CS662 Project

Sonja Linghui Shan

April 2022

This report looks at the application of finite automata in analyzing the Repeated Prisoner's Dilemma (RPD) game. Prisoner's Dilemma is a classic problem with ubiquitous real life applications. And it is intriguing to find out how uniquely suited finite automata are for RPD strategy representation. Automata help to express strategies in algebraic terms which allow for formal reasoning. The states can encode the relevant history of a game. And the well-defined structure allows for clear approaches to bound the complexity of a strategy or to simulate the mutation of strategies over time.

1 Prisoner's Dilemma

Prisoner's Dilemma (PD) is a classic example in game theory with wide applications in analyzing international trade, labour negotiations, doping in sports, etc. The problem consists of two players, A and B, each with a choice of two actions: to cooperate (C) or to defect (D). There are four types of payoffs a player can expect as described in the following matrix. If both players cooperate, then they both receive the reward R. If

		Player B: C	Player B: D
Player A	C	(R, R)	(S, T)
	D	(T, S)	(P, P)

one player defects and the other cooperates, then the defector receives the temptation payoff T while the cooperator receives the "sucker's" payoff S. If both players choose to defect, then they both receive the punishment P. And the game requires $T > R > P > S$.

This payoff structure leads to the dilemma: even though defection is the dominate strategy for each player since $T > R$ and $P > S$, mutual defection leads to worse payoffs for both players than cooperation since $R > P$. So the Nash equilibrium in PD is not Pareto efficient.

The Repeated Prisoner's Dilemma game (RPD) refers to playing PD sequentially for a finite or infinite number of times. And this version requires $2R > T + S$ to prevent alternating cooperation and defection from yielding a higher payoff than mutual cooperation.

2 Finite Automata

Here I describe a finite automaton with output or a Moore machine that is consistently used by researchers to represent strategies in RPD [7][4][6][8][5][2][9].

A Moore machine used in RPD for player i is $M_i = (Q_i, \Sigma, \Delta, \delta_i, \tau_i, q_i^0)$, where

- Q_i is the set of states M_i has,
- $\Sigma = \{C, D\}$ is the input alphabet which contains the potential actions of player j, $j \neq i$: cooperation and defection,
- $\Delta = \{C, D\}$ is the output alphabet which contains the potential actions of player i,
- $\delta_i : Q_i \times \Sigma \to Q_i$ is the transition function,
- $\tau_i : Q_i \to \Delta$ is the output function, and
- $q_i^0 \in Q_i$ is the starting state.

Consider two players i and j for a game of RPD, their strategies can be represented with M_i and M_j. In the first round, player i chooses $\tau(q_i^0)$ and player j chooses $\tau(q_j^0)$. Then in round t of the game, $t > 0$, player i transitions to the state $q_i^t = \delta(q_i^{t-1}, \tau(q_j^{t-1}))$ and takes action $\tau(q_i^t)$. Player j acts similarly. Essentially, the input to M_i in round t is the output of M_j in round $t-1$ and vice versa. This is consistent with how RPD strategies generally account for the history between the players.

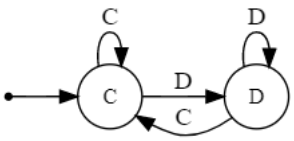
For an RPD, given a pair of Moore machines, we can deterministically generate a sequence of payoffs. Let μ be the payoff function in one round, then $\mu_i^t(\tau(q_i^t), \tau(q_j^t))$ is the payoff for player i in round t. For example, if $\tau(q_i^t) = D$ and $\tau(q_j^t) = C$ then the payoff is $\mu_i^t(D, C) = T$. And we can calculate the RPD payoff for player i by summing up the payoffs from each round, $\Sigma_{t=0}^{\infty} \mu_i^t$.

3 Axelrod's Tournament

In 1980, Robert Axelrod held a tournament of RPD with participants ranging from amateurs to experts in a variety of disciplines [1]. Each participant submitted a computer program describing a strategy. The programs faced off in a round-robin style with each pair playing an RPD of 200 rounds.

