# Generating Solution Sequences for Strategies In The Prisoners Dilemma Using Genetic Improvement Algorithms

Toby Devlin - C1432293

# This Presentation

# 1 - Project Scope

# Project Description

*When playing a given Iterated Prisoner's Dilemma strategy as an opponent, what is the best ordered sequence of moves to play in order for us to obtain the highest possible average score per move across the game?*

*Are there any patterns in these solution sequences?*

# Example

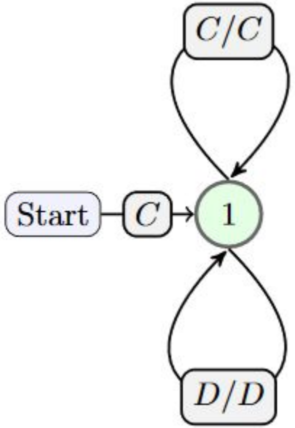
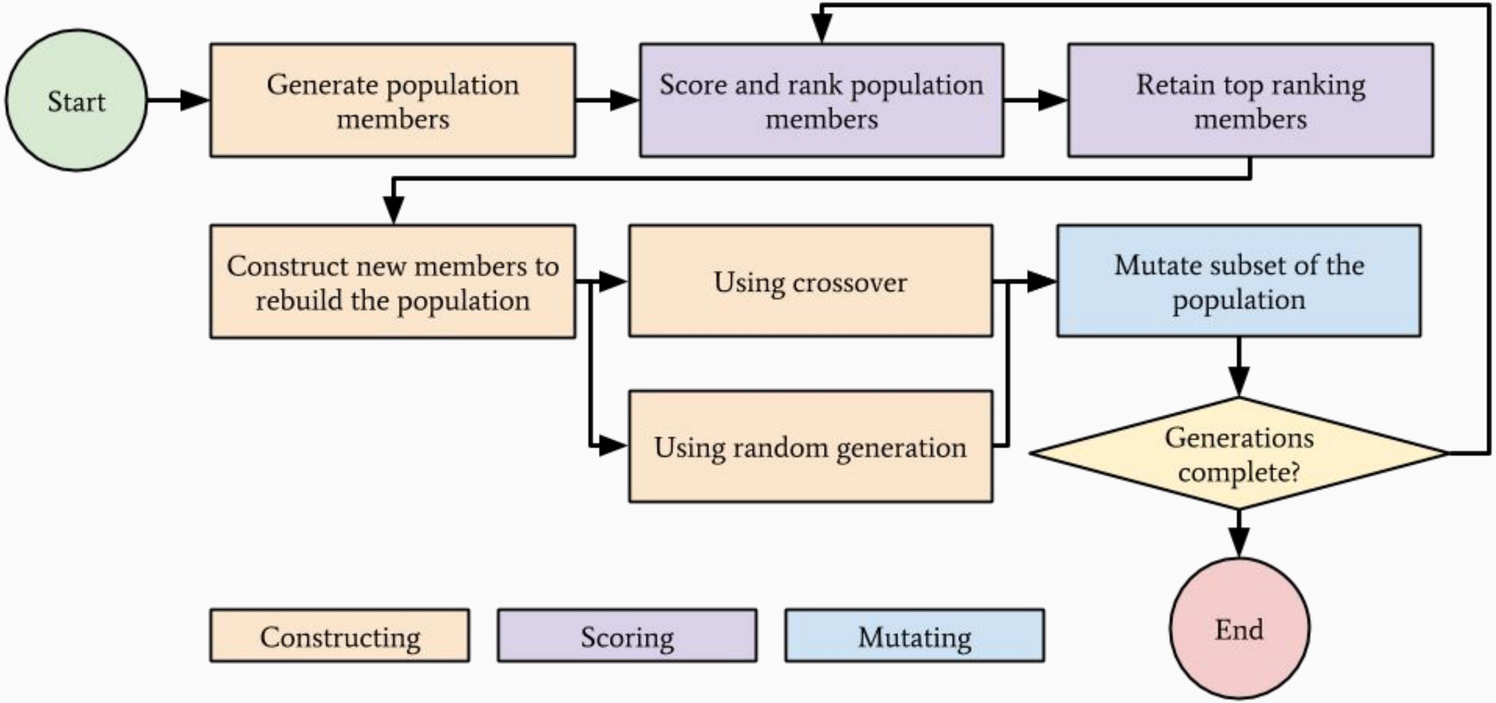


