Using agent-based simulations to investigate rates of cooperation in small populations and verify the findings of Santos, Santos and Pacheco

Konrad Cybulski

Final report for
FIT1041 Research Project

Faculty of Information Technology
Monash University
Australia
21/10/2016

Contents

1 Background

In order to understand our behaviour, our willingness and ability to cooperate with those around us, it is vital that we can create simulations to model the dynamics of populations and the interactions of agents within them. It is also important to understand the way cooperation and defection evolve in both small and large populations.

In order to investigate cooperation, game theorists often use a situation known as the prisoner's dilemma. Where given two players, they both have the choice to cooperate or defect. Cooperation implies paying a cost in order for the other player to receive a benefit. Defection implies paying no cost in the hope that the other player will cooperate and this player will receive a benefit at no cost. Thus the payoff for a player (the overall gain/loss as a result of the choices made by both players) is defined for the following three options:

1. Cooperation - Cooperation — Both players receive a payoff of the benefit minus the cost.
2. Cooperation - Defection — The cooperator pays the cost while the defector receives the benefit.
3. Defection - Defection — Both players pay no cost and receive no benefit, resulting in a payoff of zero.

In order to investigate the rate of cooperation in a population, it's required to let agents in the population play the prisoner's dilemma game against other agents in order to determine their average fitness as the their average payoff over the course of a number of games.

With the addition of a reputation system, and methods of assigning reputation (using social norms), we derive four strategies:

1. Always Defect.
2. Always Cooperate.
3. Discriminator — The individual will cooperate with another individual with a **good** reputation and defect against somebody with a **bad** reputation
4. Paradoxical Discriminator — The individual will cooperate with another individual with a **bad** reputation and defect against somebody with a **good** reputation.

The fitness function determines the fitness of some individual x against some other individual y and then using the *Fermi update rule* we determine the probability of individual x imitating the strategy of individual y.

Assigning reputations is done utilising a social norm. When individual X and Y interact, the social norm determines how we assign X a reputation based on both its action and the reputation of Y. Similarly the new reputation of Y is based on its action and the reputation of X. In order to investigate the rate of cooperation in a given population we explore possible social norms and see how each of these social norms can foster cooperation among the populace.

While the outcome of a large number of interactions can be determined, due to the stochastic nature of the simulation, there exists error. The errors that can occur include:

1. Private Assessment Error (χ) — The probability of wrongly assessing the reputation of another individual.
2. Execution Error (ϵ) — The probability that despite the strategy of an individual, they will be unable to pay the cost and will thus defect. This is to model the fact that in a given interaction, one individual may lack the resources to cooperate.
3. Reputation Assignment Error (α) — The probability of assigning a reputation to an individual that is opposite to that denoted by the social norm.
4. Reputation Update Probability (τ) — The probability of updating the reputation of any given individual after an interaction.

In addition to these errors, there exists a chance for any given individual to mutate to another random strategy. The likelihood of mutation is defined as μ, and for the simulations is set to a value of $(10 \cdot Z)^{-1}$ where Z is the population size. The simulation is ran for a large number of generations with each generation giving the opportunity for every individual in the population to mutate or imitate.

The main research paper on which this project is based is "*Social Norms of Cooperation in Small-Scale Societies*" by Santos, Santos, and Pacheco. In the paper a number of simulation parameters and methods are introduced and those specifications are replicated in this project in order to verify the results of the simulations performed in the paper.

2 Introduction

The primary aim of this project is to replicate the findings of Santos, Santos, and Pacheco which will allow further exploration into more detailed models of the prisoner's dilemma game within populations. The paper outlines simulation parameters based on equations to model the process in which the prisoner's dilemma game is played over time in a population.

In order to allow for greater research to be done in this area based on the simulation developed to verify the mathematical model developed by Santos et al. it is necessary that the simulation can be recreated and tested under similar variables and constraints to verify their results.

The four social norms investigated in this verification include two norms leading to extremely low cooperation indexes and the only two leading to high rates of cooperation. The social norms are defined as Stern Judging (high cooperation rate), Simple Standing (high cooperation rate), Image Score (low cooperation rate) and Uniform Zero (low cooperation rate).