3.1 The Winning Strategy

The winning strategy was "Tit for Tat" (TFT) depicted below. The strategy cooperates in the first round



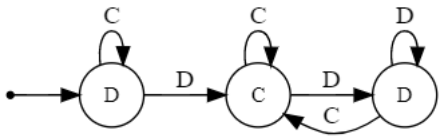
Tit for Tat

then mimics the opponent's move from the last round. This simple strategy winning is surprising as there were many more sophisticated strategies in participation.

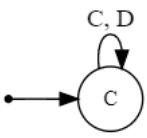
From analyzing the tournament, Axelrod concluded that the successful strategies share four properties:

- they cooperate first and as long as the other player does,
- punish the defection from the other player,
- go back to cooperation after responding to a defection, and
- show a clear pattern of behaviour.

Since the best payoff in RPD comes from mutual cooperation, these properties give clues on how cooperation can be elicited and maintained without friendship or altruism.



Mean Tit for Tat



All C

3.2 Ability to Not Be Nice

Expanding on the second property discussed above, Axelrod particularly addressed the importance of provocability. He writes that in the summer of 1915, soldiers refrained from shelling enemy ration wagons and water carts because "if you prevent your enemy from drawing his rations ... He will prevent you from drawing yours." Therefore the threat of revenge exists to defend the choice of cooperation.

Consider an RPD where both players employ the TFT strategy, neither Moore machine would ever arrive to the defecting state D. However the unused state serves as a credible threat fending off the opponent's temptation. If we take away D, the resulted All C strategy is extremely exploitable. For example, facing the Mean Tit for Tat strategy I made up, the payoff All C gets is much worse than TFT. So being cooperative without provocability will not let a strategy survive.

I also believe this point can help explain why Rubinstein's result, which states "cooperation cannot be the outcome of a solution of the infinite RPD"[7], seem to disagree with Axelrod's. See section 4.2.

3.3 Infinite Repetitions

Axelrod pointed out an important condition for cooperation to arise: the players have to interact with each other for the foreseeable future.

It is clear that in a one-shot PD game, the best action is to defect. Consider the case where the players interact for a known fixed number of times. In the last round, the best move is the same as in a one-shot game. Then via backward induction, we can conclude that a rational player would defect in every round.

Therefore, to incentivize cooperation, the future has to "loom large" over the players: either the game is infinitely repeated or the number of repetitions is unknown. Only then the potential of future encounters will make defection seem an unprofitable strategy.

This point is interestingly countered by Neyman [6] using bounded rationality. Neyman argues that if we bound the number of states allowed in a Moore machine to be at least one fewer than the number of rounds in the RPD, then cooperation can arise since neither player can maintain a count high enough to know when the last round is about to occur and defect promptly.

4 Bounded Rationality

In classical economics, there is the assumption of a perfectly rational agent who seeks to maximize utility. However this does not explain many real life observations. The idea "bounded rationality" then emerged to address the gap. Agents are still assumed to be rational, but the rational choices are made within the bounds of limited information, reasoning abilities, and time. Rubinstein wrote that although bounded rationality as an idea received high recognition, its impact on economic theory was limited due to a lack of formal representation models [7]. And that is where finite automata can help.

Rubinstein used the number of states in an automaton to capture the operational cost of carrying out a strategy, other costs like the cost of computing the optimal choices are not considered. Using this cost structure, Rubinstein found that cooperation cannot be part of the solution. This measure is fairly naive but pioneering as it sets an example for future researchers on formally analyzing elements of bounded rationality under the framework of game theory.

4.1 How to Measure Cost

The RPD in [7] are played by two Moore machines. And the goal for a player is to maximize average payoff and to minimize the number of states in the player's automaton. However the players care primarily about the average payoff, and "lexicographically only secondarily" [7] about minimizing the costs. The intention here is to not discourage complicated strategies that yield a good payoff.

Formally, the payoff and the cost for a player are arranged in a vector in that order. The vector represents the total utility considering both payoff and cost. And two vectors are compared using the lexicographic order on $\mathbb{R}^2$. So the preferred vector either has a better payoff or achieves the same payoff with fewer states.

Let $\pi_i(M_i, M_j)$ be the payoff for player i using M_i facing an opponent using M_j. Let $|M_i|$ be the number of states in M_i. Let $>_L$ be the lexicographic order on $\mathbb{R}^2$. If we have

$$(\pi_i(M_i, M_j), -|M_i|) >_L (\pi_i(M_i', M_j), -|M_i'|)$$

then player i prefers M_i to M_i' when facing M_j.

