CT421: Project 2: Iterated Prisoner's Dilemma

1 Introduction

The **Prisoner's Dilemma** can be defined as follows:

> Suppose a man with a chest full of gold coins invites you and another player to participate in a game. Each of you has two choices: you can either cooperate or defect. If you both cooperate, each of you receives 3 coins. If one of you cooperates while the other defects, the defector receives 5 coins and the cooperator gets nothing. If you both defect, each of you receives 1 coin. Suppose your opponent cooperates; you could choose to cooperate and get 3 coins, or you could defect and get 5 coins instead—so you're better off defecting. But what if your opponent defects instead? In that case, if you cooperate, you receive no coins, but if you defect, you get at least 1 coin. Thus, no matter what your opponent chooses, you are always better off defecting. However, if your opponent is also rational, they will reach the same conclusion and also defect. As a result, when both players act in their best interests, they end up in a suboptimal situation, each receiving only 1 coin, whereas mutual cooperation would have yielded 3 coins each [1].

This is one of the most famous problems in game theory and it resembles many real-world scenarios, from the Cold War to competing brands or companies and their advertising strategies. However, in these real-life situations, we are not just playing the prisoner's dilemma once; we play it repeatedly. If you defect now, your opponent will know and can use that against you. So, when this game is repeated, what is the best strategy?

2 Implementation Details & Design

2.1 Genome Structure

I defined the genome structure as a strategy class with the following fields:

- `First move`: The initial action taken by the strategy.
- `Responses`: A dictionary that defines how to respond to the opponent's last move or two moves.
- `Hold grudges`: A maximal hold-grudge trait, similar to the friedman strategy, where once the opponent defects, the strategy continues to defect for the rest of the game.
- `Random defection`: This allows the strategy to incorporate an element of sneakiness by randomly defecting with a certain probability, provoking the opponent occasionally. This probability has 10 possible values (0, 0.1, 0.2, ..., 0.9).

This makes the sample space relatively large:

$$2^6 \times 10 = 640.$$

It also enabled me to incorporate some well-known fixed strategies from Axelrod's first and second tournaments, such as `friedman`, `joss`, and `tit for two tats`.

2.2 Fitness

In this prisoner's dilemma setup, the fitness of a genome is determined by how well it performs against all fixed strategies in an Axelrod-style tournament. The more points it accumulates in total across all strategies, the higher its fitness.
Axelrod Tournament Opponents

- **always cooperate**
- **always defect**
- **tit for tat**: Starts off nice and then repeats the last player's move.
- **suspicious tit for tat**: Starts off nasty and then repeats the last player's move.
- **friedman (also known as grudger)**: Starts off nice, cooperates as long as the opponent cooperates, but once the opponent defects, it will defect for the rest of the game.
- **joss**: tit for tat but randomly defects 10% of the time.
- **tit for two tats**: Starts off nice, cooperates throughout, and only defects if the opponent defects twice in a row.
- **suspicious tit for two tats**: Starts off nasty, then cooperates throughout and only defects if the opponent defects twice in a row.
- **random**: Randomly defects or cooperates.
- **fibonacci**: tit for tat but defects on Fibonacci-numbered rounds.
- **reverse fibonacci**: tit for tat but cooperates on Fibonacci-numbered rounds.

(The fibonacci strategies were ones I decided to implement myself simply because the fibonacci sequence often appears in various contexts.)

2.3 Selection

I implemented tournament selection, as required by the assignment, where a subset of the population is randomly chosen, and within that subset, the solution with the highest fitness is selected as a parent.
I also implemented elitism to ensure that the best solution from each generation progresses to the next.

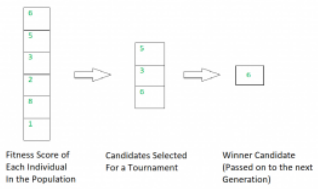


Figure 1: Illustrating tournament selection. [2]

2.4 Reproduction

Crossover

Crossover involves creating two offspring by exchanging traits between two parent strategies. I decided to implement it as follows:

- The **first move**, **hold grudges**, and **random defection rate** of each offspring are swapped between the two parents.
- The **response map**, which determines how a strategy reacts to different game states, is formed by randomly selecting each response from either parent.

Mutation

There is a `MUTATION_RATE` probability that a mutation occurs, and if it does, each trait has a `MUTATION_RATE` probability of being flipped:

- The first move may flip between cooperate and defect.
- The hold grudges trait may flip between True and False.
- The random defection rate may increase or decrease by 0.1.
- Each response in the strategy has a chance of flipping between cooperate and defect.