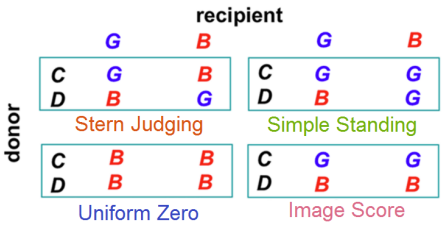


Figure 1: Social norms *Stern Judging*, *Simple Standing*, *Zero Uniform* and *Image Score*.

The source code for the algorithms can be found at a GitHub repository along with a copy of this report and all associated resources related to this project[7].

3 Methodology

The application developed was initially programmed in pure Python and based on the pseudocode provided by Santos, Santos, Pacheco in the supporting material of their paper. In developing both the Python program, the focus initially was on optimization of the program due to the intractability of the algorithm as defined in the paper for large population sizes (Z), large generation numbers (G) and large numbers of runs (R). The total time complexity for any given simulation being $O(Z^2 \cdot G \cdot R)$. Utilising numpy [3] a number of scalar optimisations were able to increase the efficiency of the program in its Python implementation. The majority of the optimisation however was done by utilising Cython. The method of vectorizing the program took was aimed at parallelizing the stage at which one individual interacted with a number of other individuals in order to determine its average fitness over a number of interactions. All optimizations made at any point in the algorithm were tested to ensure complete correctness and verify the algorithm itself had been altered functionally in any way.

3.1 Vectorization of the fitness function

The fitness function, the function to determine the average fitness of an individual over the course of a series of prisoner's dilemma games with other individual

in the population, was the aim of the optimization. The vectorization was required to compute a series of vectors in order to computer the payoff for a given agent X and update the reputations of X and a vector of other agents (the tournament vector). The following computations were required:

1. Vector of actions taken by a given agent X against each opposing individual in the tournament vector.
2. Vector of actions taken by each agent in the tournament vector.
3. Vector of the reputation of X after each interaction.
4. Vector of the reputation of each agent after each interaction with X.

The equations defining the actions of both agent X (C_x) and each agent in the tournament vector (C_y) are defined as:
A_i = the vector of size two defining the possible actions of agent i against another agent Y, this vector contains the action if Y has reputation G and if Y has reputation B.
$(A_i)_G$ = the action of agent X if the reputation of some agent Y is G (0 is defection, 1 is cooperation).
$(A_i)_B$ = the action of agent X if the reputation of some agent Y is B (0 is defection, 1 is cooperation).
R = the reputations of the agents defined in the tournament vector (0 is B, 1 is G).
R_x = the reputation of agent X (0 is B, 1 is G).

$$\overrightarrow{C_x} = \overrightarrow{(A_x)_G} \cdot \overrightarrow{R} + \overrightarrow{(A_x)_B} \cdot (1 - \overrightarrow{R})$$
$$\overrightarrow{C_y} = \overrightarrow{(A_y)_G} \cdot \overrightarrow{R_x} + \overrightarrow{(A_y)_B} \cdot (1 - \overrightarrow{R_x})$$

The constraints of this vectorization however are such that for any given tournament vector there must be no duplicates. Due to the parallel nature of the interactions, interactions are performed with incorrect or antiquated knowledge of the reputation of an agent who appears more than once. In order to perform the fitness function correctly with duplicate agents in the tournament sample the algorithm must iterate through each agent. Thus the vectorization requires that a given tournament sample must not contain duplicate agents.

The parallelization of reputation update was done utilizing *numpy* features.

3.2 Analysis of optimization

The optimizations made to increase the program's time complexity and execution time were measured utilizing *Jupyter Notebook*.