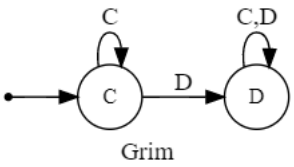
4.2 Semi-Perfect-Equilibrium

A Nash equilibrium in this game requires that

- neither player can achieve a higher payoff from their π function by unilaterally change their own automaton, and
- neither player can reduce the number of states in their automaton.

Extending from this, Rubinstein defined the semi-perfect-equilibrium which describes the solution to this game. Let G be an infinite RPD following Rubinstein's rules. Let G_t be the infinite RPD that starts at round t of G. If a pair of machines is a semi-perfect-equilibrium in G, then they form a Nash equilibrium in $G_t, t \geq 0$.

A direct consequence from this definition is that every state in a solution automaton has to be used infinitely often. If M_i contains a state that is not used after round t, then (M_i, M_j) would not be a Nash equilibrium in G_{t+1} because there exists a M_i' with at least one fewer state achieving the same payoff.



Therefore strategies like TFT and Grim are ruled out because they contain states that may never be triggered. Consider the case where two Grim strategies play each other, neither automaton would ever get to the defecting state. Rubinstein justified this by saying in real life, "human abilities degenerate and are readily discarded if they are not used regularly" [7]. So if a threat is not regularly carried out, it should be optimized away. However these "unused" states essentially serve as a credible threat needed for cooperation to flourish [1], without which cooperation is purely exploitable and cannot be the solution. And this is consistent with Rubinstein's result.

5 Evolutionarily Stable Strategies

Another approach to the RPD games is to identify the evolutionarily stable strategies. An evolutionarily stable strategy σ meets two criteria: the best response to σ is σ, and if there exists an alternate best response σ', then σ does better against σ' than σ' does against itself.

This definition is more stringent than the one Axelrod used which only requires the strategy to be in Nash equilibrium against itself. However this definition is perhaps too stringent as prior research showed that "no finite mixture of strategies is evolutionarily stable in the infinite RPD" [4].

Interestingly, Volij used this definition and proved that the always-defect "All D" strategy is the only evolutionarily stable strategy if we bound complexity and give preference lexicographically exactly as Rubinstein did [8].

Linster contributed by reporting a set of strategies that are not evolutionarily stable as defined above but "invasion-proof against permissible mutants" [4]. Consider a population made of strategies from this set, as mutants invade, mutants can only disturb the mix of the population, the set of strategies remains the same. And even though this set will not last forever, it may last for "a very long time"[4].

5.1 Mutation

Let us investigate the "permissible mutants" mentioned above. Linster stated two types of mutation.

One choice is to assume that an automaton can turn into any other automaton with uniform probability. But this is not very interesting.

To vary the probability in a sensible way, we need a notion of "closeness" so that the automata closer to each other is more likely to mutate into each other. And if we think of automata as physical machines then the hardware breakdowns give a natural example of mutation. For example, TFT can mutate into Grim if the wire handling the transition from the defecting state to the cooperating state breaks down. Note that this style of mutation does not allow for increased complexity unlike the crossover mutation used in section 6.2.

Additionally, Linster considered the possibility of reversed signals, that is the automaton would output cooperation when defection is intended and vice versa. Not much details are given on this here but this noise factor is thoroughly explored by Miller [5] in section 6.

5.2 Evolution

Linster considered all Moore machines with at most two states which comes to a total of 26 automata. He constructed

- a proportion vector $\mathbf{p}(t) = (p_1(t), p_2(t), ..., p_{26}(t))^T$ with each $p_i(t)$ being the proportion of strategy i in the population in round t,
- a payoff matrix $\mathbf{B}$, and
- a mutation matrix $\mathbf{M}$.

The game starts with $\mathbf{p}(0)$ being a random point in the unit simplex in $\mathbb{R}^{26}$. Then in each round, we get $\mathbf{p}(t+1)$ from $\mathbf{p}(t)$ in two steps.

