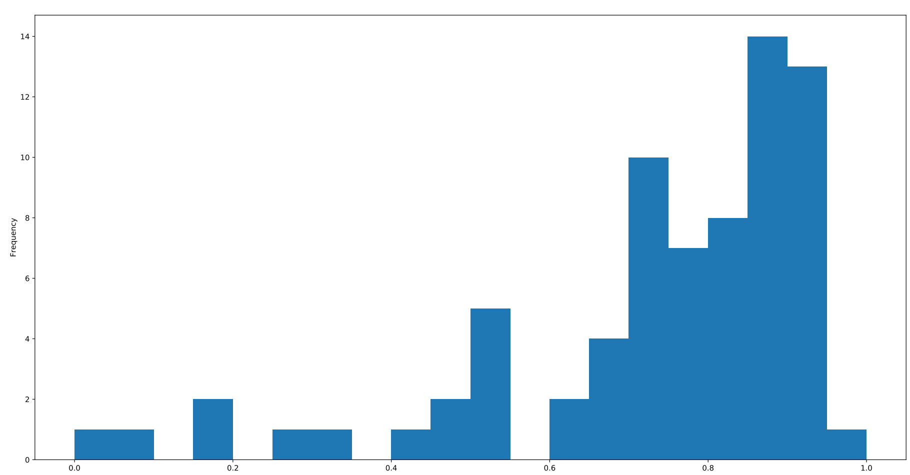


Figure 11: Distribution of cooperation rates for the Stewart and Plotkin tournament

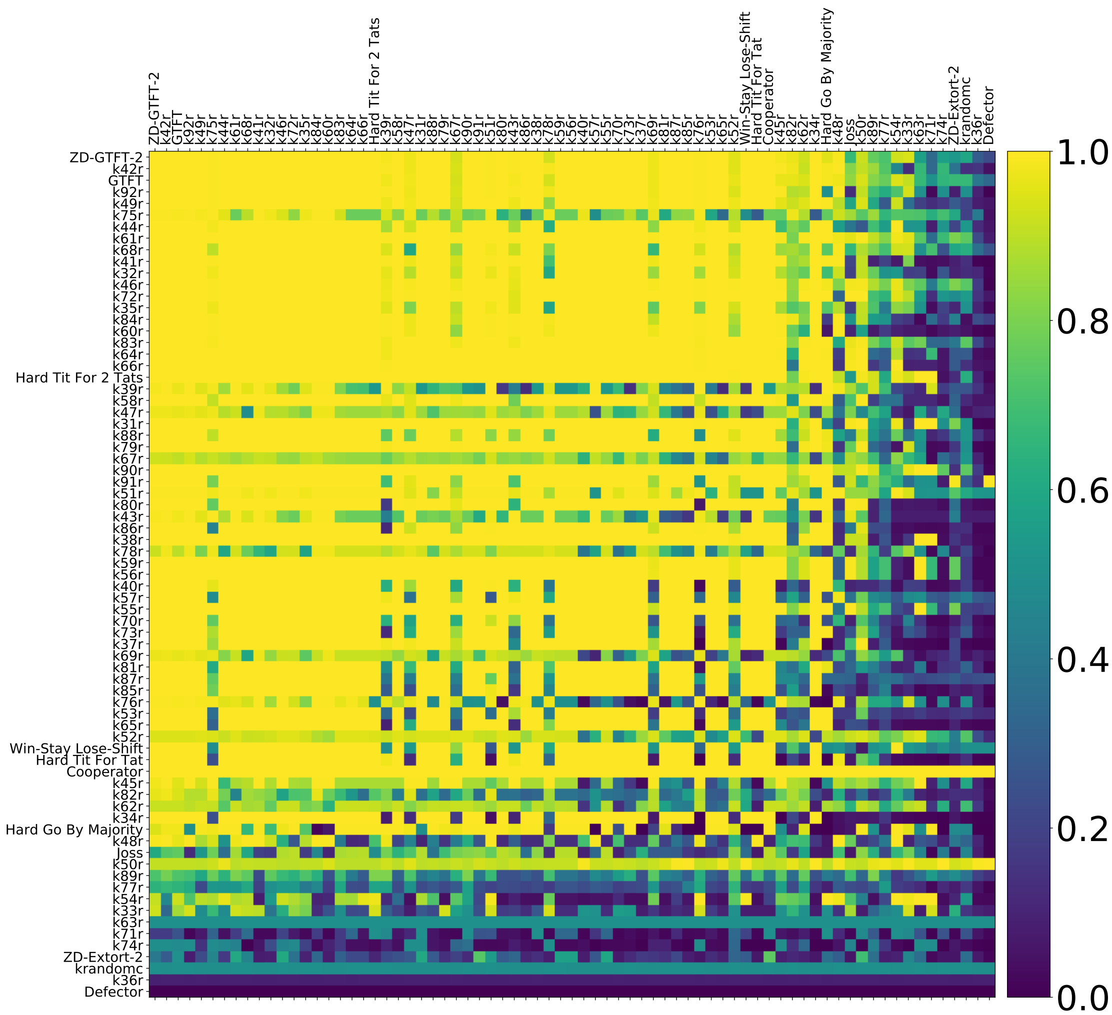


Figure 12: Cooperation rates between each pair of players (ordered by rank) for the Stewart and Plotkin tournament

4.2.2 Running a large tournament

Table 8 shows the rankings of the top 20 strategies when including all strategies over 20000 repetitions. The cooperation rate for the tournament that pits all the library strategies against each other (without the Fortran ones) is 0.624.