Figure 3.1: Finite State diagram of strategy Tit for Tat

| **Game 1** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
|---|---|---|---|---|---|---|---|
| Tit For Tat: | $C$ | $D$ | $D$ | $D$ | $D$ | 4 | 0.8 |
| Solution: | $D$ | $D$ | $D$ | $D$ | $D$ | 9 | 1.8 |
| **Game 2** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
| Tit For Tat: | $C$ | $D$ | $C$ | $D$ | $D$ | 7 | 1.4 |
| Solution: | $D$ | $C$ | $D$ | $D$ | $D$ | 12 | 2.4 |
| **Game 3** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
| Tit For Tat: | $C$ | $D$ | $C$ | $D$ | $C$ | 10 | 2.0 |
| Solution: | $D$ | $C$ | $D$ | $C$ | $D$ | 15 | 3.0 |
| **Game 4** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
| Tit For Tat: | $C$ | $D$ | $C$ | $C$ | $C$ | 11 | 2.2 |
| Solution: | $D$ | $C$ | $C$ | $C$ | $D$ | 16 | 3.2 |
| **Game 5** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
| Tit For Tat: | $C$ | $C$ | $C$ | $C$ | $C$ | 12 | 2.4 |
| Solution: | $C$ | $C$ | $C$ | $C$ | $D$ | 17 | 3.4 |
| **Game 6** | 1 | 2 | 3 | 4 | 5 | final score | per turn |
| Tit For Tat: | $C$ | $C$ | $C$ | $C$ | $C$ | 15 | 3.0 |
| Solution: | $C$ | $C$ | $C$ | $C$ | $C$ | 15 | 3.0 |

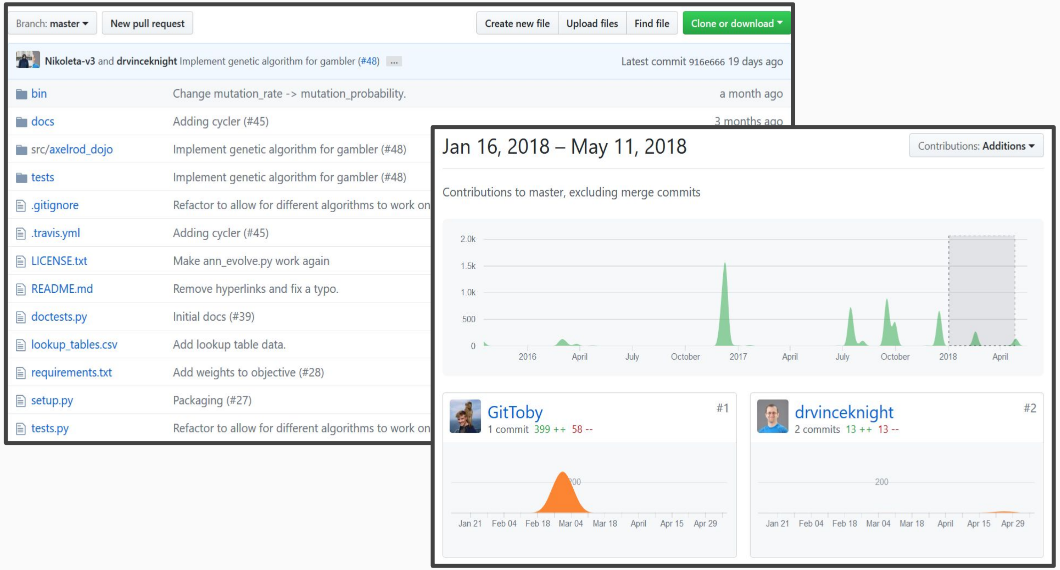
# 2 - The Genetic Algorithm and its Components