1. We get $\mathbf{p}(t+1)'$ from several matrix vector multiplications of $\mathbf{B}$ and $\mathbf{p}(t)$. For every strategy i, the change $p_i(t+1)' - p_i(t)$ depends two factors. Whether the strategy generates a good payoff decides the direction of the change. And the magnitude of the change depends on $p_i(t)$. This is to show that even if a strategy is good, it has to be known for it to prosper.
2. Then we get $\mathbf{p}(t+1)$ from $\mathbf{p}(t+1)'$ via a matrix vector multiplication of $\mathbf{M}$ and $\mathbf{p}(t+1)'$. This is to simulate mutant invasions.

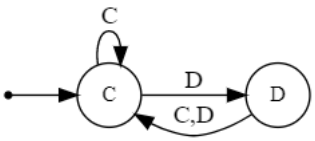
Notice that the first step above echos Axelrod's point that cooperation can arise from a small cluster of cooperative yet discriminating strategies interacting with each other [1]. These strategies are characterized as cooperating on the first move, then responding differently based on the opponent's reaction to the initial cooperation.

Additionally, in some simulations, Linster tried to capture the cost of complexity by giving adding a premium to the payoffs of single-state machines.

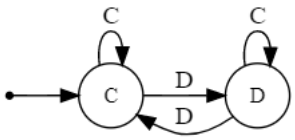
5.3 The Success of Being Grim

Linster ran five types of simulations.

1. Evolution without mutation.
2. Evolution with mutations of uniform probability.
3. Evolution with mutations of varied probability. The probability is varied base on the "closeness" concept as discussed above.
4. Evolution with mutants who enter as a group. This is inspired by the idea that animals separated from their own species can evolve differently. So Linster allowed one percent of the population to become mutants and then invade the population. Once the effect of the mutant group invasion settles down, Linster let in another group of mutants. This is analyzed for both mutations of uniform and varied probability.
5. Evolution with costly states. This is to see the impact of adding a premium to the payoffs of single-state machines: the All C strategy and its opposite the All D strategy.



Forgiver



Win-Stay Lose-Shift

The first four simulations all ended up with the same set of strategies: Grim, TFT, All C, Forgiver and Win-Stay Lose-Shift (WSLS). Although each simulation resulted in a different combination of strategies from this "invasion-proof" set, Grim was consistently the majority in every mix.

Note that all strategies in the set are "nice" [1] in that they never defect first. Grim is the least forgiving out of the set yet the most successful. This appears to contradict Axelrod who characterized forgiveness as a key property of a winning strategy. An explanation is: Linster's design severely restricts the complexity of a strategy by allowing at most two states and, to reveal the deficiency of being unforgiving, more complexity is needed. A strategy similar to Grim entered into Axelrod's tournament but did not enjoy much success [4] [1]. This shows that the success of Grim is very dependent on Linster's complexity restriction.

These simulations also reveals a weakness of TFT: Grim is better than TFT at taking advantage of irrational play since TFT largely copies the opponent.

The other surprise is the survival of the defenseless All C strategy. All C survived because it does well against other "nice" strategies and Grim is always around to keep the percentage of exploitative strategies low. Grim essentially acts as an enforcer and prevent the environment from turning too hostile for All C.

The last simulation revealed that when the cost of maintaining two states is too high, All D prevails. Otherwise, the populations cycles between All D and a cooperative mix led by Grim. This is an intriguing effect. A common cognitive bias is that our brain tends to swap a easier question for the right question [3]. When the cost is high, asking how to balance defection and cooperation in an rational way becomes too difficult. And a much easier question is to ask to defect or not to defect. Then All D becomes the winning choice.

6 Evolution with Noise

Miller studied the evolution of RPD strategies under perfect and imperfect reporting [5]. Three noise levels were used in the experiment: no noise, 1%, and 5%. That is, 1% or 5% of the time, an opponent's move is reported to be the opposite of the actual move. We will look into the design of Miller's experiment and then the results.