4 Results

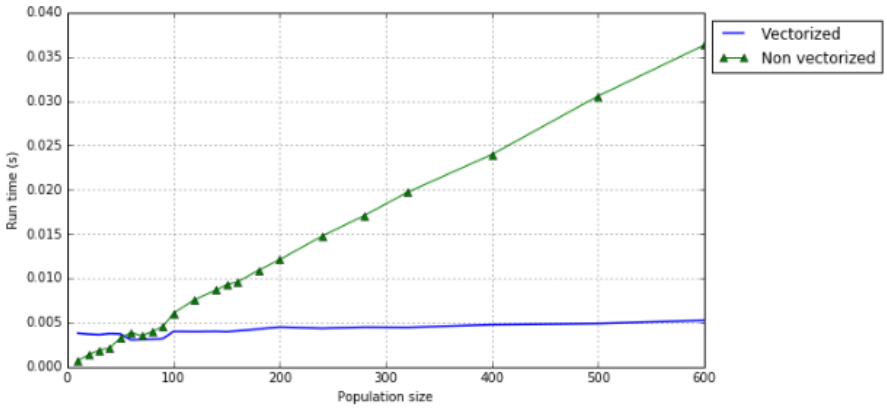


Figure 2: Run time of the fitness function for both pure Python implementation and vectorized Python implementation for $10 \leq Z \leq 600$.

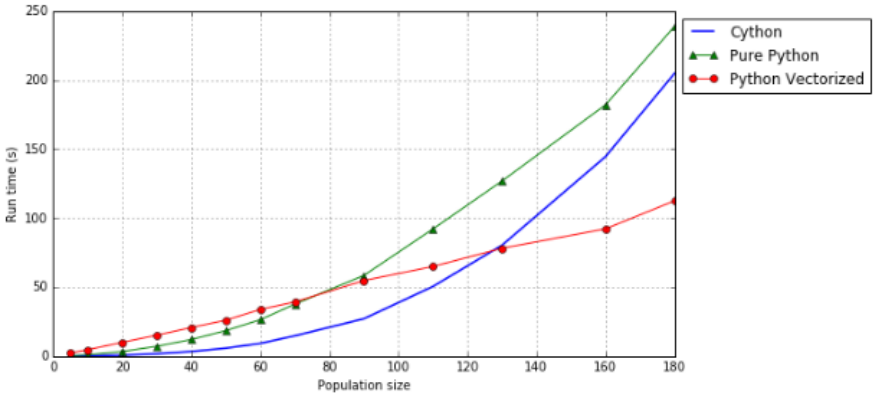


Figure 3: Run time of entire simulation for G=$3 \cdot 10^2$ and $5 \leq Z \leq 180$ of pure Python, vectorized Python, and Cython versions of the simulation.

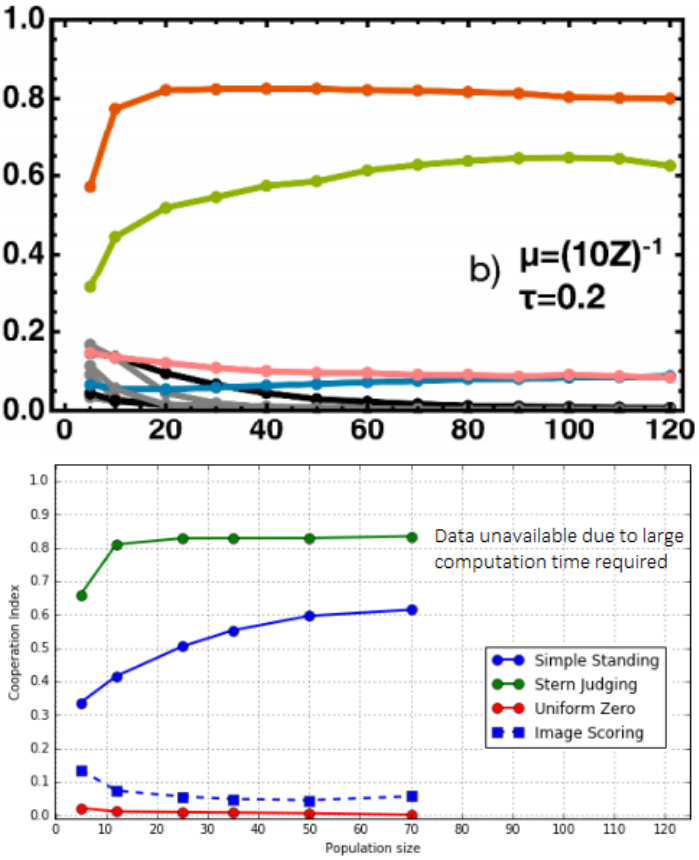


Figure 4: Results of Santos Santos Pacheco[2] (Top) and the simulation specified in this report (Bottom) under parameters $\epsilon = 0.08, \chi = 0.01, \alpha = 0.01, \mu = (10 \cdot Z)^{-1}$.