3 Results and Analysis

3.1 Testing Logical Soundness of the GA (Against fixed strategies for 100 rounds)

(**Note:** These GA tests were performed with a 0.02 mutation rate and a 0.8 crossover rate.) To test the accuracy of my genetic algorithm, I first ran it against always defect. In this scenario, the optimal solution should be always defect, as it maximises the payoff.

```
0-{(1,): 0, (0,): 0, (1, 1): 0, (1, 0): 1, (0, 1): 0, (0, 0): 0}-False-0.5-100
```

This is effectively an always defect strategy in this environment. As expected, the genetic algorithm successfully evolved an always defect strategy, which is the best possible outcome in this environment.
Next, I tested the GA against a tit for tat strategy. Tit for tat is a form of negative reinforcement, meaning that if the GA behaves optimally, it should discover that the best strategy is to always cooperate. Against tit for tat, an always defect strategy would score 104 points, while an always cooperate strategy would score 300 points.

```
1-{(1,): 1, (0,): 0, (1, 1): 1, (1, 0): 1, (0, 1): 0, (0, 0): 0}-False-0.0-300
```

Again, the algorithm finds an always cooperate solution (at least against tit for tat, it will always cooperate).
Lastly, I tested the genetic algorithm in an environment containing always defect and tit for tat strategies. In this scenario, the optimal solution should be tit for tat.

```
1-{(1,): 1, (0,): 0, (1, 1): 1, (1, 0): 1, (0, 1): 0, (0, 0): 0}-False-0.0-399
```

Once again, in this environment, the evolved strategy is effectively tit for tat (as the situations where the opponent's moves are (1,0) or (0,1) cannot arise here).
These results provided strong evidence that my genetic algorithm was functioning correctly and was logically sound.

3.2 Analysis of Evolved Strategies

I carried out tests where I did not initialise the first population randomly. Instead, I initialised the population entirely with defectors. I did not allow grudges or random defections, meaning the first move was always defect and it continued to defect on every subsequent response. I played these genomes only against a tit for tat strategy. Through mutation and then crossover, I wanted to see if the genetic algorithm could evolve from this always defect strategy, which is suboptimal in this environment, into an always cooperate strategy, or at least one that consistently cooperates in this scenario.

```
Best strategy - first move: 1 - responses: {(1,): 1, (0,): 1} - hold grudge: False - randomly defect: 0 = Fitness score: 300
```

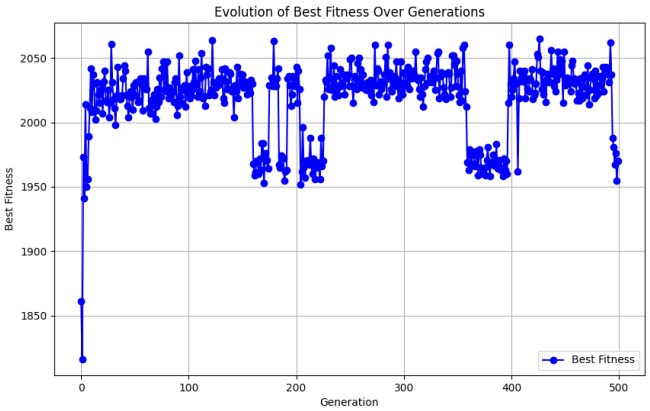
And it did. Through mutation and crossover, the population was able to evolve an always cooperate strategy from an initial population consisting entirely of defectors.
I also tested an initial population of always cooperate strategies against other always cooperate strategies to see if, through mutation, the strategy could evolve into an always defect strategy (which is optimal in that environment).

```
Best strategy - first move: 0 - responses: {(1,): 0, (0,): 1} - hold grudge: False - randomly defect: 0 = Fitness score: 500
```

Again, through mutation and crossover, the GA evolved an always defect strategy (at least in this environment, where its opponent will only cooperate) from an initial population of always cooperate strategies.

3.3 Against Fixed Strategies

I ran the genetic algorithm with a population size of 75 for 500 generations, using a mutation rate of 0.02 and a crossover rate of 0.8. Each tournament lasted for 100 rounds. For each generation, I saved the best strategy's fitness and plotted the evolution of the best fitness over generations. I observed that after about 50 generations, the fitness no longer improved. As a result, for the remainder of my tests, I used 75 generations.



I then ran the GA 10 times and saved the best strategy from each run to analyse which strategies came out on top. After that, I took the best strategy from those 10 runs and tested it in an Axelrod tournament to see where it would rank among its competitors.

3.3.1 Results

```
1 - fitness: 2001 - first move: 1 - responses: {(1,): 0, (0,): 1, (1, 1): 1, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.2
2 - fitness: 2042 - first move: 1 - responses: {(1,): 1, (0,): 1, (1, 1): 0, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
3 - fitness: 2064 - first move: 1 - responses: {(1,): 0, (0,): 0, (1, 1): 1, (1, 0): 0, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
4 - fitness: 2047 - first move: 1 - responses: {(1,): 1, (0,): 0, (1, 1): 0, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
5 - fitness: 2049 - first move: 1 - responses: {(1,): 1, (0,): 0, (1, 1): 1, (1, 0): 0, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
6 - fitness: 2015 - first move: 1 - responses: {(1,): 0, (0,): 1, (1, 1): 1, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.2
7 - fitness: 2073 - first move: 1 - responses: {(1,): 1, (0,): 1, (1, 1): 1, (1, 0): 0, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
8 - fitness: 2051 - first move: 1 - responses: {(1,): 1, (0,): 0, (1, 1): 0, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
9 - fitness: 2070 - first move: 1 - responses: {(1,): 1, (0,): 1, (1, 1): 0, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
10 - fitness: 2063 - first move: 1 - responses: {(1,): 1, (0,): 1, (1, 1): 0, (1, 0): 1, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
```

3.3.2 Key Findings

Among the strategies that evolved in each GA run, `random_defection` was employed by only two strategies, and even then, their defection rates were low (both 0.2). These two strategies also performed the worst among the ten solutions. The eight best solutions had no random defection at all and did not rely on any deceptive tactics. This might seem counterintuitive, as one might expect that being sneaky or unpredictably defecting could be advantageous. However, the results indicate that such an approach is not particularly effective in this setting.