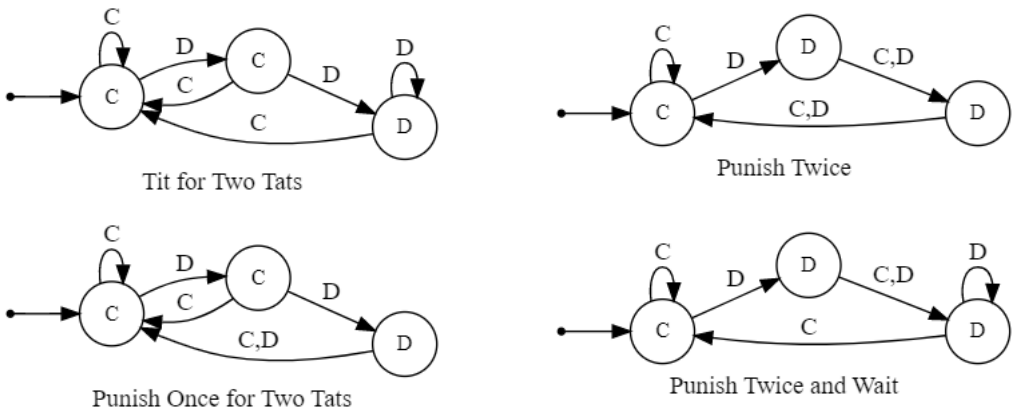
6.1 Binary String Representation

We have seen the English to Moore machine abstraction in the sections above. Miller went one step further and abstracted Moore machines into binary strings.

Each machine is represented by a string of 148 bits. The first four bits (in black in the example) indicate the starting state. The next 144 bits are divided into sixteen 9-bit sections with each section representing a state. Within a 9-bit section, the first bit (in orange) gives the output of the state with 0 being cooperation, the next 4 bits (in blue) give the transition destination if the next input (the opponent's move) is cooperation, and the last 4 bits (in purple) give the transition destination if the opponent is observed to defect. For example, TFT is 0000 000000001 100000001, omitting trailing zeroes.

It is clear that this representation is limited to automata with sixteen states or less. However 2^{148} unique structures offer Miller a sizeable population despite that not all string can give a meaningful strategy. And as we will see below, this representation plays a key role in the "reproduction" of strategies.

6.2 Crossover



New strategies in evolution are generated via crossover. Consider two parents P_1 and P_2 and a crossover point c. Since each strategy has a string representation, we can write $P_1 = uv$ and $P_2 = xy$, $u, v, x, y \in \{0, 1\}^*$, and $|u| = |x| = c$. Then we can create two new strategies $S_1 = uy$ and $S_1 = xv$ via crossover. Therefore a new strategy exemplifies traits of both parents. An example is if we have $c = 18$ and cross

- Tit for Two Tats: 0000 000000001 000000010 100000010 and
- Punish Twice: 0000 000000001 100100010 100000000,

we get

- Punish Once for Two Tats: 0000 000000001 000000010 100000000 and
- Punish Twice and Wait: 0000 000000001 100100010 100000010.

6.3 The Evolutionary Process

Miller derived the evolutionary process from a class of genetic algorithms originally developed by John Henry Holland in the 1970s for optimization problems.

The design is to start with 30 randomly generated strategies. Every strategy then plays 30 RPD games against the other 29 strategies and a clone of itself. Each RPD game has 150 rounds. And the final score for a strategy is an accumulation of its payoffs from every game. Then the next generation is produced as follows.

- The top 20 highest scoring strategies are copied.
- The 10 new strategies are generated via crossover. The parents are picked from the existing strategies. The likelihood of a strategy becoming a parent is positively correlated to its performance. The c value is randomly generated.
- The 10 new strategies are probabilistically mutated. Mutation involves flipping a random bit in the strategy string.

And the experiment is designed to go through 50 generations.

This process is a form of adaptive learning as it keeps the good performers and replaces the bad ones with new strategies that incorporate traits of existing high performers.

6.4 Misinformation Breeds Defection

We discuss some results of Miller's experiment.

- The average PD game payoff is about 2.3 across the randomly formed first generation. The PD payoff structure here is $T = 5$, $R = 3$, $P = 1$, and $S = 0$. So if everyone cooperates the average payoff would be 3. Then, over the next 49 generations, the average payoff first dropped then recovered. And this is where the noise factor makes an impact. With 5% noise, the drop is more pronounced , the recovery is slower, and the final average payoff is about 1.9 which is lower than the initial value. Whereas with no noise, the dip is less harsh and the final average payoff is about 2.8 which improves on the initial value.

  Since more cooperation leads to a higher average payoff, this implies that cooperation is less likely to occur in a noisy environment. This is consistent with Axelrod's point: for a strategy to elicit cooperation, it should exhibit a clear pattern of behaviour [1]. Defection appears as the safe bet against an unpredictable opponent.