# Core Concept

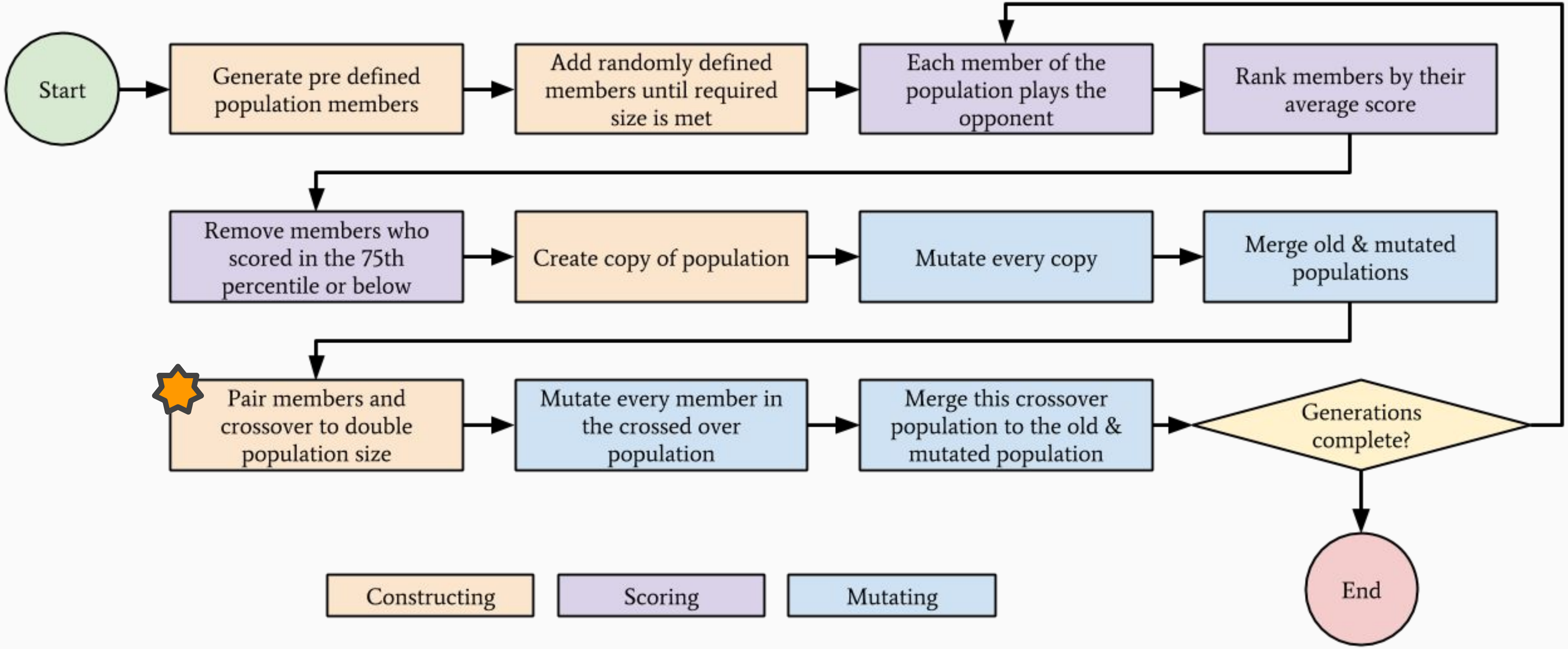


# Axelrod Dojo

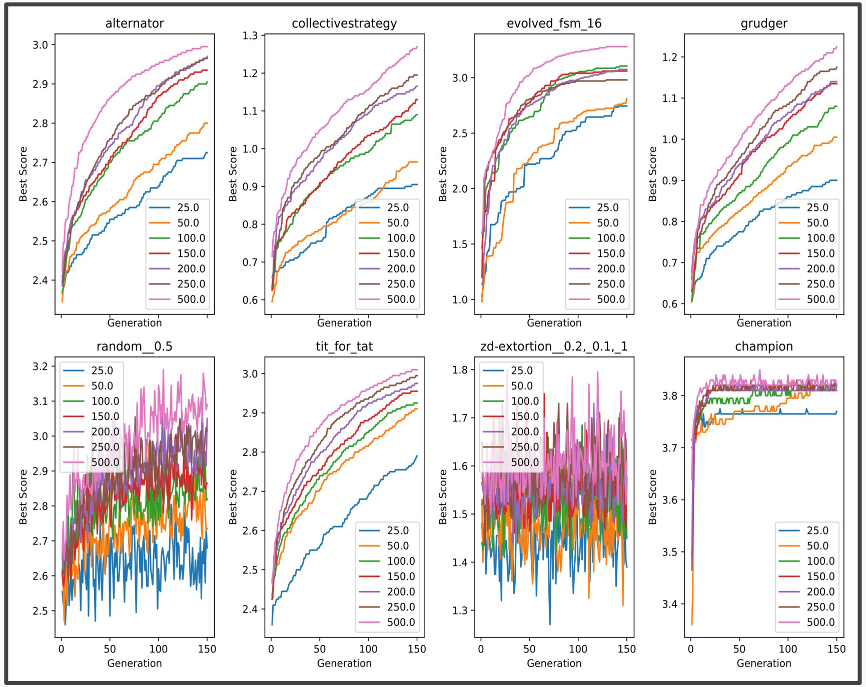
- Actively developed at Cardiff Uni
- Contains 3+ optimization techniques
- Allowed me to investigate VCS
- Flexible for game theory models

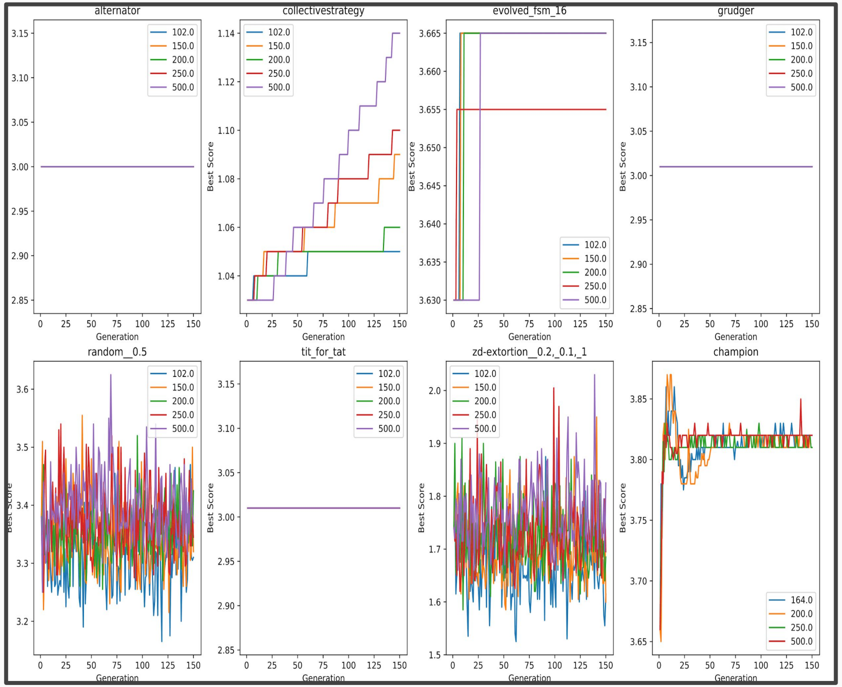


# Extended Algorithm



# Optimal Parameters



# 3 - Analysis and Outcomes

# Raw Results & Metadata

| | generation | score_mean | score_median | score_pop_var | score_range | best_score | best_sequence |
|---|---|---|---|---|---|---|---|
| **2401** | 2 | 2.80574 | 2.8950 | 0.338914 | 2.635 | 3.140 | DDCDDCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC... |
| **2402** | 3 | 2.80780 | 2.9600 | 0.447188 | 2.850 | 3.375 | CCCCCCCCCCCCCCCCCCCCCCCCCCCDDDCDDCDCDDCCDCCD... |
| **2403** | 4 | 2.83276 | 3.0000 | 0.529338 | 2.865 | 3.390 | DCDDDDDDCDDCCDDCDDDCCDDCDDCCDCDDCDDCDCCDCDCCCD... |
| **2404** | 5 | 2.94730 | 3.1300 | 0.609064 | 2.895 | 3.425 | CCCCCCDDDCDCDCCCCDDDCDDCDDDDCDDCCDCDCCDDDDDCDD... |
| **2405** | 6 | 3.06512 | 3.2525 | 0.601824 | 2.885 | 3.445 | DCDDDDDDCDDCCDDCDDDCCDDCDDDDCDCCCDCDCCDDDDDCDD... |

