Working Title: Evolution of Behavior in the Repeated Nash Demand Game. A Computer Simulation.

Nguyen Linh Chi
University of Trento

Abstract

Bargaining is an important problem in economics literature. We simulate a population of agents adopting different strategies of the repeated Nash Demand Game and let it evolve. The result suggests that, when the game is a one-shot bargain, the 50-50 division is very stable. However, when the game is repeated (and/or the discount factor is sufficiently high), the 50-50 share is not that stable any more. Throughout the simulation, inefficient periods appear consistently. Upon closer inspection, these inefficient periods are due to aggressive strategies in the population. With this result, we would like to offer a way on *a priori* ground to explain why there is not always fairness in dividing a pie, and different societies may have different ways of sharing it.
Keywords Nash Demand Game, agent based simulation, evolutionary game theory.

1 The Stage Game

We consider a specific version of the bargaining game called the Nash Demand Game (NDG) in which two players simultaneously make a claim over a given pie of size 1. Each player has 3 strategies: to claim a High, Medium or Low fraction of the pie (which are equivalent to claiming $1-\pi$, $\frac{1}{2}$, or π, with $0 < \pi < \frac{1}{2}$). If their claims are compatible (i.e. the sum doesn't exceed 1), both get what they claim, otherwise, none gets anything. The payoff matrix of this NDG is in Table 1.

NDG	Low	Medium	High
Low	π, π	$\pi, \frac{1}{2}$	$\pi, 1-\pi$
Medium	$\frac{1}{2}, \pi$	$\frac{1}{2}, \frac{1}{2}$	0,0
High	$1-\pi, \pi$	0,0	0,0

Table 1: Nash Demand Game payoff matrix

An instant of the NDG, for example, is shown in Table 2, with the total pie is 10 and $\pi = 3$. Henceforth, to be convenient, we mostly refer to this instance as a focal example.

NDG	Low	Medium	High
Low	3,3	3,5	3,7
Medium	5,3	5,5	0,0
High	7,3	0,0	0,0

Table 2: Nash Demand Game payoff matrix

2 The Repeated Game

When the stage game is played for just one time, the number of strategies is three. However, when it is played more than once between two players, the number of strategies increases exponentially with the number of rounds. In each round, one can choose among three possible strategies, hence, with an example of ten rounds, the number of possible strategies is 3^{10}. The large strategy set makes it difficult to calculate the game analytically hence it motivates us to run computer simulation on the matter.

2.1 Strategy Representation

There is a vast literature on strategy representation and agent based simulation. Some very close references would be the works of Axelrod et al. [1, 2, 3, 4, 5] and Fogel et al. [22, 23, 24, 25]. In general, the strategy can be described in two overlapping methods: ruled based or state based. An example of ruled based strategy is a looking-up book in which one will look up what to do next based on the outcome of previously played rounds. In this way, the game can have a slightly different name called the iterated game. This method was initially used by Axelrod (1987) in his competition of the Prisoner's Dilemma game in which the strategy book prescribes action based on 3-previously played rounds. We can say that the strategy has memory 3 in that case. Then Fogel (1991, 1993) suggests to adopt the finite state machines in representing the strategies.

The method used here is finite state machine. A finite state machine is a machine with finite number of states. Each state contains two pieces of information: one about the move to make in that state, one about how to response to the opponent's move (i.e. what state to jump to). Besides, the strategies we consider here are deterministic strategies in the sense that they prescribe deterministic actions. An extension would be to cover the probabilistic strategies that choose actions randomly in each round.

As an example, a basic strategy in the repeated NDG simply prescribes the player to claim Low as long as the game lasts, independent of the opponent's move. This strategy can be represented as a machine with only one state, in that state, the move to make is to claim Low, and all the responding rules say the machine to stay in that same state (i.e. no matter what the opponents does). The graphical representation of the machine is shown in Figure 1a, in which the state is the circle, with the letter L inside the circle prescribing the move, the trajectories are the responding rules, with the letters on those trajectories being the corresponding moves of the opponent. So to say, in total this game has 3 unconditional strategies as such: the one that always claim Low, an other one always claims Medium, and an other that always claims High (Figure 1).

Figure 1: Unconditional strategies: Lows, Mediums and Highs

A more complex strategy is shown in Figure 2. It's an accommodating strategy: it starts out to play Medium, but then plays best response to whatever the opponent plays in the previous round. To make it clearer, the match between this Accommodator and the Highs strategy is shown in Table 3.

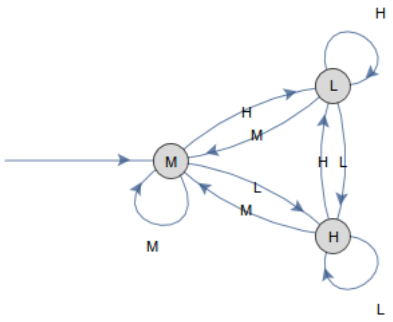

Figure 2: Accommodator

Round	Accommodator	Highs	A	H
1	M	H	0	0
2	L	H	3	7
3	L	H	3	7
..	..	..	..	..

Table 3: Accommodator vs. Highs

2.2 Payoff Calculation

Given that in a repeated game, each player gets a sequence of payoffs after each match, we shall transform this sequence in a final representative payoff for each player. One way is to take the mean of the sequence, hence each instance gets an equal weight over the final payoff. Another way is to use δ. δ can have two different interpretations. First, it can be the discount factor. Hence the payoff sequence is regarded as an income stream over time and the value of it would be the present value of the whole stream. In this way, δ can be regarded as a measurement of the player's patience. A δ very close to 1 says that the player treats payoff in the far future just as today. $\delta = 1$ is the case of taking the mean of the sequence above, it's as if the player is infinitely patient. A δ very close to 0 says that the player doesn't care much about future payoffs. A $\delta = 0$ is the case of one shot game (only payoff in the first round is considered). Second, δ can be the probability to continue the game after the first round is played for sure. In this way, the payoff sequence is summed up after the game ended. At $\delta = 0$, the game is one shot game. At $\delta = 1$, the game never ends. In between, everybody knows that the game will ends but in an uncertain manner (i.e. they don't know exactly whether there would be tomorrow or not).

Mathematically, as the number of rounds to be repeated grows very large (the game becomes infinite), the two methods converge to be equivalent. However, this difference would lead to two ways of implementing the simulation. Further details on this would be discussed in the next section of simulation implementation.

3 The Evolutionary Framework

The classical literature [6, 11, 12, 18, 27] surely provides substantial tools and equilibrium notions to study such games. However, it has received quite criticising contributions recently from the empirical literature of having strong assumptions on the (hyper) rationality of players. The evolutionary approach (with some close references in [7, 8, 9, 11, 13, 20, 21, 29, 31, 32]) relaxes such assumptions by exploiting the large number effect. It lets a population play the game or let the game be played over and over again by an individual. The player may not have the ability to do complex optimization but she *learns*. And the population approach shows that a simple motivation at individual level can make remarkable pattern emerge at macro level. In fact, in many cases, this approach leads to the hyper solutions already established in the classical literature.

