Iterated Prisoner's Dilemma, Trust Strategy and FSM's as a Mind Model for Searching Optimal Strategy

by Aleksej Tomazic, Chenghao Liu, Mingwei Sun and Zuoru Jin
University of Auckland - Faculty of Computer Science

Abstract:

In our research paper we touched and presented Iterated Prisoner's Dilemma as proposed by Axelrod in a number of his works. We developed our own strategy to compete in the tournament setting and also included Finite State Machines (FSMs) of various sizes, which we considered as a viable mind model for discovering optimal strategies. FSM Strategies were obtained through simulated evolution. In this work, we present our methodology and our observations.

1. Implementation

In this study, we leverage the programming language Python to implement a novel strategy in game theory. We primarily make use of Axelrod[1], a Python library that boasts a rich array of strategies, which also facilitates the development of new ones. In addition, we utilise Dojo[2], another Python library that houses reinforcement learning training code specifically designed for Finite State Machines (FSMs) (Harper et al., 2017)[3]. Our experiment entails implementing a new strategy, selecting suitable strategies, including FSMs, and employing a Genetic Algorithm (Goldberg, 1989)[4] to discern the optimal player across 20 generations. The chosen strategies then engage in a series of matches, each set containing 1000 iterations. This combination of strategic implementation, selection, and optimization through genetic algorithms, along with extensive gameplay, underpins our approach to exploring the dynamics of game theory.

In our experiment, we aim to evaluate the performance of the FSMs strategy - an approach that mirrors human evaluation of other strategies with the ultimate goal of winning. Our expectation is to obtain comprehensive insights from three specific metrics: Wins, Scores, and the Payoff matrix. The "Wins" metric provides a count of matches won by each player, directly reflecting their performance in the tournament. This information will be represented visually in a graph, enabling us to easily compare the competitive prowess of each strategy. Next, we consider "Scores," which denote the cumulative points garnered by each player. A graph displaying these scores will give us a clearer picture of the overall performance of each strategy beyond just the number of matches won. Finally,

[1] Axelrod-Python/Axelrod at 97db1f7c15d5a66807215278bde85a3cc89a2c01 (github.com)

[2] Axelrod-Python/axelrod-dojo: Trains machine learning strategies for the IPD with evolutionary and particle swarm algorithms, including neural networks and finite state machines (github.com)

we examine the "Payoff matrix" This matrix presents the payoffs of each player against every other player, offering a more nuanced understanding of the interactions between different strategies.

To explore the dynamics of FSMs in our game theory experiment, we need to define the number of states, choose the opponents, and set certain parameters. Hence, we created a function that accepts the number of state nodes and opponents as inputs. The opponents selected for this study are Second by Borufsen, Tit For Tat, Second by Champion, Grudger, Defector, Cooperate (Axelrod, 2006, pp. 8)[1], and Trust, which is our newly implemented strategy. Given the variability in the FSM's state, we have decided to range it from 3 to 80, thereby encompassing a broad spectrum of potential FSM configurations. To ensure the stability and generalizability of our results, we have also settled on the number of turns and repetitions within the game. Based on preliminary tests, we determined that a low number of turns and repetitions, such as 3 and 5, resulted in significant variation in the experimental outcomes, suggesting that the results might not be representative or consistent. To overcome this issue, we opted for 200 turns and 1000 repetitions in our simulation. These figures were chosen after conducting smaller-scale experiments to ascertain the optimal parameters that would yield representative and consistent results.

1.1. Implementation of New Strategy ‘Truth:C’

In introducing new strategies to our experiment, we sought to embody the ideas proposed in our initial framework. Our primary objective was to emulate human thought processes and actions during interpersonal interactions, particularly the concept of trust. When individuals engage in competitive situations, they often remember the outcome of their last encounter. If one party betrays the other in the previous game, there's a higher likelihood of reciprocal betrayal in the subsequent one. To capture this behavioural pattern, we used the opponent's game history as a decision-making criterion, where recent game outcomes carry more weight in influencing the next action. Specifically, we looked at the last three results from the opponent's history, assigning the most recent result a weight of 3, with the weight progressively decreasing for older results. In the implementation, the trust value changes depending on the opponent's actions. If the opponent chooses to betray, the weight is subtracted from the trust value. Conversely, if the opponent opts to cooperate, the weight is added to the trust value. We set a trust threshold at zero: if the trust value is greater than or equal to zero, the strategy opts to trust the opponent, but if the trust value falls below zero, the strategy chooses betrayal. This approach allows us to integrate complex behavioural dynamics into our game theory model. See Figure 1.