- The average number of states is about 12 for the first generation. Then, over the next 49 generations, we observe this value drop then eventually leveling off. However, with 5% noise, the drop is much more significant and goes on for longer. The final average number of states is about 11 with no noise or 1% noise and 9 with 5% noise.

  Another side of this result is that the average proportion of terminal states drastically increases with 5% noise. Terminal states are the states whose transitions loop back into itself. The defecting state in Grim is an example.

  These results suggest a preference for simpler strategies facing elevated uncertainties. I think this is an interesting analog for a behavioural bias. Prospect theory points out that we are more sensitive to perceived loss than perceived gain which can lead to a bias towards the status quo [3]. With a higher level of noise, the perceived gain from adding more states or more transitions becomes increasingly unclear, then having fewer states and more terminal sates is preferred.

7 Evolution with Bounded Rationality

Ho presented an interesting study on the emergence of cooperation in an initially hostile population [2].

7.1 Experiment Design

The study shares Miller's experiment design in the use of binary string representation and reproduction by crossover and mutation. However, Ho parameterized the "toughness" of the environment and his did not use cloning in the experiment's reproduction design.

Like Rubinstein, Ho bounded the complexity of strategies by restricting the number of states in an automaton. Ho's approach is more lenient in that charging a cost is less punishing than requiring every state to be used infinitely often. Notably, a quadratic cost function was used. So each additional state incurred a higher fee, this is to reflect the increasing marginal costs of memory.

Additionally, Ho imposed a cost on switching states using a linear and a quadratic cost function. In the linear case, every switch is equally expensive. In the quadratic case, the strategies with a high frequency of state-switches were penalized more than proportionally.

7.2 A Lack of History Cuts Cooperation

We discuss some results from Ho's experiment.

- Ho found "the evolution of cooperation is not sensitive to the hostility of the initial environment" [2]. Individual strategies mutate, and once there exists a small cluster of cooperators, they can outperform the defectors and drive the defectors to extinction. Axelrod made a very similar observation in his research [1].

- The fittest strategies that survived in Ho's experiment are more complex than TFT or trigger strategies like Grim. The survivors tend to reciprocate cooperation less readily compared to TFT, but are more forgiving than Grim.

  This resonates with one of Miller's results. Miller found that, as strategies evolve, they tend to develop higher defection reciprocity and lower cooperation reciprocity. And the noise in Miller's experiment exaggerated this phenomenon.

- When a cost is imposed based on the number of states in an automaton, it can destroy the emergence of cooperative behaviors. The cost imposed incentivizes machines with fewer states. Having fewer states essentially shrinks the amount of history a strategy considers. Taken to the extreme, this turns RPD into PD. Consider playing RPD against an opponent with no memory. Then the game is played as a sequence of independent PD games. Facing such an opponent, there is no risk of retaliation and no incentive to cooperate.

  We can use this result as a different perspective on one of Miller's results. Miller found a preference for simpler and more unforgiving strategies in populations under a high level of noise [5]. Essentially both high noise and having fewer states contribute to an inability of properly understanding the opponent's pattern of behaviour which breeds defection.

- Surprisingly, charging a quadratic cost based on the frequency of switching states can aid cooperation if the fee is at a suitable level. Ho found the survivors in this case tend to have fewer terminal states and respond more readily to a change in the opponent's move. This signals forgiveness and flexibility in strategies are important for a cooperative population. This is a point also noted by Axelrod and we examine it more in the next section.

8 The Forgiver Triumphs

8.1 Alternating Prisoner's Dilemma

For this section we consider a modification to the RPD game. The Alternating Prisoner's Dilemma (APD) is when we allow players to choose their moves in turns. This change paired with noise from misreporting the opponent's move can lead to a change in what makes a strategy effective. For example, when two TFT strategies play each other in RPD, a mistake leads players to cooperate and defect in turns. But the same mistake would lead to mutual defection in APD as shown below.

TFT C C D* C D C ...
TFT C C C D C D ...

TFT plays itself in RPD

TFT C C D* D ...
TFT C C D D ...