The evolutionary approach originates in biology. It can be traced back to Darwin and the accumulated contribution before him. Maynard Smith [14, 15] formally proposes to apply the survival of the fittest mechanism to treat social interaction (i.e. put cultural norms as units under selection and mutation process). However, the generalised Darwinism (Dawkins 1983 [10]) has gained ground on many other critical fields also, such as technology.

4 Simulation process

4.1 Population and Agents

Starting from a world with many slots, we populate them with agents. The population will be kept fixed and each agent will adopt a strategy/machine to play the game. Theoretically the machine can have infinite number of states. However, in implementation, to accommodate for a consistent mutation mechanism, we would like to limit the number of states to be a fixed number. This would be discussed further in the mutation section. As a side note, the larger the fixed number of states, the better the result would look. We speculate partly that the complex machine has time to figure out rational playing hence they don't reach an optimal play soon at initial rounds. This can lengthen the learning period (even irrational) quite long and should be considered noise (i.e. of less interest).

4.2 Cycle

A typical cycle has 3 phases: matching phase, learning (or regeneration) phase, and mutation phase. In the matching phase (Figure 3), agents will be matched randomly in pair to play the NDG for multiple rounds. Their payoff sequence will be collected to calculate the corresponding fitness (i.e. the relative share of each final payoff in the total population payoff). The fitness vector of the whole population will be used in the regeneration phase (Figure 4). In the regeneration phase, a small number of agents will be allowed to *learn*. Technically, they randomize over the fitness vector to choose their new strategy. Because the fitness of each strategy is different, a better strategy will be more attractive and it'll be more likely to be chosen hence it'll become more popular at the next cycle. In this way, the strategy that performs better will grow at the expense of the poor doers. In the mutation phase (Figure 5), a small number of agents will experiment on their current strategy (mutating). This is to keep adding variety into the selection pool. Specifically, each machine is born with a fixed number of states, then in the mutation phase, one of these kinds of mutation will be carried out:

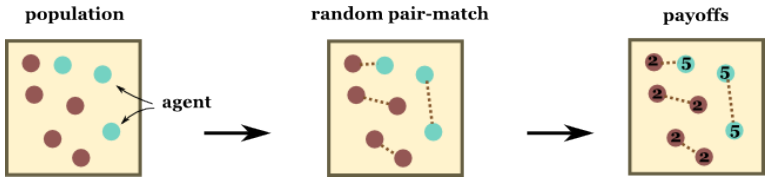

Figure 3: Matching phase

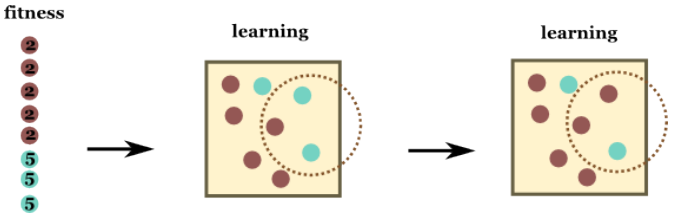

Figure 4: Learning phase

a state action is changed or a transition rule is changed. In this way, each mutation is a single unit of change. On the other hand, mutating by adding or deleting a state would involve tracking and changing all the related transition rules hence leading to a burst of mutations. The price of using fixed number of states (and let it connected at core initially) is that the machine can evolve to be complex over time but doesn't seem to de-evolve into a simpler one ever. So if we want a simple machine to be a representative, we ought to run with smaller fixed number of fixed states. The result doesn't qualitatively change however.

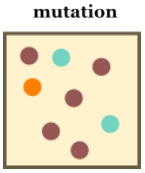

Figure 5: Mutation phase

4.3 Payoff Calculation

In implementation, there are two ways to calculate payoff after matching agents. First, in the discount method, we let the pair play the game for a large number of rounds then calculate the present value of the stream accordingly. The number of rounds, though finite, would be large enough so that the instances at the end are negligible after being discounted. In specific, the rounds are set at 400, at which, $\delta^{400} \to 0$. For example, at $\delta = 0.99$ we have $\delta^{400} \approx 0.02$, hence, even at $\pi = 0.5$ and the highest payoff possible in a round is 9.5, the present value of 9.5 is only 0.19. We assume that we can safely cut off the match after that. Second, in the continuation probability method, we let the pair play the first round for sure, then a Bernoulli draw is carried, at the probability δ, another round will be played, at the probability of $1-\delta$, the match terminates. The payoff instances will be summed up to be the final payoff of that match. Then we plot the population average payoff over cycles.

The continual probability method is less viable because it generates matches with very different length of play even with the same pair of machines. For example, a continual probability δ being anything below 0.7 will generate a distribution containing many value 1 which is to say that it generates more or less the one shot game. With higher value of δ, says, at 0.9, the distribution of rounds that it generates is very disperse. Hence, the final payoff (and the fitness) depends largely on the number of rounds that the machine is allowed to play. This is a factor that shouldn't be relevant in evaluating the merit of each strategy. Hence we focus on the discount method and report the result on the discount method. That to say, we do run with the continual probability method and it generates similar but highly fluctuated simulation.

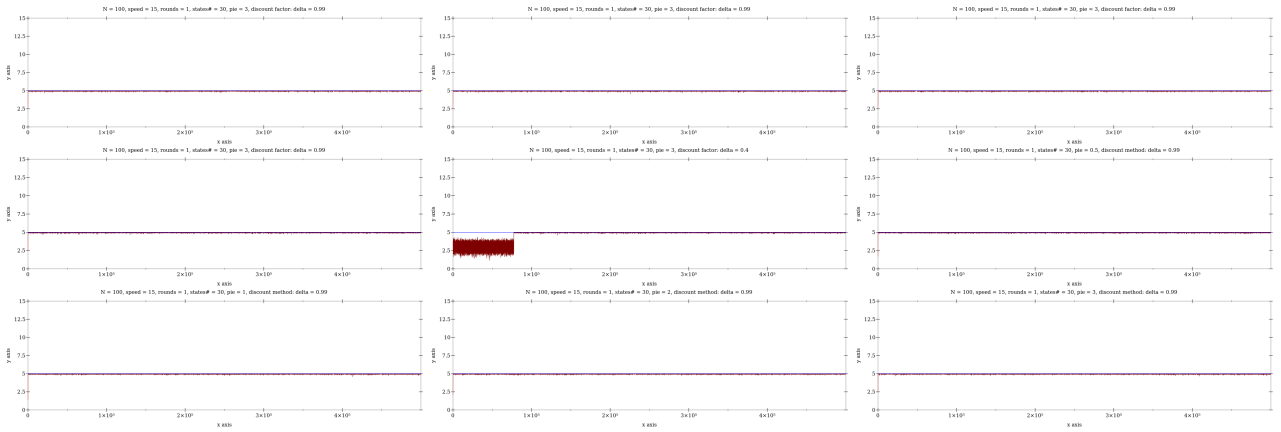

Figure 6: Runs of the one shot game

5 Result

5.1 One shot game

Across the simulation (Figure 6), the population average payoff is almost exclusively 5 which means that the whole population is dividing Medium all the time. The period of lower average in the middle graph is the mixed equilibrium in which the population plays High and Low. So far this equilibrium only appears initially as the simulation starts and it will always find a way into the fair equilibrium from that mixture. There is no sign of a reverse transition from the stable 50-50 division into this mixed equilibrium.