In fact, the content of adding trust strategies is basically consistent with the general idea in the proposal, but since we were not aware of using Axelrod and Dojo packages at the time, we focused on the implementation and simulation of genetic algorithms in the proposal. These two packages can be built-in to help us complete some of the processes in genetic algorithms, which makes our experiments much more convenient. However, due to the issue of long matching times each time, the idea of modifying the trust threshold was not implemented. Finally, we added the section on using finite state machines to train the FSM strategy. In this experiment, we validated the impact of different numbers of states on training the FSM strategy. We will focus on the analysis and discussion of the results for the final finite state machine part of the experiment.

```python
def strategy(self, opponent: Player) -> Action:
    if not opponent.history or len(opponent.history) < 3:
        #print("no history or short")
        return C
    else:
        try:
            last_3_moves = opponent.history[-3:]
            k = 0
            for i in range(len(last_3_moves)):
                if last_3_moves[i] == D or last_3_moves[i] == "D":
                    k += -i
                else:
                    k += i
            if k >= 0:
                #print(str(last_3_moves) + ": C")
                return C
            else:
                #print(str(last_3_moves) + ": D")
                return D
        except Exception as e:
            #print(e)
            return C
```

Figure 1. Snapshot of New Strategy 'Truth: C' Code[3]

2. Testing/Evaluation

To evaluate the Trust strategy, we ran a simulation involving 10 different strategies to compete with our own strategy 'Truth: C'. The parameters for the tournament are 100 turns and 20 repetitions, and we took the mean score on each different strategy of Truth: C as an evaluation. A list of other strategies and the evaluations are as follows.

2.1 Strategies used in the tournament

1. Tit for tat: The player will make the move that the opponent previously used.
2. Defector: The player will always defect regardless of the opponent's move.
3. Cooperator: The player will always cooperate regardless of the opponent's move.
4. Grudger: The player will first cooperate, once getting defected, he will always defect.
5. Second by champion: This player cooperates on the first 10 moves and plays Tit for Tat for the next 15 more moves. After 25 moves, the program cooperates unless all the following are true: the other player defected on the previous move, the other player cooperated less than 60% and the random number between 0 and 1 is greater than the other player's cooperation rate.
6. Tranquilliser: Submitted to Axelrod's second tournament by Craig Feathers Description given in Axelrod's "More Effective Choice in the Prisoner's Dilemma" paper: The rule normally cooperates but is ready to defect if the other player defects too often. Thus, the

[3] All of our work can be found in [veseli-kletar/CS765_Project: Prisoners Dilemma Experiment (github.com)]

rule tends to cooperate for the first dozen or two moves if the other player is cooperating, but then it throws in a defection. If the other player continues to cooperate, then defections become more frequent. But if Tranquilizer is maintaining an average payoff of at least 2.25 points per move, it will never defect twice in succession and will not defect more than one-quarter of the time.

7. Second by Otto Borufsen: Strategy submitted to Axelrod's second tournament by Otto Borufsen (K32R) and came in third in that tournament.This player keeps track of the the opponent's responses to own behaviour:

   cd_count counts: Opponent cooperates as response to player defecting.

   cc_count counts: Opponent cooperates as response to player cooperating.

   The player has a defect mode and a normal mode. In defect mode, the player will always defect. In normal mode, the player obeys the following ranked rules:

   If in the last three turns, both the player/opponent defected, then cooperated for a single turn.

   If in the last three turns, the player/opponent acted differently from each other and they're alternating, then change the next defect to cooperate. (Don't block the third rule.)

   Otherwise, do tit-for-tat.

8. Start in normal mode, but every 25 turns starting with the 27th turn, re-evaluate the mode. Enter defect mode if any of the following conditions hold: - Detected random: Opponent cooperated 7-18 times since last mode evaluation (or start) AND less than 70% of opponent cooperation was in response to player's cooperation, i.e. cc_count / (cc_count+cd_count) < 0.7 - Detect defective: Opponent cooperated fewer than 3 times since last mode evaluation.