5 Discussion

5.1 Vectorization of the fitness function

The pure Python vectorization of the simulation aimed to reduce the overall time complexity of the fitness function from O(Z) to O(1). While all vector operations performed may take up to O(N) time complexity at a lower level, due to the large overhead of Python function calls the vectorized function reduces the number of function calls from Z to 1.

The fitness function itself reduces to almost O(1) for values of Z within the testing range ($Z \in [10, 600]$) (*Figure 2*). The results show that the increase in running time for the vectorized function is marginal for large increases in population size. While the vectorized fitness function is slower than the non-vectorized function for population sizes below 70, for very large population sizes, it allows the simulation to be tractable and brings the total simulation time complexity

to $O(Z \cdot G \cdot R)$.

As evident in *Figure 3*, the vectorized function performs much worse on smaller population sizes compared to both the pure Python and Cython programs, however past the mark of Z=130, the vectorized function performs much faster with a greatly improved rate of increase. However we see in the run time for the fitness function itself, it is far quicker for populations less than 70, however we only see a reduction in run time for the entire simulation for population sizes above 130. One possible reason for this discrepancy is that there is a lot of computation time in the vectorized function spent creating new arrays to contain existing data. The process of array creation is very CPU heavy and as a result, in order to further speed up the program, code should be written to run in place with no added space complexity.

This reduction in time complexity allows further work to be done on larger population sizes and to aid in the understanding of reputation and its effect on cooperation in population sizes of greater than 130.

5.2 Verification of results

Simulations were tested using $\epsilon = 0.08, \chi = 0.01, \alpha = 0.01, \mu = (10 \cdot Z)^{-1}, \tau = 1.0$, and contrasted against the data provided in the supporting material of Santos, Santos, Pachecho given $5 \leq Z \leq 70$. The four social norms investigated were Stern Judging, Simple Standing, Uniform Zero and Image Score Each of the results was the average of R=100 runs and G=$3 \cdot 10^5$ generations.

Figure 4 (bottom) shows the results of the simulation run as part of this report compared to those provided in the paper. The general trend for Stern Judging in the paper is a steep change between population size of 5 and 10 and converges to 0.82 very quickly and maintains this cooperation index for the entirety of the sample population sizes. We see this trend as an exact match to the results of this report. For Simple Standing we see a gradual increase in cooperation and reaches a stable value around 0.60, this is evident in both the paper's results and those of this report. For the Uniform Zero social norm, we see an almost zero cooperation index for all population sizes. Image Scoring provides a higher cooperation for lower population sizes however goes towards zero maintaining a low cooperation index of around 0.1, as shown in both graphs.

The program produced results that closely reflect those of the paper for the population sizes tested. Thus the simulation produced verifies the results produced in the paper according to the simulation parameters detailed in the supporting material given the sample of population sizes tested in this report.

5.3 Further work

With a simulation developed that is both fast for small population sizes and tractable for large populations, there is a platform for testing new models with relatively small changes to the simulation itself. As shown in section 6 (*Extension*) there lies little difficulty in implementing new functionality to test both different parameters and different processes by which we perform the simulation.

Regarding large population sizes, there exists work that can be done regarding converting the vectorized Python simulation to a Cython or C/C++ program to improve the speed again. Further improvements can be made to the program to increase the speed of the simulation including converting of the Cython program to pure C/C++ in addition to the vectorized simulation. This adaptation will most likely further improve the run time of the simulation for small population sizes ($Z \leq 130$) in a similar fashion to that shown in *Figure 3*.