5.2 Repeated game

The simulation of the repeated game has a different pattern. Because the initial population is generated randomly and so is the mutation phase, there are runs that have better and worse results, we report here different runs to convey a broad picture of the result in general (Figure 7).

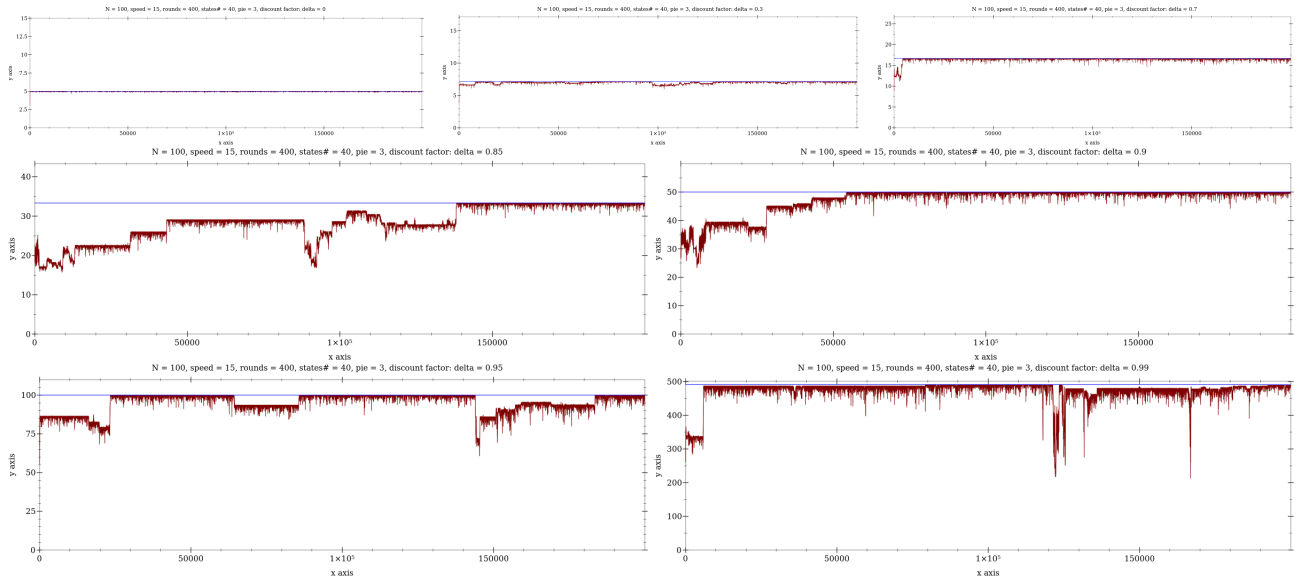

Figure 7: A typical run, $\delta = 0, 0.3, 0.7, 0.85, 0.9, 0.95, 0.99$

Some key notes the simulations suggest are:

- When δ is sufficiently small (smaller than 0.7), the 50-50 division is still highly stable, hence the population average is 5 and the fluctuations are due to random perturbations. The mixed equilibrium (High and Low) is rare and only happens when initially the population is created in its basin of attraction. After the population transits from the mixed equilibrium into the fair division, there is no reverse transition.

- When δ is sufficiently large (larger than 0.7), there are periods that the population average is significantly less than 5 (a lot or just a little). These can be called low periods (or inefficient) and they appear consistently across simulations.

- When δ is close to 1, the low periods in general appear more but really low periods appear less.

- These inefficient periods show different characteristics if we analyse the automata in them. A general pattern is that, before the inferior periods, there would be Accommodator. The inferior periods host Tough and Bully which are a neutral pair. However, Tough resists Fair but Bully doesn't hence Bully paves the way for Fair to invade back into the population. Fair are neutral toward Accommodator hence Accommodator in turn can make the Fair population be vulnerable to different kinds of intruders (which can be classified as *Tough*). Further details would be discussed in the next section.

- When π runs from 0.5 to 4.5 (increasing), the population still have low periods but these low periods become less low.

6 How to analyse population characteristic

Take the run in Figure 8 as an example. The blue line is a benchmark that represents the best scenario in which everybody plays Medium all the time. So whenever we see the population payoff "hover" below that benchmark, it's clear that the population is not dividing 50-50 anymore. Then we proceed to analyse by collecting the demographic data in cycles before, in and after a low period, but only counting the machine with at least 10% representation in the population. Machines that has number lower than that threshold are considered noises. Note that a machine can be quite complex so it's difficult to say about its characteristic based on its contingent plan alone. We match it against itself, Lows, Mediums, and Highs machine to see how it responds and make judgement on that.

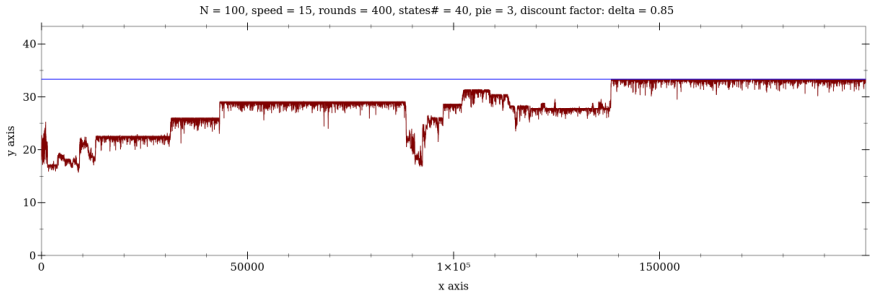

Figure 8: $\delta = 0.85$

To make it easier to draw conclusions on machine, we'd like to define some characteristics: Tough, Bully, Lows, Mediums (Fair), Highs, Accommodator. The basic Lows, Mediums, Highs and Accommodator are presented in the previous section. Here we'd show the basic Tough and Bully in Figure 9. Based on the payoff matrix among these machines in Table 4, we'd like to generalise these following things about their characteristics: A Tough machine is a machine that resists both Highs and Fair (i.e. it doesn't cooperate with Fair and doesn't accommodate Highs), but does well with its own kind. A machine does well among its own kind in the sense that it claims High for a few initial rounds but then retreat to claim Medium so both can earn something onwards. Hence the payoff sequence it generates can be: 0 0 0 5 5 5 ... Another less well behavior is that it mostly play Medium but occasionally goes back to play High so it is not a Fair machine but it is not a Highs machine either. The typical payoff sequence it generates can be 0 0 5 5 5 0 0 5 5 5 ... A Fair machine does completely well with its own kind (with the payoff sequence 5 5 5 ...) and a Highs machine doesn't do well at all with its own kind (with the payoff sequence 0 0 0 ...). A Bully machine resists Highs but it cooperates with Fair machine, hence it's dominated by the Fair machine. A Bully machine also does well among its own kind. Tough and Bully are a neutral pair. A Fair machine gets maximal payoff among its own kind and resists Highs. In order to plot these machine types, we associate them with different colors. Tough, Bully have bluish colors, Fair has green color, Accommodator has pink color, Highs has red color and Lows has light orange color. [1]

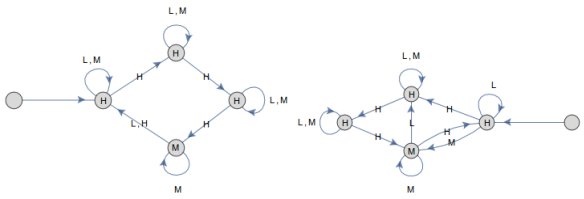

Figure 9: Basic Tough and Bully

[1] We'd like to note that, the word Fair, as well as Tough, Bully and others, despite being helpfully intuitive, are loaded words.