   When switching to defect mode, defect immediately. The first two rules for normal mode require that the last three turns were in normal mode. When starting normal mode from defect mode, defect on first move.

9. Evolved FSM [size of FSM]: Works as described in Harper, M., Knight, V., Jones, M., Koutsovoulos, G., Glynatsi, N. E., & Campbell, O. (2017). Reinforcement Learning Produces Dominant Strategies for the Iterated Prisoner's Dilemma. Cornell University Library, arXiv.org. https://doi.org/10.1371/journal.pone.0188046
   We will try and train it in different sizes and compare if larger (more complex) strategies are able to compete better.
10. RANDOM: Axelrod included a player that plays randomly Cooperate of Defect in his tournaments, we include it as well to assess how the strategies perform against a completely random strategy.
11. Truth:C: the player will make the decision based on the opponent's last three moves, for a detailed explanation, see section 1.1 above.

2.2 Evaluation

The Truth:C strategy was designed to compete with other strategies in a simulated environment, where multiple rounds of the Prisoner's Dilemma game are played. It has a relatively ideal performance throughout the tournament (ranking 4th out of 11 strategies), with a mean score of 2.77. Referring to Figure 2, Truth:C has an average score of approximately 2.5 when competing with most strategies. However, when competing with strategies such as Random 0.5 and Defector, the result was not satisfied. Our hypothesis is that when competing with Random 0.5, the player would encounter the same number of defecting and cooperating, which made it complex for the player to decide. When playing against Defector, the opponent's decision was always defecting. Therefore the player will never cooperate, and the score for every round is consistently 1.

This evaluation provides insights into the strategy's strengths, weaknesses, and overall competitiveness.

- Win rate: 'Truth:C' did not manage to win a single game during the tournament. Our hypothesis is that since we set the initial move to be cooperating consistently, it has a relatively ideal scoring versus other strategies in the first round in a 'nice' environment (where every player cooperates at the first move). When versus other players, Truth:C never defects on the opponent if the opponent was willing to cooperate.
- Cooperation Rate: Truth:C has a cooperation rate of 86.88%, and therefore it is a cooperative and forgiving approach. It can lead to mutually beneficial outcomes and long-term success. In addition, the high cooperation rate implies the reason why Truth:C has a relatively high performance while not winning a single game.
- Consistency: Referring to Plot 1, the violin graph indicates that Truth:C has the lowest standard deviation comparing with other strategies. Therefore, Truth:C is the most consistent strategy among all candidates.

To conclude the evaluation, our simulation result indicates that Truth:C is ranked 4th out of 11 candidates, and it has an ideal performance competing with most strategies while being the most consistent strategy.