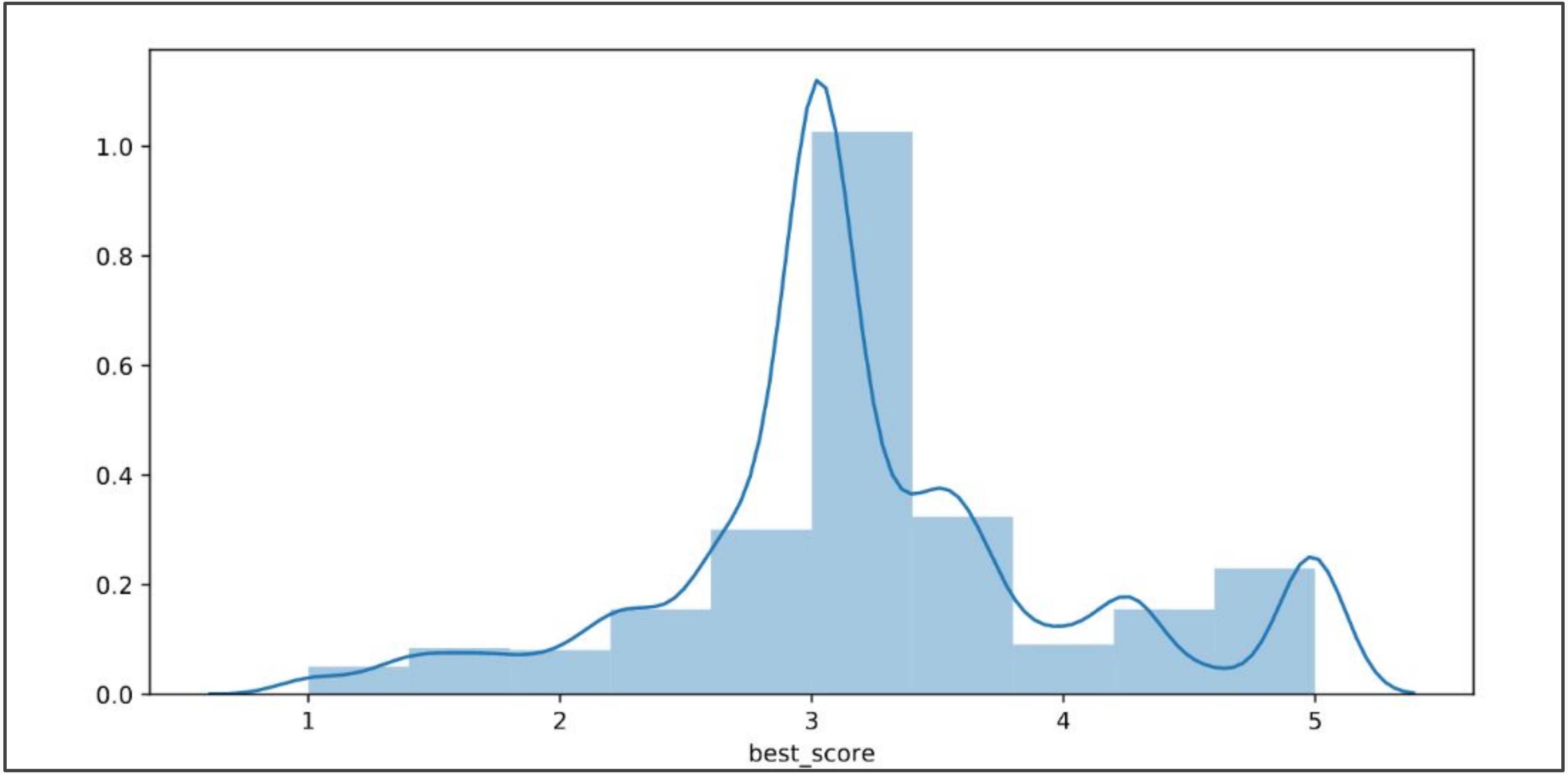
| | inspects_source | long_run_time | makes_use_of | manipulates_source | manipulates_state | memory_depth | stochastic | opponent_name |
|---|---|---|---|---|---|---|---|---|
| $\phi$ | False | False | {} | False | False | inf | False | $\phi$ |
| $\pi$ | False | False | {} | False | False | inf | False | $\pi$ |
| $e$ | False | False | {} | False | False | inf | False | $e$ |
| **ALLCorALLD** | False | False | {} | False | False | 1 | True | ALLCorALLD |
| **Adaptive** | False | False | {game} | False | False | inf | False | Adaptive |
| **Adaptive_Pavlov_2006** | False | False | {} | False | False | inf | False | Adaptive_Pavlov_2006 |