NDG	Tough	Bully	Fair	Accommodator	Highs	Lows
T	20.47 . 20.47	20.47 . 20.47	0 . 0	39.67 . 17.0	0 . 0	46.67 . 20.0
B	20.47 . 20.47	20.47 . 20.47	28.33 . 28.33	32.91 . 14.83	0 . 0	46.67 . 20.0
F	0 . 0	28.33 . 28.33	33.33 . 33.33	33.33 . 33.33	0 . 0	33.33 . 20.0
A	17.0 . 39.67	14.83 . 32.91	33.33 . 33.33	33.33 . 33.33	17.0 . 39.67	44.67 . 20.0
H	0 . 0	0 . 0	0 . 0	39.67 . 17.0	0 . 0	46.67 . 20.0
L	20.0 . 46.67	20.0 . 46.67	20.0 . 33.33	20.0 . 44.67	20.0 . 46.67	20.0 . 20.0

Table 4: Payoff matrix across classic automata, $\delta = 0.85$, $\pi = 3$

6.1 A detailed character classification

For a given automaton x, it'd be matched with itself to see how good it is with its own kind. The benchmark (100% kindness among itself) would be the case of a Fair pair. As in Table 4, the Fair machine gets 33.33 with its own kind so the *kindness* among Tough machines would be $\frac{20.47}{33.33} \approx 0.6$. We divide the kindness spectrum into 3 interval [0;0.6;0.995] as follows.

- The machine that gets in the region of having kindness index larger than 0.995 will be questioned further to be classified into Fair or Accommodator (because Accommodator plays exclusively fair among itself, just like Fair). We proceed by measuring the *accommodation* index. The accommodation index = $\frac{Highspayoff}{Highspotential}$. Highs payoff is the payoff Highs gets when it matches against the automaton. Highs potential is the payoff Highs gets when it matches with Lows hence Highs potential is the largest payoff possible in the game. The threshold is 0.5, if accommodation index is larger than 0.5, this automaton is showing accommodating characteristic and we classify it as Accommodator. As in Table 4, Highs payoff against Accommodator is 39.67 and Highs potential is 46.67. Hence the accommodation index of Accommodator is $\frac{39.67}{46.67} = 0.85$. In the other case, we classify it as Fair. (Tough, Bully, Fair have accommodation index to be zero. They show no tolerance toward Highs).

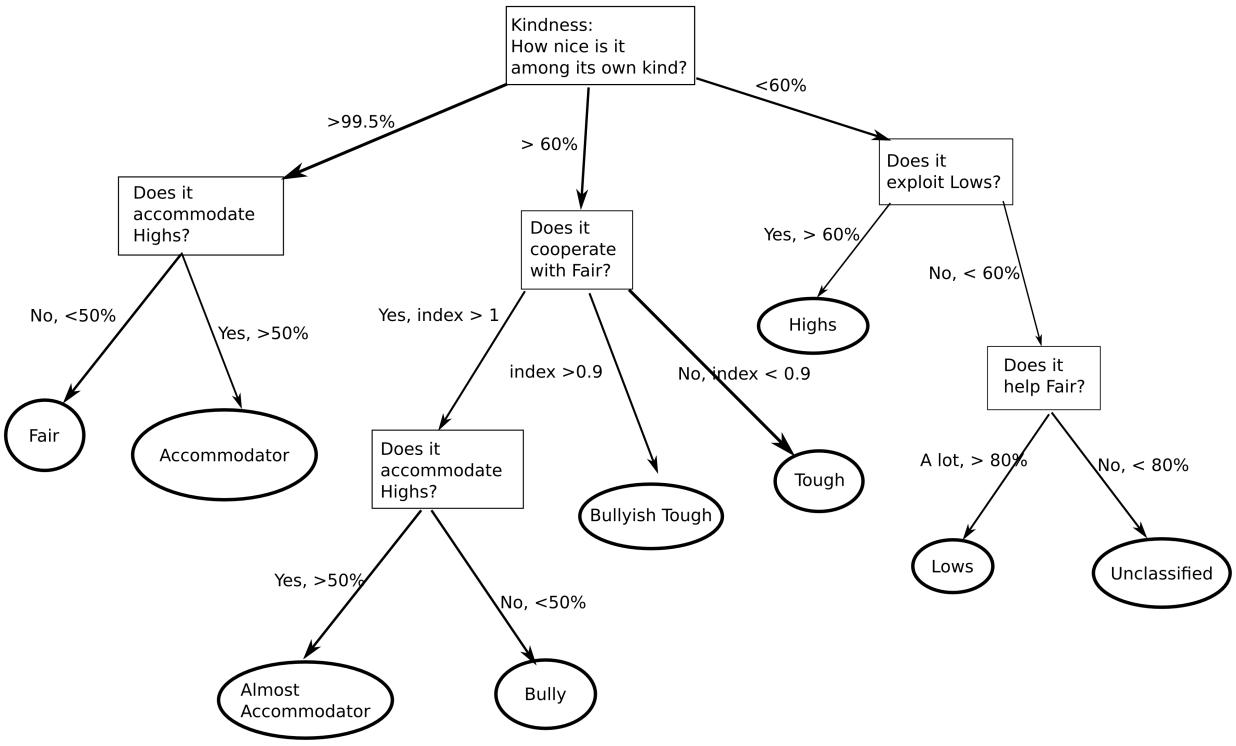

Figure 10: Classification test

- The machine that gets kindness index to be between (0.6, 0.995] will be in the middle ground. It can be Tough, Bully or Almost-Accommodator depends on its *cooperation* attitude toward Fair. The cooperation index = $\frac{Mediumspayoff}{Automaton'spayoffwithitself}$. We divide the cooperation spectrum into 3 intervals [0,0.9,1,∞]. This cooperation index is the ability to resist Fair so that the automaton can't be dominated by Fair. As in Table 4, the cooperation index of Tough is $\frac{0}{20.47} = 0$. Tough has zero tolerance toward Fair. However, the cooperation

index of Bully is $\frac{28.33}{20.47} > 1$. This means that Bully is dominated by Fair. In short, if the cooperation index is smaller than 1, it's a Tough machine. If the cooperation index is larger than 1, it's a Bully machine. However, we add a gray zone between Tough and Bully at the cooperation index being [0.9,1) because as the Fair tolerance increases, the machine (though technically is a Tough) grows very close to be a Bully so it can be dominated by Fair very soon. This kind of middle machine is called Bullyish-Tough. In the case of cooperation index being larger than 1, we'd like to be a bit more specific, because if Fair gets almost its maximal potential with this machine, then this machine has gone from being a Bully to an Accommodator. So we test its accommodation index, if the index is smaller than 0.5, this has the character of a Bully (Bully resists Highs completely). If the index is larger than 0.5, this is almost an accommodator. It's almost because this machine may not exploit Lows (though it cooperates with Fair and accommodates Highs).