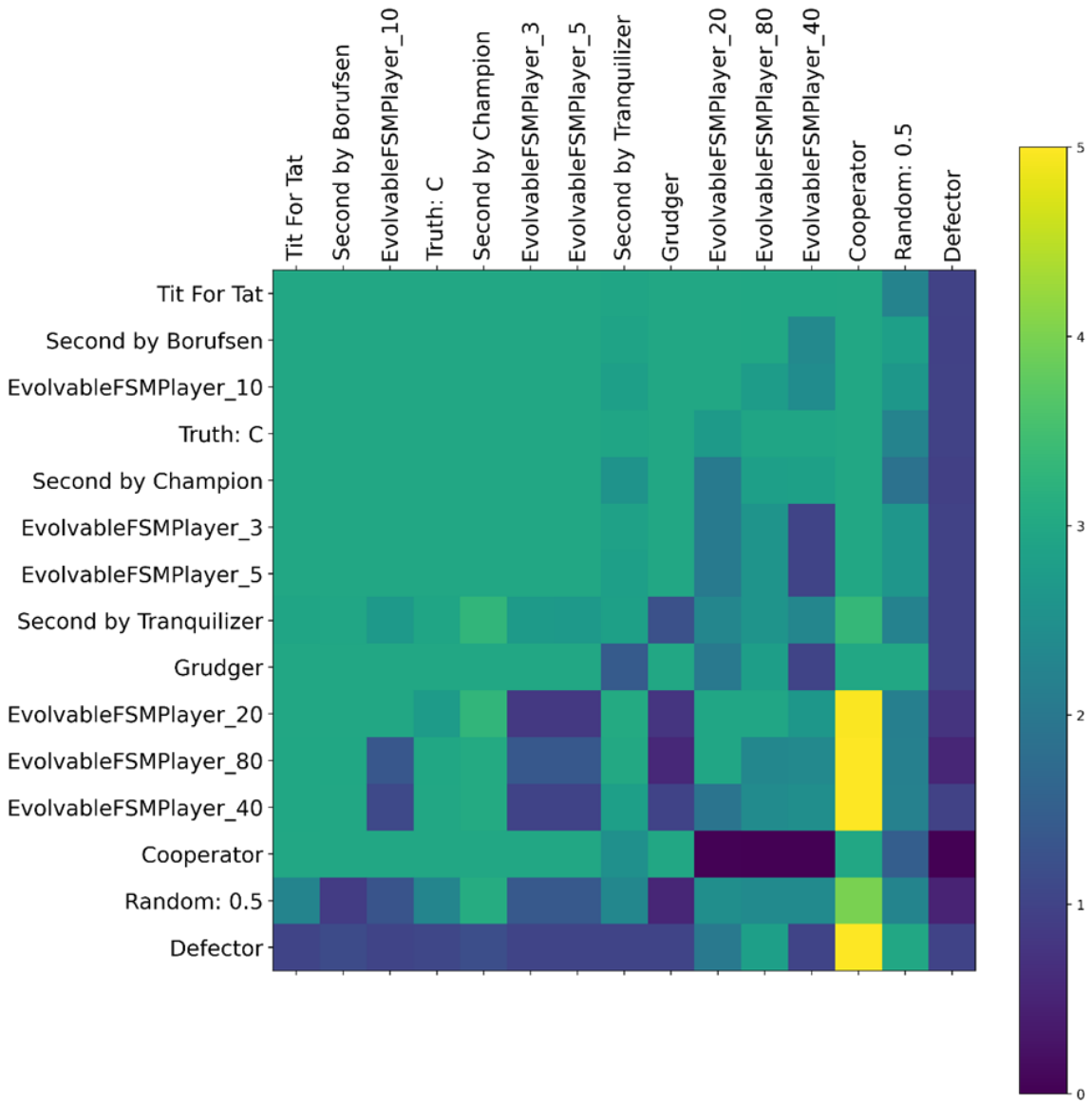