	Original Author	Scores	Rank	Library Rank	Original Rank	Reproduced Rank
EvolvedLookerUp2_2_2	NA	2.8345	1	1	NA	NA
Evolved HMM 5	NA	2.8210	2	2	NA	NA
Evolved FSM 16 Noise 05	NA	2.8057	3	5	NA	NA
Evolved FSM 16	NA	2.8015	4	4	NA	NA
PSO Gambler 2_2_2	NA	2.7997	5	3	NA	NA
Evolved ANN	NA	2.7898	6	7	NA	NA
Evolved ANN 5	NA	2.7891	7	6	NA	NA
Omega TFT	NA	2.7863	8	13	NA	NA
PSO Gambler Mem1	NA	2.7811	9	9	NA	NA
Evolved FSM 4	NA	2.7789	10	10	NA	NA
PSO Gambler 1_1_1	NA	2.7766	11	8	NA	NA
PSO Gambler 2_2_2 Noise 05	NA	2.7722	12	16	NA	NA
Gradual	NA	2.7637	13	17	NA	NA
Evolved ANN 5 Noise 05	NA	2.7634	14	11	NA	NA
DBS	NA	2.7618	15	12	NA	NA
Winner12	NA	2.7569	16	14	NA	NA
k85r	Robert B Falk and James M Langsted	2.7409	17	NA	33	37
Spiteful Tit For Tat	NA	2.7357	18	22	NA	NA
k80r	Robyn M Dawes and Mark Batell	2.7333	19	NA	36	35
k42r	Otto Borufsen	2.7311	20	NA	3	2

Table 8: Top 20 strategies in the tournament when using all available strategies

The overall cooperation rate of this tournament is 0.630 and the various cooperation rates are shown in Figure 13 shows the cooperation rates of each strategy (ordered by rank).

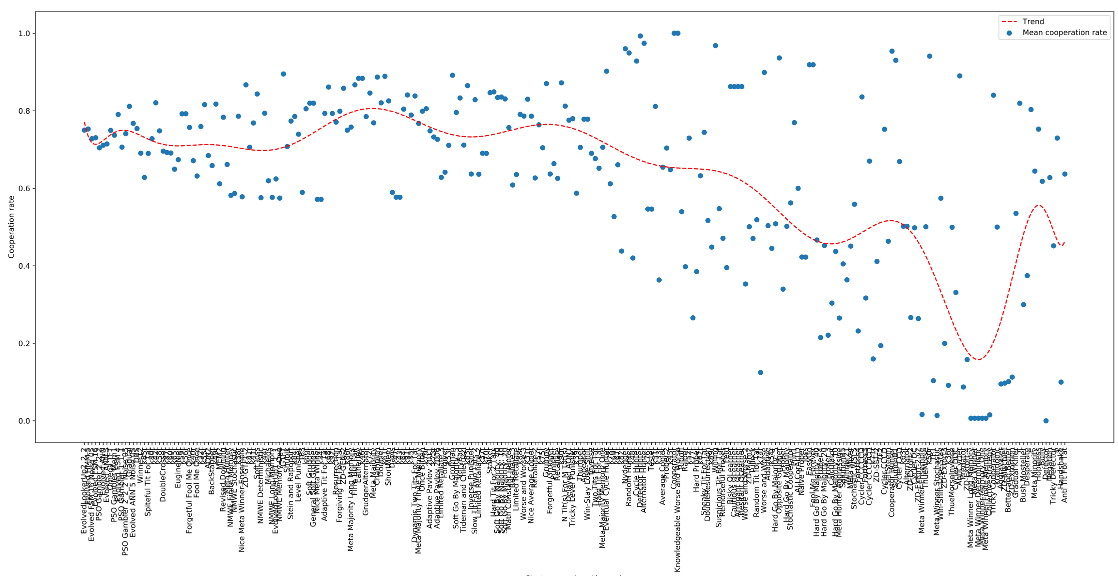


Figure 13: Cooperation rate versus rank for tournament with all available strategies

Figure 15 shows the pair wise cooperation rates.