# Interesting results

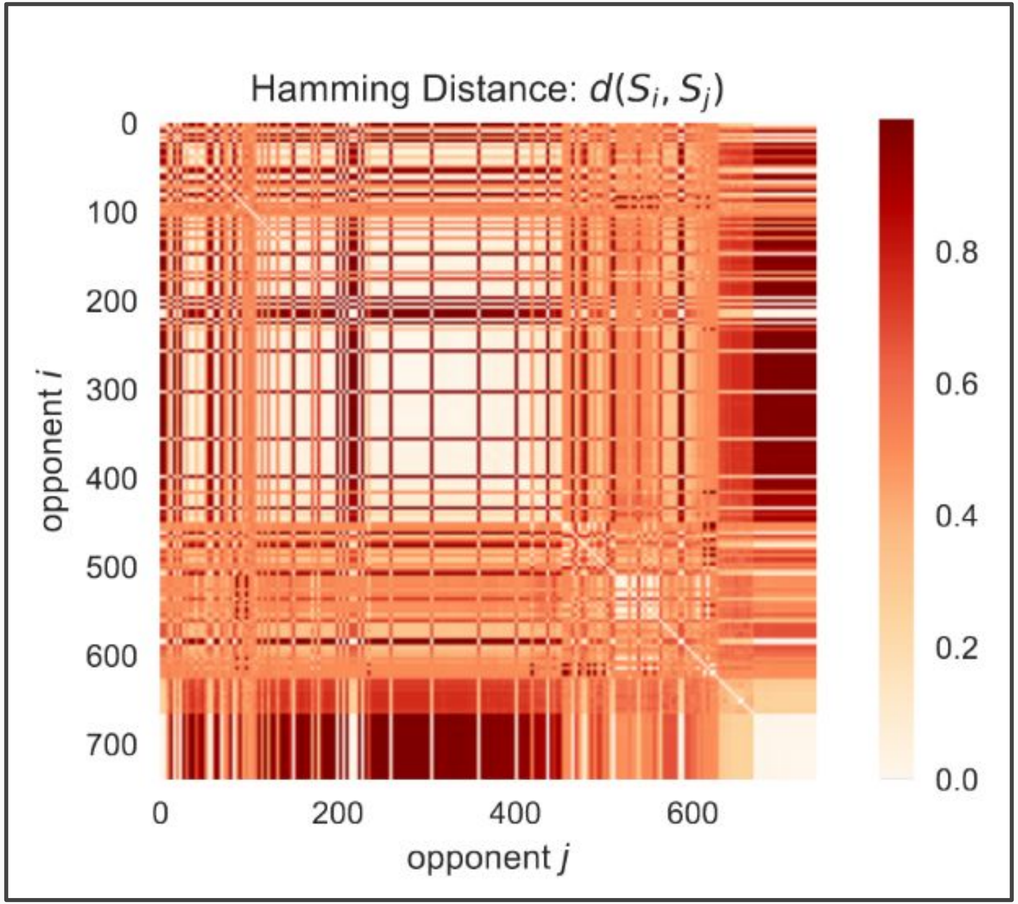
1 - Effectiveness of solutions

2 - Difference with expected results

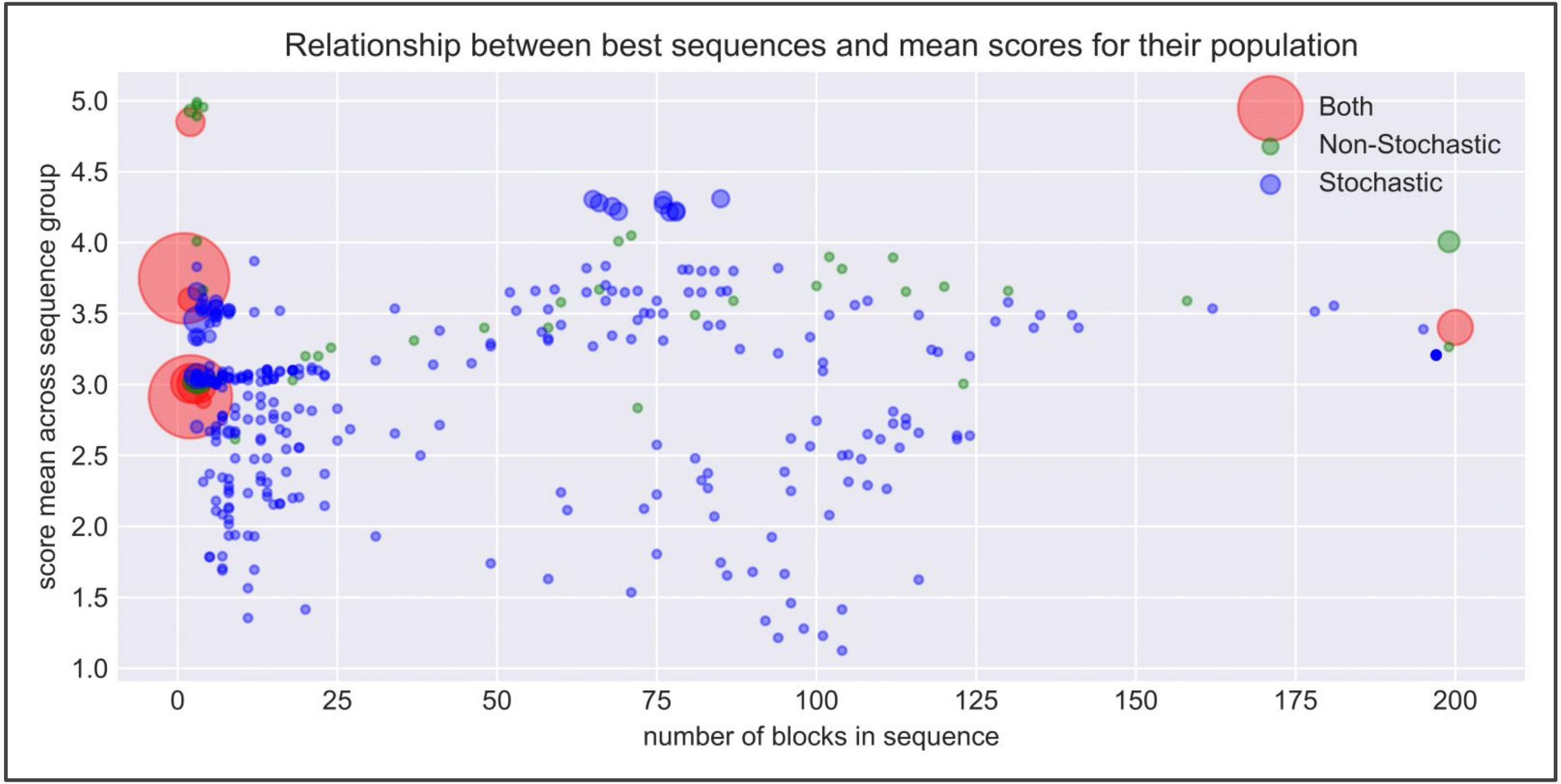
3 - Trends in complexity




Effectiveness of solutions - histogram of score distributions




Difference with expected results - distance matrix


Relationship between best sequences and mean scores for their population
score mean across sequence group
number of blocks in sequence
Both
Non-Stochastic
Stochastic
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
25
50
75
100
125
150
175
200