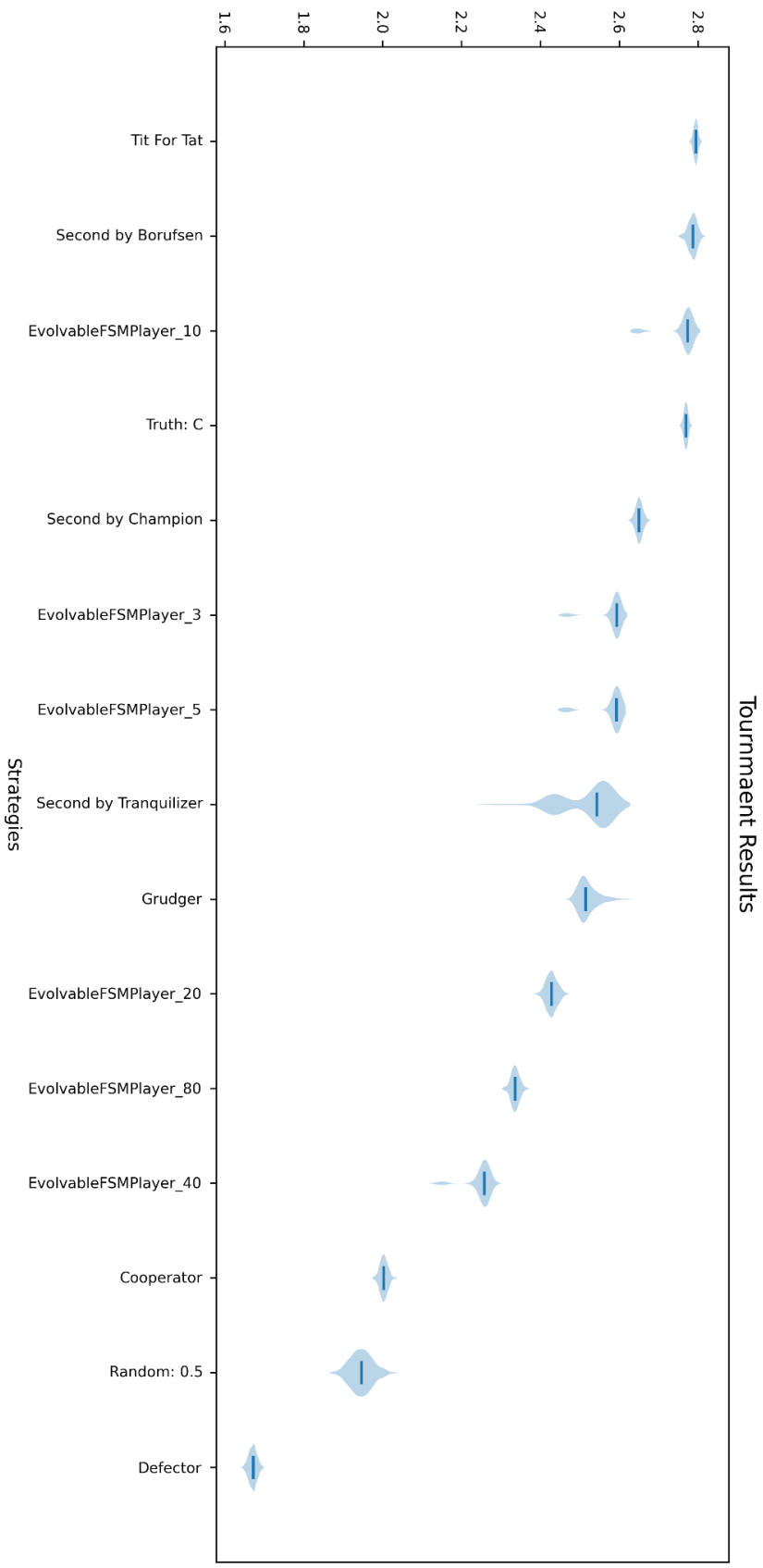