- The machine that gets kindness index to be between [0, 0.6) can be Highs or Lows or some unidentified cases. In specific, if this automaton exploits Lows, it's classified as Highs. If it doesn't exploit Lows, it can be Lows or unidentified. If it cooperates with Fair, it's Lows. Lows has high cooperation index just like Bully but Lows is different in several aspects, for example, Lows among itself doesn't do very well, and Lows accommodates Highs absolutely.

The overall classification diagram is shown in Figure 10. Note that this is the test for one automaton and we apply it as follows. We collect the set of significant machines in the each cycle, then we test them individually. This would be fine if the machines in the population state are the same and they constitute a pure equilibrium. However, if the population hosts a mixed equilibrium, it'd be good to know the overall "character" of the mixture, whether it's favorable or hostile to Fair and other machines. Hence we carry out both a test of individuals and a test of the mixture as a whole. Usually the results are the same but it'd provide complementary information in the case the population hosts a mixed equilibrium.

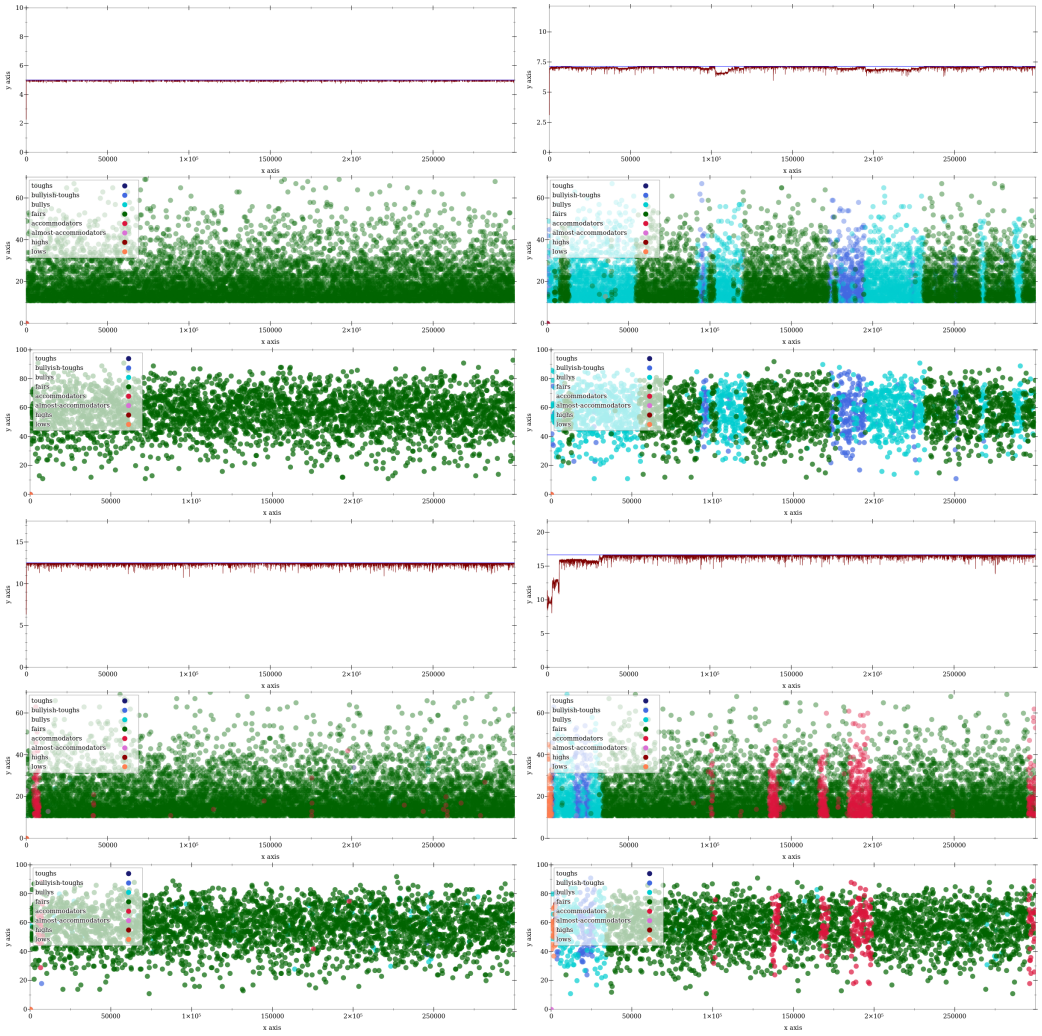

Figure 11: Population demographic over cycles, small δ. Each graph has 3 parts: the population mean, classifying popular machines in each states, and classifying the whole state. At small δ, most of the machines are fair machines (in green color) and the population mean is always 5.

7 Population state over cycles

7.1 Small δ: almost one shot game

We present four simulations of different values of small δ in Figure 11. Each simulation has three part: a graph plotting the average payoff across cycles, two other graphs plotting the characters in the population across cycles. One graph plots the character test for machines individually and one graph plots the character test of the population mixture as a whole. We can see that, at small δ, the population will spend most of its time in the fair division. Sometimes the simulation shows the mixed equilibrium of the one shot game (a mixture between High and Low) as in Figure 12. So far the mixed equilibrium as in Figure 12b only sustains initially. The transition from the mixed equilibrium to the fair equilibrium is very likely but not the other way around. That's why it hasn't appeared any simulation that has the reverse transition. Figure 12b has another thing to note. This is where the two tests show it difference. When we test the popular machines as individuals (separately), it shows that there are Highs, Lows and Accommodator in the population. However, if we test the whole population state as a character, it shows the characteristic of Highs and mostly Tough. This is to say that if we treat this mixture of Highs and Lows and Accommodator as a single character, it has the tendency to be Tough (i.e. it resists Fair and be good enough among itself).