6 Extension

The method by which Santos, Santos and Pacheco incorporate private assessment errors of the reputation of any other agent in the simulation involves the use of a single variable χ which determines this error for all agents. The error χ determines the probability that a given agent will wrongly assess the reputation of another agent in a given interaction. The extension on this project aimed to investigate the effect of dynamic assessment errors and the delay of reputation information propagation on rates of cooperation. The process by which this variable assessment error is incorporated into the simulation is outlined in *Methodology (6.1)*

While it is expected that the cooperation index for smaller population sizes will stay relatively constant for a given social norm in comparison to a constant private assessment error, for larger population sizes, this propagation delay may have a greater impact on the rate of cooperation.

6.1 Methodology

The likelihood of wrongly assessing the reputation of a given individual is defined as a function of the time since the last update of that individual's reputation. In turn this process emulates the delay that exists between the creation of new information about an agent's reputation and when any other agent in the population comes into that knowledge. The way that this is done, while maintaining the stochastic framework of the simulation itself, is by utilizing a function for the private assessment error for a given individual i (denoted further as χ_i) and keeping track only of the number of time steps (interactions, denoted as t_i) since agent-i's reputation was *changed*. While we may update an agent's reputation after every interaction, t_i is only reset to 0 when R_i has been changed to a value other than its current value.
The private assessment error the function is defined as:

χ_{min} = the minimum private assessment error.
Z = the population size.
R_s = the rate of information spread.
For $R_s \in [0, 1]$, R_s of value $\boldsymbol{a}$ signifies that at each interaction every agent with knowledge of agent-i's reputation will give this information to another agent with probability $\boldsymbol{a}$.
For $R_s > 1$, R_s signifies that at each interaction every agent with

knowledge of agent-i's reputation will give this information to R_s other agents.

$$\chi(t) = 1 - \chi_{min} - \frac{(1+R_s)^t}{Z}, \qquad \chi_{min} \leq \chi(t) \leq 1 - \chi_{min}$$

6.2 Results

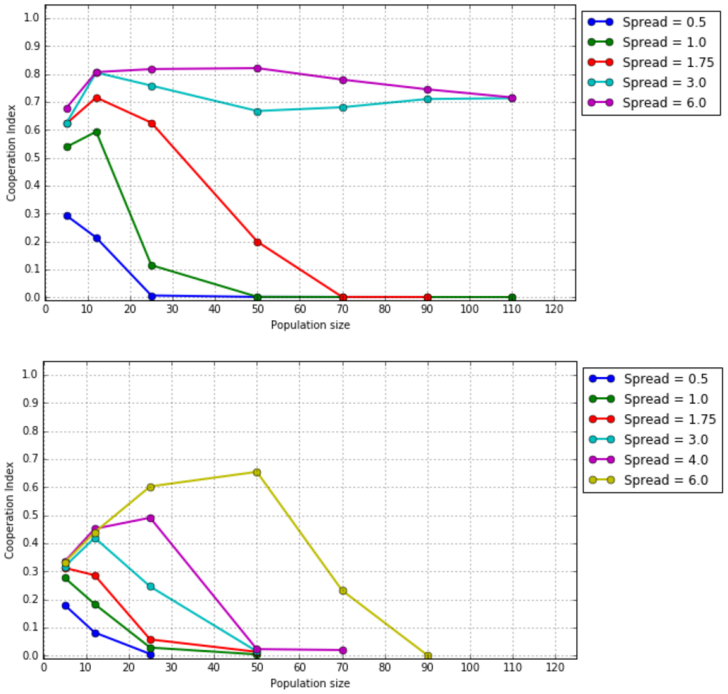
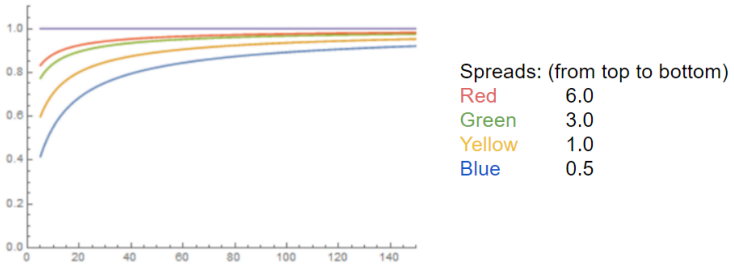


Figure 5: Results of Stern Judging (Top) and the Simple Standing (Bottom) under parameters $\epsilon = 0.08, \chi_{min} = 0.01, \alpha = 0.01, \mu = (10 \cdot Z)^{-1}$ under different information spread rates.