All ten solutions started off friendly—they all cooperated. Surprisingly, however, eight "nasty" solutions emerged (1, 2, 3, 4, 6, 8, 9, and 10), as they responded to either (1,) or (1,1) with a defection. This suggests that these strategies might be the first to defect, making them "nasty."

Another interesting finding was that none of the evolved strategies held grudges. (As mentioned earlier, in my implementation, a strategy can only hold a maximal grudge.) It might seem logical to retaliate indefinitely after one defection, but in the long run, this approach yields poor results.

All strategies chose to defect after being defected against twice in a row, and all strategies decided to cooperate if their opponent defected but then cooperated in the next round (showing that they are forgiving).

Additionally, none of the strategies were complete pushovers. None evolved into an always cooperate strategy, and none adopted an always defect strategy. Eight of the strategies were "nasty," meaning they could be the first to defect, but the other two were nice yet provokable.
Among the two friendly strategies (7 and 5), they ranked first and sixth, respectively, and followed the key principles observed in Axelrod's tournament:

- Be nice (don't be the first to defect).
- Be forgiving (don't hold grudges).
- Be provokable (respond to defections; don't be a pushover).
- Be predictable (strategies that employ random or sneaky defections aren't predictable).

Strategy 7, which ranked first, is effectively very similar to strategy 5. However, strategy 7 forgives an opponent that defects on the first move and then adopts a tit for tat approach, whereas strategy 5 is strictly tit for tat.

3.3.3 Analyzing the Best Solution

I then analysed the best solution among the ten.

```
7 - fitness: 2073 - first move: 1 - responses: {(1,): 1, (0,): 1, (1, 1): 1, (1, 0): 0, (0, 1): 1, (0, 0): 0} - hold grudges: False - random defection: 0.0
```

The best solution seems to be a slightly more forgiving version of tit for tat:

- It cooperates on the first move.
- If the opponent's first move is a defection, it will forgive them.
- Then it will only defect if the opponent defected on their last move.
- It does not employ a sneaky random defection strategy.
- It does not hold grudges.

Next, I tested this solution in an Axelrod-style round-robin tournament, where each strategy played against every other strategy and their scores were totalled.

```
[Axelrod Tournament Results]
Always Cooperate: 2529
Always Defect: 1736
Tit For Tat: 2739
Suspicious Tit For Tat: 2099
Friedman: 2163
Tit For Two Tats: 2663
Suspicious Tit For Two Tats: 2235
Joss: 2042
Random: 2370
Fibonacci: 2105
Reverse Fibonacci: 2692
GA: 2814
```

The GA-evolved solution emerged as the winner. Due to the randomness in the random and joss strategies, different runs of the tournament may yield slightly different results. However, after multiple runs, the GA solution consistently came out on top.
Its closest competitor was tit for tat, which is well known to be highly effective. The reason the GA solution performed slightly better is the presence of suspicious tit for tat in the environment. When tit for tat plays suspicious tit for tat, the latter starts off with a defection, prompting tit for tat to retaliate, reducing their mutual gains. However, when the GA solution plays suspicious tit for tat, it forgives that initial defection, and they establish mutual cooperation for the rest of the game, leading to a higher total score for the GA solution.

3.3.4 More Key Findings

- In that tournament, there were 12 competitors, six of whom were "nice" strategies (always cooperate, tit for tat, friedman, tit for two tats, reverse fibonacci, and the GA-evolved strategy). Five of these six ranked in the top five.
- always defect can never lose against its direct opponent; it can either win or draw. However, in a tournament setting, it finished in last place.
- tit for tat, tit for two tats, and the GA-evolved solution can never *beat* their direct opponents—they can either draw or lose. Yet, in the tournament, these strategies emerged as top performers.
- A strategy is only as good as the environment in which it evolves. When I tested my GA against always cooperate alone, it evolved into an always defect strategy, which was optimal in that specific environment. However, in an Axelrod tournament setting, that approach finished in last place.

4 Part 2

Link to GitHub Repository

References

[1] Wikipedia Contributors (2019) *Prisoner's dilemma*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Prisoner%27s_dilemma (accessed: 1 March 2025)

[2] GeeksforGeeks. (2018). *Tournament Selection (GA)*. [online] https://www.geeksforgeeks.org/tournament-selection-ga/ (accessed: 1 March 2025)