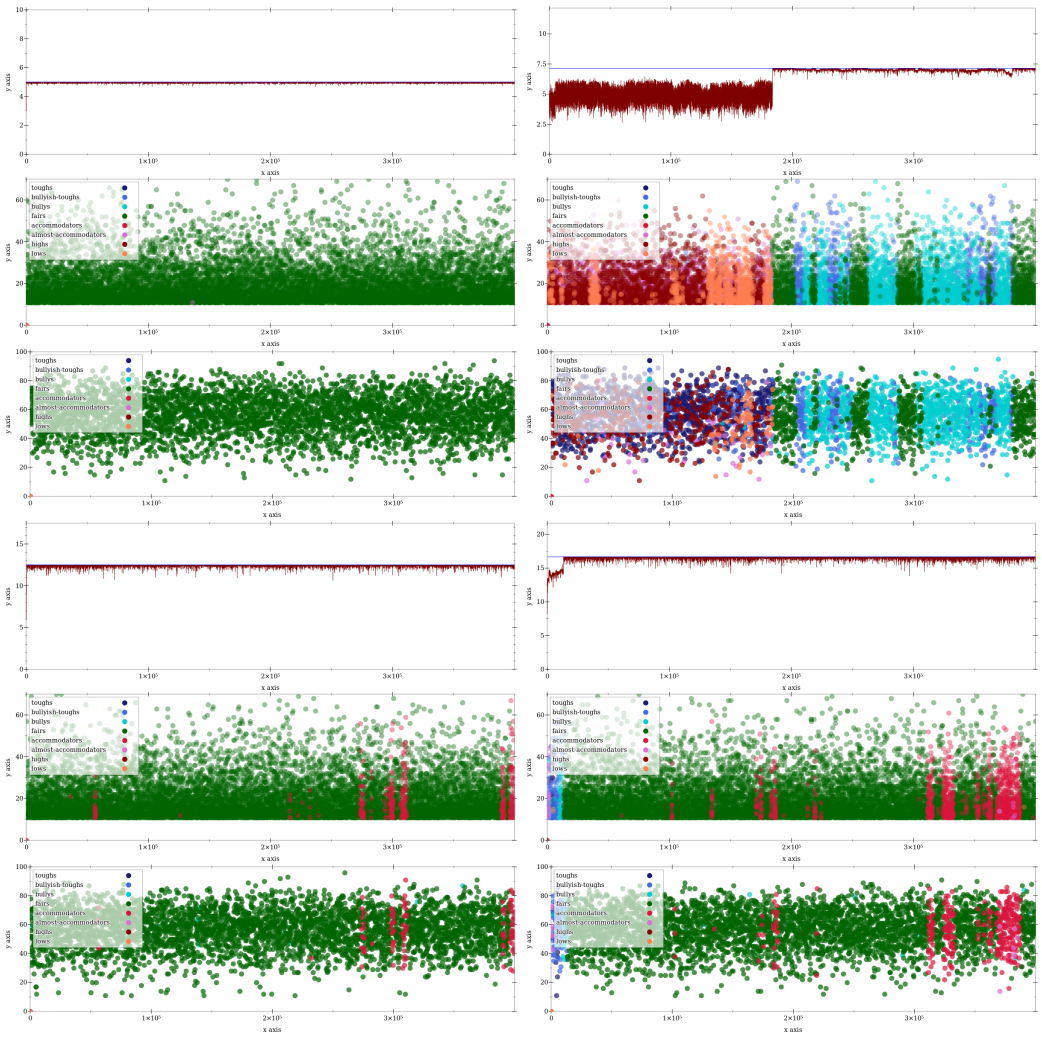

Figure 12: Population state over cycles, small δ, with a rare case in which the population mean is low. This is the mixed equilibrium of the one shot NDG (i.e. machines playing High and Low).

7.2 Large δ: Repeated game

As δ grows larger, the inefficient periods appear consistently across simulations. As in Figure 13, these inefficient periods are likely to host Tough and Bully. The Fair tolerance increases as we go from Tough to Bully, so Bully is likely to appear close to the rise of Fair equilibrium. However what causes the collapse of Fair equilibrium (the transition from Fair equilibrium to inferior ones) is more complicated. Sometimes it has the presence of

Accommodator (Figure 13b) right before the collapse but in Figure 13a, the existence of Accommodator in the population toward the end of simulation doesn't cause a collapse of the fair equilibrium. This speculation can be seen in Figure 14 also.

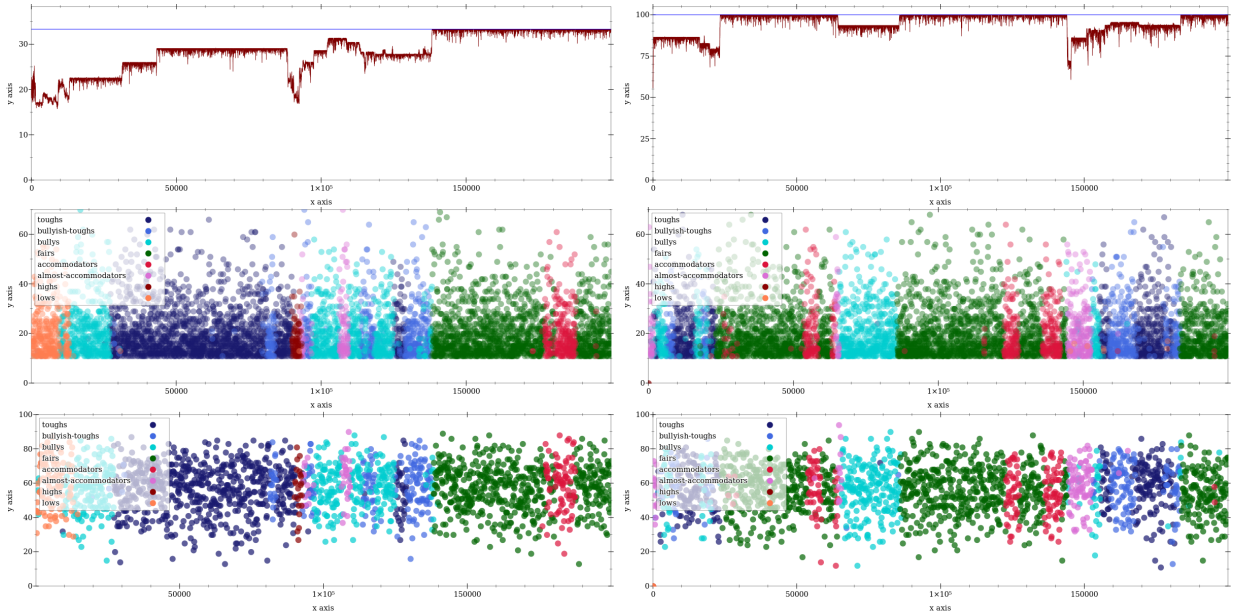

Figure 13: Population state over cycles, large δ. The population mean is not stable at 5 any more. Inefficient periods host Tough and Bully (in bluish colors). The Accomodator (in pink) appears before a collapse of the fair equilibrium.