6.3 Discussion

The tests conducted involved investigating the effect of different spread rates over a range of population sizes. With the expectation being that a slower reputation spread rate results in information being unavailable to agents in the population when interacting with an agent whose reputation has recently changed. The time it takes for information to spread to the entire population increases logarithmically with the population size, however the probability of interacting with an agent whose reputation has recently changed decreases with the increase of the population size. From this we expect that for a large enough population size, this information delay will become insignificant and interactions should be unaffected by this additional error.

Investigating the probability that any given agent will **not** interact with an agent whose reputation has changed within the amount of time required for this information to spread to the entire population, we arrive at the following graph. The probability of not interacting with an agent within this time frame is shown for a series of population sizes.



We see that over time the probability decreases since the time taken to spread increases logarithmically and the rate of increase decreases. Since the chance of interacting with any given agent is uniformly $\frac{1}{Z}$ the probability of **not** interacting will an agent that has had their reputation changed within the time for their assessment error to go to χ_{min} converges to 1.

Looking at the results for the Stern Judging social norm, we see for low spread rates, the cooperation index starts low and then converge to zero very quickly. As the information spread rate increases, we see that the plot of cooperation indexes approaches that of the simulations ran without a variable private assessment error. Extremely low spread rates show poor rates of cooperation for all population sizes however spread rates of 1.0 and 1.75 still show relatively high rates of cooperation for small population sizes. As the spread increases, we see that for spread rates for 3.0 and 6.0 both reach the same cooperation index for a population size of 110. The spread rate of 3.0 defies the trend of the lower spread rates. While lower spread rates go towards zero very quickly, a spread of 3.0 decreases slightly at a population size of 50 yet continues to rise for 70, 90 and a marginal increase at 110.

For the social norm Simple Standing however, it is more evident that regardless of the spread rate, the cooperation indexes goes to zero. The larger the spread rate, the more it approaches the shape of the constant assessment error model of the simulation. A spread rate of 6.0 is most similar to that of the constant assessment error trend, yet begins to fall to zero after a population size of 50. Each plot shows a trend which follows the constant assessment error trend and quickly drops off before approaching zero as the population size increases. Similarly to the results of the social norm Stern Judging, lower spread rates consistently result in a lower cooperation index, however spread rates in the midrange (spread rates of 1.75-3.0) can provide relatively large cooperation indexes for small populations yet cannot maintain this for larger population sizes.

Regardless of the spread rate for both Stern Judging and Simple Standing, the difference in the respective cooperation rates holds true with regard to the results of the constant assessment error model. Stern Judging maintains a high cooperation index all population sizes, and in comparison to Simple Standing, outperforms, for all population sizes with respective spread rates.

6.4 Further work

There is work still required to investigate the effect of variable assessment error on larger population sizes. The expectations formed around the effect of variable assessment error on large populations have not been looked into. While we see that lower spread rates lead to zero cooperation rates for larger population sizes, for much larger population sizes simulations are required to verify the expectation that the cooperation rate will increase as the population size reaches much larger numbers.

References

[1] Martin A. Novak, Karl Sigmund, (2005) *Evolution of indirect reciprocity.* Nature.

[2] Fernando P. Santos, Francisco C. Santos, Jorge M. Pacheco, (2016) *Social Norms of Cooperation in Small-Scale Societies.* PLOS Computational Biology.

[3] Numpy Developers, (2016) *Numpy.* `http://www.numpy.org/`.

[4] Continuum Analytics, Inc., (2016) *Anaconda and Anaconda Accelerate.* `https://docs.continuum.io/`.

[5] Hisashi Ohtsuki, Yoh Iwasa, (2005) *The leading eight: Social norms that can maintain cooperation by indirect reciprocity.* Journal of Theoretical Biology.

[6] Olof Leimar, Peter Hammerstein, (2001) *Evolution of cooperation through indirect reciprocity.* Proceedings of the Royal Society B.

[7] GitHub Repository, (2016) *GameTheory1041.* GitHub. `https://github.com/konradcybulski/GameTheory1041`.