# 4 - Final Conclusions & Discussion

# Subject Specific Knowledge Gained

# Subject Specific Knowledge Gained

## **Unclear Conclusions**

# Subject Specific Knowledge Gained

**Unclear Conclusions**

**Nuanced Problem**

# Subject Specific Knowledge Gained

**Unclear Conclusions**

**Nuanced Problem**

**Initial Populations**

# Subject Specific Knowledge Gained

**Unclear Conclusions**

**Nuanced Problem**

**Initial Populations**

**Efficient Algorithms**

# General Knowledge Gained

# General Knowledge Gained

Complex Problems

- Approximation methods
- Leveraging code

Data Analysis

- Distributed computing
- Techniques

# General Knowledge Gained

Complex Problems

- Approximation methods
- Leveraging code

Independent learning

- Citing others work
- Identifying useful sources

Writing Reports

- Overcoming personal difficulties
- Managing time

Data Analysis

- Distributed computing
- Techniques

# Value Added With This Report

zenodo Search Upload Communities

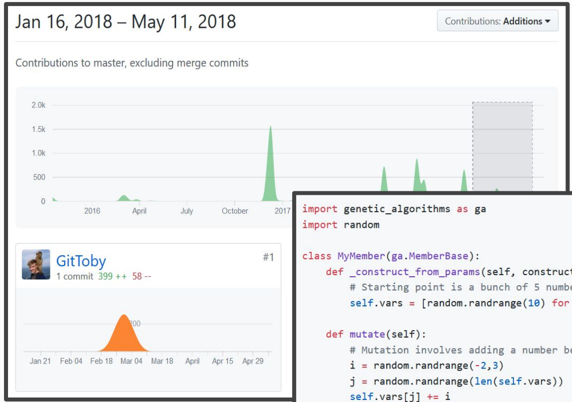
June 4, 2018 Dataset Open Access

Iterated Prisoner's Dilemma Best response data for games of length 200

Toby Devlin

- D55,1,14,1,2,1,1,1,10,1,9,1,2,1,1,1,2,1,5,2,1,1,1,2,7,2,4,1,1,3,2,2,1,1,1... for 1 opponent(s):
  - Bush_Mosteller@_6* (score: 3.67)
- D57,2,1,1,5,1,1,3,1,1,2,2,13,1,1,1,1,1,6,2,3,2,1,1,2,2,4,1,5,2,3,1,1,1,3... for 1 opponent(s):
  - Bush_Mosteller@_2* (score: 3.59)

- C196,4 with 3 opponents
- C194,6 with 1 opponent
- C193,7 with 2 opponents
- C100,100 with 8 opponents
- C5,195 with 3 opponents
- C2,1,1,196 with 1 opponent
- C2,198 with 2 opponents
- C1,1,1,1,... with 17 opponents
- C1,1,1,197 with 1 opponent
- C1,2,1,196 with 1 opponent
- C1,199 with 11 opponents
- D1,198,1 with 21 opponents
- D1,196,3 with 1 opponent
- D1,194,5 with 2 opponents
- D1,4,195 with 4 opponents
- D1,3,196 with 4 opponents
- D1,2,197 with 9 opponents
- D2,197,1 with 3 opponents




$$
\begin{aligned}
C1,4,3,2 &= CDDDDCCCDD \\
D(1,1)^5 &= DCDCDCDCDC \\
C1,(2,1)^2,2,1 &= CDDCDDCDDC \\
\{D[i,5-i]\} \quad i \in [2,4] &= \{DDCCC,\ DDDCC,\ DDDCC\}
\end{aligned}
$$

```python
import genetic_algorithms as ga
import random

class MyMember(ga.MemberBase):
    def _construct_from_params(self, construction_parameters=None):
        # Starting point is a bunch of 5 numbers [0-9]
        self.vars = [random.randrange(10) for _ in range(5)]

    def mutate(self):
        # Mutation involves adding a number between -2 and 2 to a random variable
        i = random.randrange(-2,3)
        j = random.randrange(len(self.vars))
        self.vars[j] += i
```

| Rank | Name | Median score | Cooperation rating | Wins |
|---|---|---|---|---|
| 0 | Cycler: C1,199 | 4.9625 | 0.005 | 4.0 |
| 1 | Tricky Defector | 2.23625 | 0.2475 | 1.5 |
| 2 | SolutionB1 | 1.7675 | 0.71775 | 2.0 |
| 3 | Defector Hunter | 1.508125 | 0.995 | 0.0 |
| 4 | Willing | 1.5 | 0.849 | 0.5 |

Table 6.3: Table of test opponents

# Questions?