Figure 2: PayOff Matrix of the Axelrod Tournament performed in our experiment.

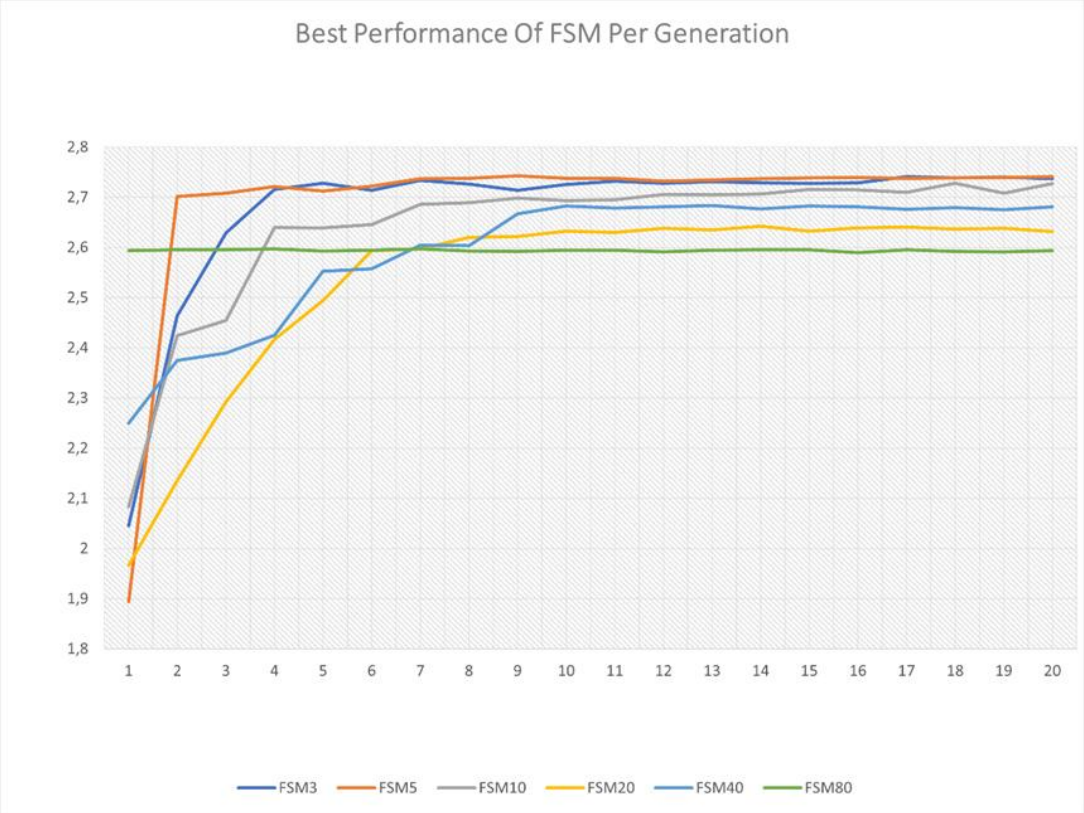


Plot 1: Tournament Results, Best performing by points is on top, and lowest performer on the bottom, respectively.

3. Discussion of Significance

Prisoner's Dilemma is an exciting problem for various reasons; to name one is that it can be applied to many fields outside of mathematics and computer science [7]. We wanted to approach searching for a solution to the Prisoner's Dilemma by finding an optimal Finite State Machine that would perform better than other strategies. We can observe that the best-performing strategy remains Tit-For-Tat, which has been the Winner of Axelrod's First and Second Tournament [7][8]. We also introduced a simple strategy called Truth and observed how it performs in a tournament setting.

We chose Finite State Machine as another approach because we consider how a human would approach the Prisoner's Dilemma when there is more than one strategy to be tackled. However, as we will see later, the optimum of the problem's solution is also dependent on the opponent's strategy.



Plot 2: Here we can see how the maximum score of each FSM generation changes throughout the evolution.

We expected that the Finite State Machines (of various node sizes) would be able to learn how to play the game optimally over multiple generations. If we analyse maximum performance over generations of evolution, we see that the performance growth rapidly stops only after a few generations. Looking at Plot 2, we may observe that after generation 9, the maximum performance stalls at values between 2,6 – 2.7.

After analysing the problem in-depth, this result is what we should expect. If we only look at two problems, the winner Tit-For-Tat and Defector, we see that the optimum results are either 3 or 1, and the mean is 2. However, long-term cooperation will have a better payoff than mutual Defection.

Maximum performance is then dependent equally on our strategy based on the behaviour of other strategies in the tournament.

As described in original papers by Axelrod[7][8], the strategies have three main properties: niceness, forgiveness and provocability. If both strategies are nice, neither of them will play Defect first, and their mutual score will be 3. If both strategies are not nice, as Defector per example, then the mutual score drops towards 1, which is the optimal score for this case. If we decide to cooperate with Defector, this strategy will always be inferior to Defector. Defector has a high win rate for this reason since it will always punish nice strategies first, and then if we decide to forgive the Defect move to Defector, this strategy lowers its score.

As an exciting discovery, we should mention that increasing the number of nodes of FSM does not result in massively better performance either in generating the strategies with GA, neither they able to perform better than the Truth (our strategy), except FSM with ten nodes as we can observe, despite being more complex mental models for playing the game. The explanation could be that we must train the FSM strategies on more opponents to better utilise the more complex model.

On the other hand, instead of modelling the strategy in a way that a person would approach it by solving it with paper and pen, we should consider this more as a model of mind and deciding based on this aspect. By modelling the problem like this, we try to learn a rule with the Random strategy. However, there is no rule for randomness. While a person might notice that the opponent is just playing solely on luck, some could try and exploit this. People base their decisions on luck, which sometimes grants us better results. Despite mathematically low chances, it would be a relevant aspect of how the human mind functions.

Our Truth strategy was designed with simplicity in mind since best-performing strategies tend to perform better than complex ones, as we learn from Axelrod's previous tournaments. Tit-For-Tat still performs better and is impossible to trick into our favour since it punishes Defections for each made by opposing strategy.