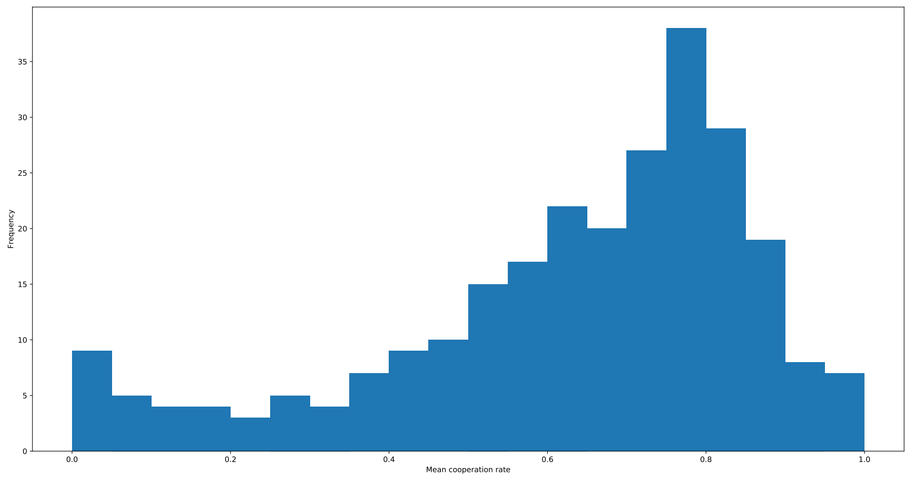


Figure 14: Distribution of cooperation rates for the full tournament.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

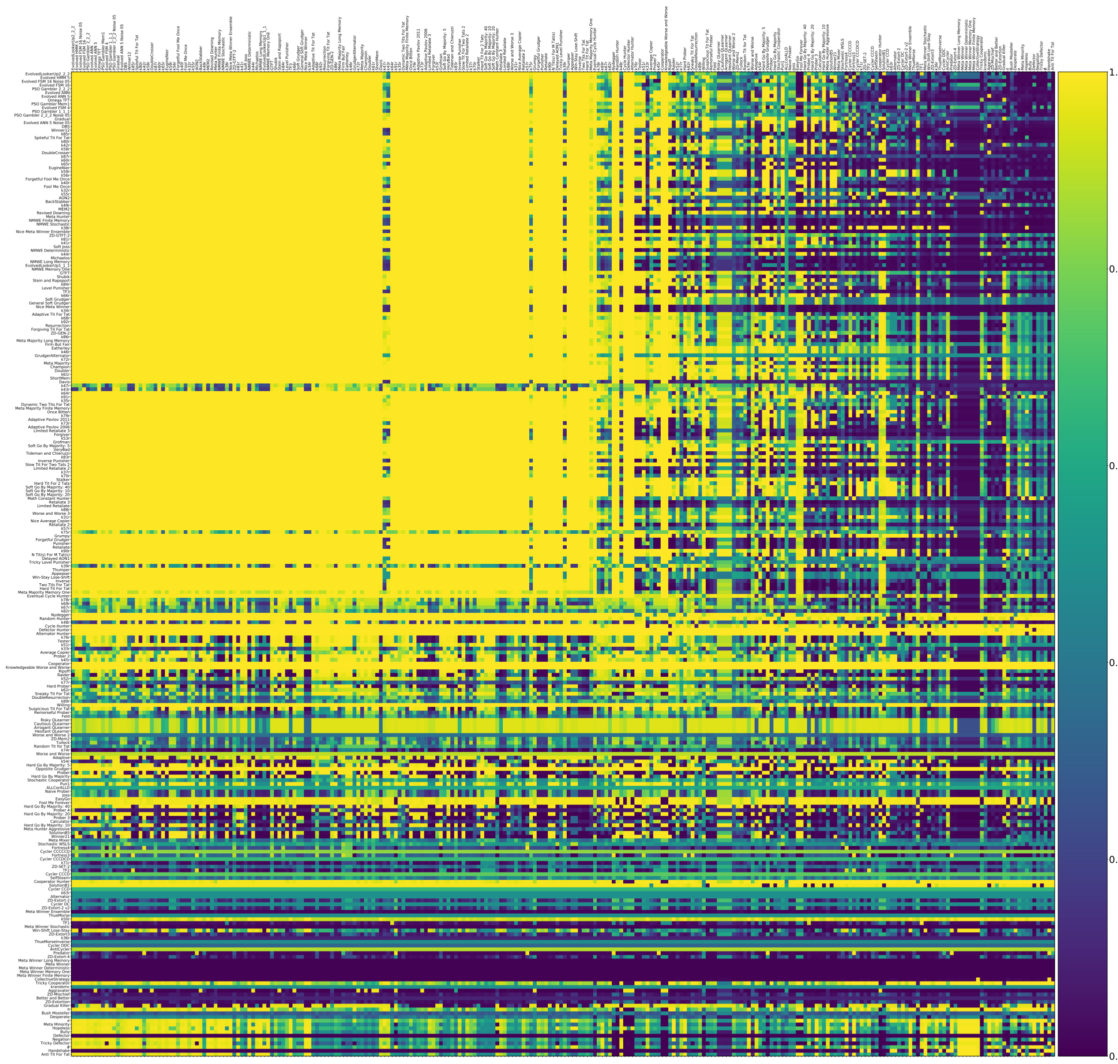


Figure 15: Cooperation rates between each pair of players (ordered by rank) for tournament with all available strategies

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