TFT plays itself in APD

8.2 Experiment Design

Zagorsky, Reiter, Chatterjee, and Nowak studied APD strategies using methods similar to what Linster used to study RPD strategies [9] [4]. They both

- considered only the 26 strategies that can be represented by Moore machines with at most two states,
- started with a population that is a random mix of these 26 automata, and
- observed how the representation of each strategy change in the population with and without mutation.

However the authors introduced parameters to control the level of noise and the value of payoff. These factors were not considered by Linster.

8.3 Forgiveness Promotes Cooperation

The result points to Forgiver (shown in section 5.3) as the best APD strategy in the presence of noise. In all investigated scenarios, each with a different set of noise and payoff parameter values, Forgiver makes up at least half of the final population. The authors attributed its success to the following properties.

- A good strategy should cooperate as much as possible without being excessively exploited. Strategies like Forgiver, TFT, WSLS, Grim all have a design element to stay cooperative as long as the cooperation is reciprocated and to move to the defecting state if the opponent defects. The authors named this the "conditional cooperation element" and cited it as a key feature in successful strategies.
- What further differentiated Forgiver is its ability to recover from mistakes caused by noise. Once in the defecting state, in order to return to the cooperating state, TFT requires the opponent to cooperate; WSLS requires the opponent to defect; whereas Forgiver moves back to the cooperating state regardless of what the opponent does. To curb the impact of noise, this generous move is crucial. Grim enjoyed a lot of success in Linster's experiment, but the unforgiving nature of Grim makes it very vulnerable to noise; once it moves to defection, there is no going back.
- Forgiver does have some weaknesses. It can be exploited when the payoff parameters are tuned so that cooperation is not valuable enough. And if the population is very hostile, Forgiver cannot be an enforcer like how Grim was in Linster's experiment.

9 A Thought

It is worth noting the four properties raised by Axelrod (see section 3.1) are extremely insightful. The research that came after invariably tests and affirms one or more of these principles. For example, provocability can be connected to Rubinstein's work, forgiveness is related to aspects of experiments conducted by Zagorsky et al., Linster, Miller, and Ho, and the importance of signaling clearly is affirmed by the results of Miller and Ho.

References

[1] Robert M. Axelrod. *The evolution of cooperation.* eng. New York: Basic Books, 1984. ISBN: 0465021220.

[2] Teck-Hua Ho. "Finite automata play repeated prisoner's dilemma with information processing costs". In: *Journal of economic dynamics and control* 20.1-3 (1996), pp. 173–207.

[3] Daniel Kahneman. *Thinking, fast and slow*. New York: Farrar, Straus and Giroux, 2011. ISBN: 978-0374275631.

[4] Bruce G. Linster. "Evolutionary Stability in the Infinitely Repeated Prisoners' Dilemma Played by Two-State Moore Machines". In: *Southern Economic Journal* 58.4 (1992), pp. 880–903. ISSN: 00384038. URL: http://www.jstor.org/stable/1060227.

[5] John H. Miller. "The coevolution of automata in the repeated Prisoner's Dilemma". In: *Journal of Economic Behavior and Organization* 29.1 (1996), pp. 87–112. ISSN: 0167-2681. DOI: https://doi.org/10.1016/0167-2681(95)00052-6. URL: https://www.sciencedirect.com/science/article/pii/0167268195000526.

[6] Abraham Neyman. "Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma". In: *Economics letters* 19.3 (1985), pp. 227–229.

[7] Ariel Rubinstein. "Finite automata play the repeated prisoner's dilemma". In: *Journal of Economic Theory* 39.1 (1986), pp. 83–96. ISSN: 0022-0531. DOI: https://doi.org/10.1016/0022-0531(86)90021-9. URL: https://www.sciencedirect.com/science/article/pii/0022053186900219.

[8] Oscar Volij. "In Defense of DEFECT". In: *Games and Economic Behavior* 39.2 (2002), pp. 309–321. ISSN: 0899-8256. DOI: https://doi.org/10.1006/game.2001.0893. URL: https://www.sciencedirect.com/science/article/pii/S0899825601908930.

[9] Benjamin M. Zagorsky et al. "Forgiver Triumphs in Alternating Prisoner's Dilemma". In: *PLoS ONE* 8.12 (Dec. 2013). Ed. by Attila Csiká sz-Nagy, e80814. DOI: 10.1371/journal.pone.0080814. URL: https://doi.org/10.1371%2Fjournal.pone.0080814.