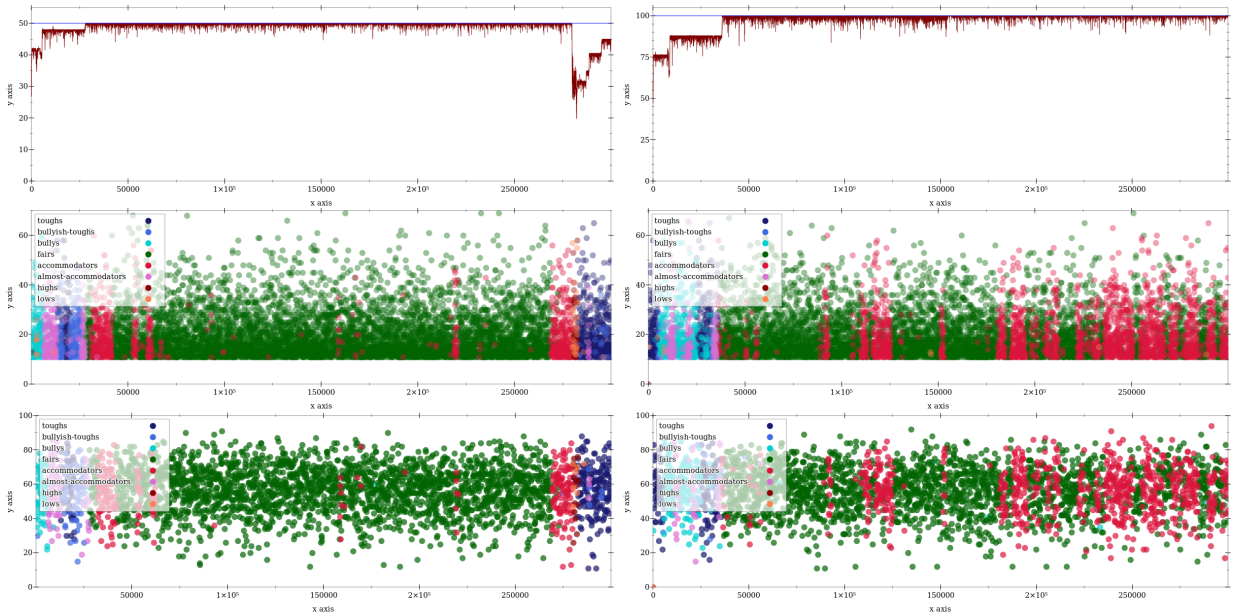

Figure 14: Population state over cycles, large δ. Before the collapse there's Accommodator but in these simulations, the presence of Accommodator (in pink) doesn't guarantee a collapse.

7.3 δ is very close to 1

When δ gets very close to 1, the population goes through a lot of ups and downs (Figure 15). The population seems to spend a lot of time in (slightly but not too much) inefficient periods in which the machines can be identified to be Tough or Bully. In contrast to the high δ simulation, the population when δ is close to 1 seems to have more inefficient periods but these low periods are not very low.

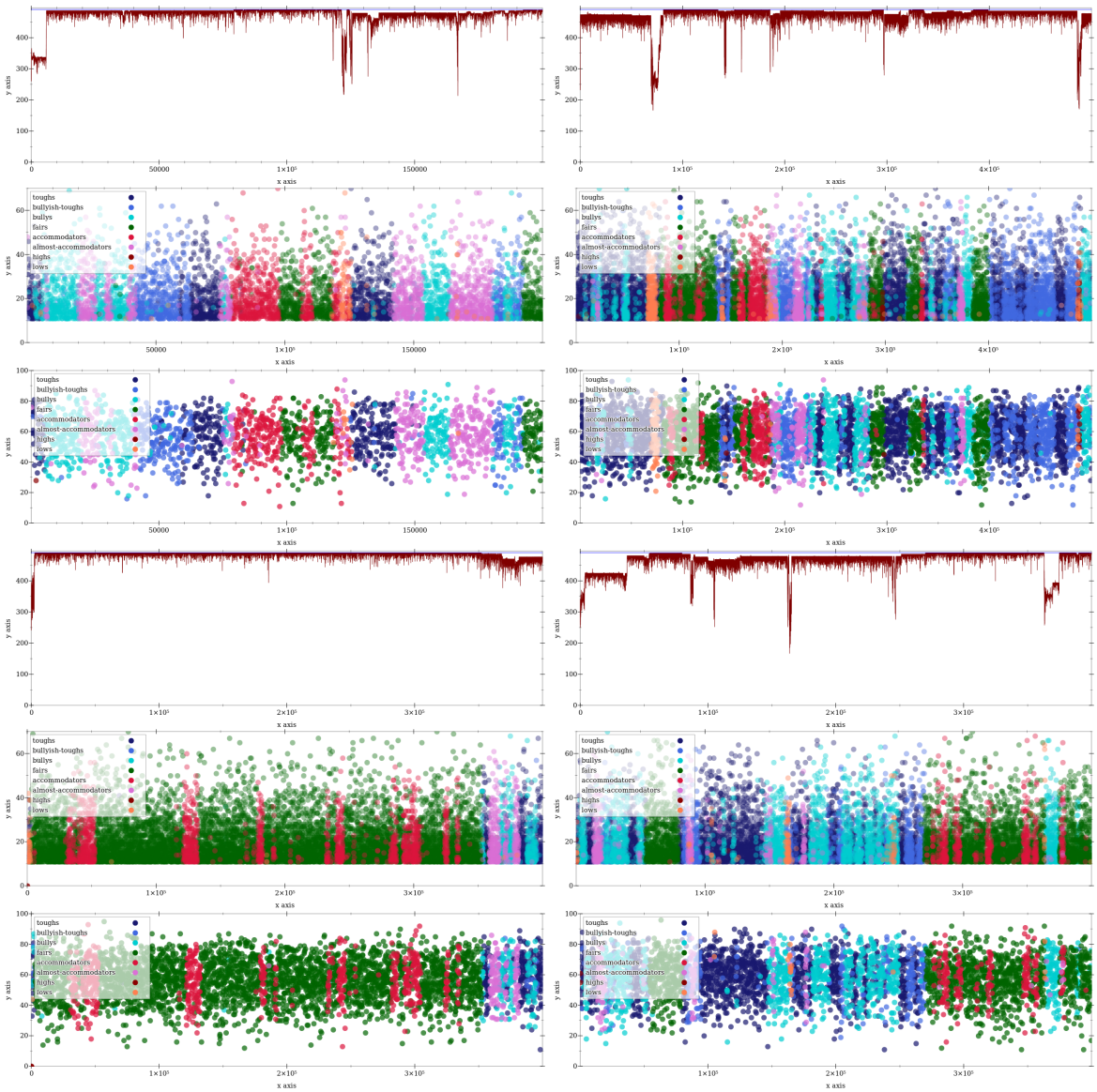

Figure 15: Population state over cycles, $\delta \to 1$. The inefficient periods appear more frequently, but are less inefficient. The inefficient periods host Tough and Bully (in bluish colors). The presence of Accommodator (in pink) doesn't guarantee a collapse of the fair equilibrium but before a collapse, they exist.

8 Concluding remarks

To sum up the main point of the article, we use Figure 16 and note different types of machines for different periods of the simulation. The thing that decides whether a strategy can form an inefficient equilibrium with itself or not lies in the fact that it cooperates with a Fair machine or not. If it doesn't (it plays High a lot, and both get 0), then it can sustain itself. It is a Tough one. If it cooperates (so the Fair machine can get a lot 5 after the match), it's very likely that it'd be dominated by the Fair machine. It is a Bully one. This reasoning can be applied to a mixed equilibrium also. If the mixture creates a favorable environment for the Fair machine, the Fair machine would dominate it. But if the mixture is somewhat "hostile" to the Fair machine, though both sides fare badly, the Fair machine cannot invade.

Vice versa, the *attitude* of the automaton toward the Fair machine can be used to classify it. A machine that cooperates with the Fair machine is a Bully. A machine that doesn't is a Tough machine. The Tough and Bully machine are neutral together. However, as Tough is hostile toward Fair, Bully makes the population becomes favorable for Fair. Once Fair is in place, it will start to develop variation that grows tolerance toward Highs. On the verge of the fair equilibrium collapse, there appears Accommodator. The Accommodator is neutral toward Fair, but it makes the population vulnerable toward many kinds of (pure or mixed) intruders. However, the presence of Accommodator doesn't guarantee a collapse. After a collapse, the Tough machine can regain and the evolution cycle goes on.