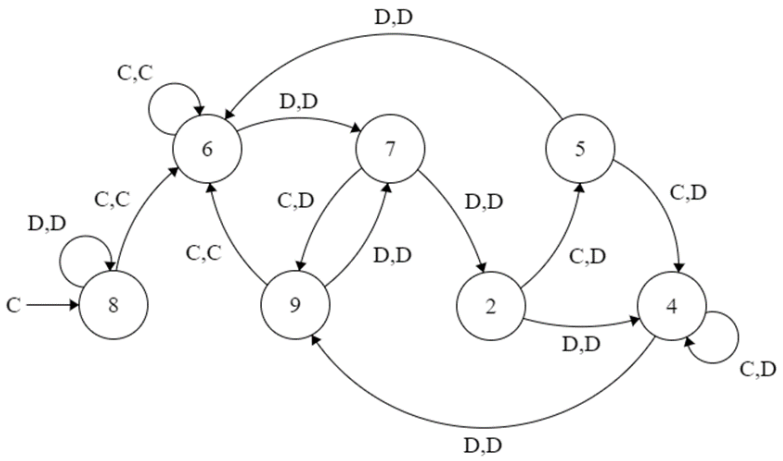


Figure 3: Representation of the evolved strategy of the FSM with 10 nodes. 3 of the nodes are never reached therefore they are pruned away and left out of this schema.

Similar behaviour is observed in FMS of size 10, the best performer amongst FSMs. Despite scoring lower than Tit-For-Tat, we can determine from the graph that it developed the property of niceness. It will not make the Defect move first. Another thing to note is that it will cooperate only in nodes 8,6,9; otherwise, it always Defects. It performed worse because of its complex punishment mechanism, which makes it too complicated to recover when the opponent begins to cooperate again. This behaviour makes them less cooperative in the long run if the opponent punishes Defects because the FHM remains in this loop of Defects.

Another vital thing to note is that it would be an easy score gain and likely ensure a win for the FSM of size ten strategy if it could recognize and punish the Cooperative strategy. We can observe that the Defector and FSM of sizes 20, 40 and 80 do this and gain almost maximal possible points (5). Perhaps through more generations of evolution, this behaviour could be observed.

4. Limitations

We want to observe the impact of different state machine nodes on game results. Furthermore, we discovered that the best score among such evolvableFSMPlayer is an evolvableFSMPlayer with ten states. However, the result is obtained when the game's turn and number of repetitions are fixed. Axelrod changed this in his second tournament so that entries could not use this to win unfairly by evaluating the number of turns taken. Changing the number of rounds and repetitions may result in better results. However, due to the long duration of each experiment, it is also difficult for us to set the range of changes in turn and repetition times too short to observe the results.

Moreover, the trust threshold of our new strategy cannot be dynamically changed. That is to say, and we cannot see the impact of different trust thresholds on game results. Because every time we reset the trust threshold, we must thoroughly re-match it, which still consumes much time. Even if we could conduct multiple experiments, the trust threshold step size set at this point must be significant, and the results may not necessarily be meaningful. This methodology is also different from our ideas in the proposal.

For future research directions, we should find ways to dynamically change the thresholds of the trust strategy to conform more to aspects of mindfulness in the aspect of trust strategy. Axelrod's library has many trust-related strategies, including the consistently high-scoring TitForTat algorithm. Trust is essential to this theory because it relies on other strategies and, as such, determines punishment for opponents. Furthermore, we can also set some other parameters and even add some player personality traits to simulate real human players and real-world situations, where Prisoner’s Dilemma might apply.

References:

[1] Axelrod, R. M. (2006). The evolution of cooperation (Rev. ed.). Basic Books.

[2] Lee, D., & Yannakakis, M. (1996). Principles and methods of testing finite state machines-a survey. Proceedings of the IEEE, 84(8), 1090-1123

[3] Harper, M., Knight, V., Jones, M., Koutsovoulos, G., Glynatsi, N. E., & Campbell, O. (2017). Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma. PloS One, 12(12), e0188046–e0188046. https://doi.org/10.1371/JOURNAL.PONE.0188046

[4] Goldberg, D. E. (David E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Pub. Co.

[5] Pipes, C. D. (2022). Understanding Sink State Formation in Finite State Machines Using the Prisoner's Dilemma. ProQuest Dissertations Publishing.

[6] Fogel, D. B. (1993). Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, *1*(1), 77-97.

[7] Axelrod, R. (1980). Effective Choice in the Prisoner's Dilemma. The Journal of Conflict Resolution, 24(1), 3–25. https://doi.org/10.1177/002200278002400101

[8] Axelrod, R. (1980). More Effective Choice in the Prisoner's Dilemma. The Journal of Conflict Resolution, 24(3), 379–403. https://doi.org/10.1177/002200278002400301