As mentioned in the abstract, with this simulation, we hope to offer an angle on the bargaining problem. The

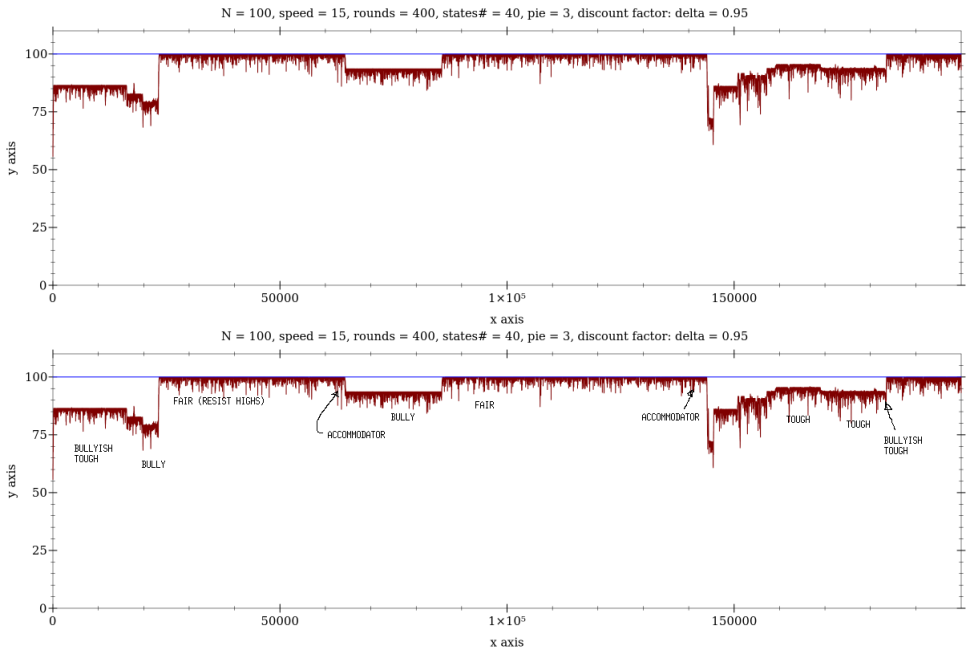

Figure 16: A sketching result of probing the population states, $\delta = 0.95, \pi = 3$

meaning of a repeated game lies in the assumption that players have enough data history to form expectation on how the opponent would behave in a particular situation. In that manner, the one shot game is a game between strangers (or anonymous matching). From the simulation, we suggest an interpretation of the result that says repeated interaction destabilises the fair equilibrium resulting in delayed negotiation and tough behavior. In other words, it's to suggested that *aggressive* behavior in barganing is supported on *a priori* ground though it can lead to the (undesirable or not) consequence of destroying inefficiency.

9 Acknowledgement

This work benefits gratefully from the guidance of professor Luciano Andreozzi. The simulation (that can be found here) is a customised version of the base code by Matthias Felleisen (here).

References

[1] Axelrod, R. (1980a). Effective choice in the prisoners dilemma. Journal of Conflict Resolution,24, pp. 3-25.

[2] Axelrod, R. (1980b) More effective choice in the prisoners dilemma. Journal of Conflict Resolution, 24, pp. 379-403.

[3] Axelrod, R. (1984). The Evolution of Cooperation. New York: Basic Books.

[4] Axelrod, R. (1987). Evolution of strategies in the iterated prisoners dilemma. In Genetic Algorithms and Simulated Annealing, L. Davis editor, Morgan Kaufman Publishers, Inc., pp. 32-41.

[5] Axelrod R.. *The evolution of cooperation: Revised edition*. New York: Basic Books, 2006.

[6] Binmore K., Samuelson L., Young P., Equilibrium Selection in bargaining models. *Games and Economic Behavior*. 45, 296-328, 2003.

[7] Binmore K., Piccione M., Samuelson L., Evolutionary Stability in Alternating-Offers Bargaining Games. *Journal of Economic Theory*. 80, 257-291, 1998.

[8] Binmore K., Samuelson L., Evolutionary Drift and Equilibrium Selection. *Review of Economic Studies*. 66, 363-393, 1999.

[9] Boyd R., Lorberbaum J. P.: No pure strategy is evolutionarily stable in the repeated Prisoner's Dilemma game. *Nature*, 327, 58-59 (1987)

[10] Dawkins, R., 1983. Universal Darwinism. In: Bendall, D.S. (Ed.), Evolution from Molecules to Man. Cambridge University Press, Cambridge, pp. 403425.

[11] Hilbe C., Nowak M. A., Sigmund K., Evolution of extortion in Iterated Prisoner's Dilemma games. *PNAS*. 110, 6913-6918, 2013.

[12] Kreps D. M.. *A course in microeconomic theory*. Princeton University Press, 1990.

[13] Lorberbaum J.: No strategy is evolutionarily stable in the repeated Prisoner's Dilemma. *J. theor. Biol.* 168, 117-130 (1994)

[14] Maynard Smith, J.. *Evolution and Theory of Games*. Cambridge University Press, 1982.

[15] Maynard Smith J. and Price G.R. 1973. The logic of animal conflict. Nature 246: 1518. Maynard Smith J. and Szathmary E. 1995. The Major Transitions in Evolution. Oxford University Press, Oxford

[16] Myerson, R. B., Nash Equilibrium and the History of Economics Theory. *Journal of Economic Literature*. 37, 1067-1082, 1999.

[17] Myerson, R. B.. *Game theory: Analysis of conflicts*. Havard University Press, 1991.

[18] Nash Jr. J. F., The Bargaining Problem.. *Econometrica*. 18, 155-162, 1950.

[19] Nash, Jr. J. F., Noncooperative games. *Annals Math.*. 54, 289-95, 1951.

[20] Nowak M., Sigmund K., A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature*. 364, 56-58, 1993.

[21] Press W. H., Dyson F. J., Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent. *PNAS*. 109, 10409-10413, 2012.

[22] Fogel, D. B. (1991) The evolution of intelligent decision making in gaming. Cybernetics and Systems, 22, pp. 223-236.

[23] Fogel, D. B. (1992). Evolving artificial intelligence. Doctoral Dissertation, University of California at San Diego.

[24] Fogel, D. B. (1993). Evolving behaviors in the iterated prisoners dilemma. Evolutionary Computation, 1 (1), pp 77-97.

[25] Fogel, L., A. Owens and M. Walsh (1966). Artificial intelligence through simulated evolution. New York: John Wiley & Sons.

[26] Roth, A. E, Bargaining Experiments. In: *Handbook of Experimental Economics*, ed. John Kagel and Alvin E. Roth. Princeton University Press, 253-348, 1995.

[27] Rubinstein A., Perfect equilibrium in a bargaining model. *Econometrica*. 50, 97-109, 1982.

[28] Schelling, T. C.. *The Strategy of Conflict*. Havard University Press, 1980.

[29] van Veelen M., Garcia J., Rand D. G., Nowak M. A., Direct reciprocity in structured populations. *Proceedings of the National Academy of Sciences*. 109, 9929-9934, 2012.

[30] Vega-Redondo, F.. *Economics and Theory of Games*. Cambridge University Press, 2003.

[31] Weibul, J. W.. *Evolutionary Game Theory*. MIT Press, 1995.

[32] Young P.. *Inidividual Strategy and Social Structure*. Princeton University Press, 1998.