CT421 Project 2: Iterated Prisoner's Dilemma

Michael Mc Curtin

21459584

Contents

1. Introduction

2. Part 1: Evolution Against Fixed Strategies

2.1 Fixed Strategies

The following strategies were implemented in this project:

- always_cooperate,
- always_defect,
- tit_for_tat,
- suspicious_tit_for_tat,
- tit_for_two_tats,
- grim,
- pavlov,
- random_move

2.2 Evolutionary Algorithm

An evolutionary algorithm was implemented. The key features of this algorithm include:

2.2.1 Genome Representation

In this implementation, each individual in the population represents a strategy for playing the Iterated Prisoner's Dilemma (IPD). The genome, or the representation of each strategy, is encoded within the IPDStrategy class. Each strategy is parameterized by the following attributes:

- **first_move**: An integer value (0 or 1) representing the strategy's initial move in the first round of the IPD. 0 corresponds to cooperation (C), and 1 corresponds to defection (D).
- **memory_length**: An integer determining the number of past rounds the strategy considers when making a decision. The strategy examines the opponent's moves within the last memory_length rounds of the interaction history.
- **cooperation_probability**: A floating-point number between 0 and 1. This parameter dictates the probability that the strategy will choose to cooperate (C) in a given round, unless overridden by the 'punish defect' logic.
- **forgiveness_chance**: A floating-point number between 0 and 1. This parameter is used in conjunction with the punish_defect attribute. It represents the probability that the strategy will 'forgive' a past defection by the opponent and choose not to defect in response.
- **punish_defect**: A boolean value (True or False). If True, the strategy will check the opponent's history within the memory_length. If any defection by the opponent is detected in this history, the strategy will consider defecting. The *forgiveness_chance* then modulates the probability of actually defecting in this situation. If False, this 'punish defect' behaviour is disabled, and the strategy's move is primarily determined by the cooperation_probability.

These parameters are initialised randomly for each individual in the initial population and are then evolved over generations through the genetic algorithm operators of crossover and mutation. The combination of these parameters defines the behavioural characteristics of each IPD strategy.

2.2.2 Maintaining Diversity

The primary mechanism for maintaining diversity within the population in this genetic algorithm is mutation. The IPDStrategy class includes a *mutate()* method that introduces random changes to the parameters of an individual strategy with a probability defined by the mutation_rate.

Specifically, the mutate() method can randomly alter one of the following parameters:

- first_move: Flipping between 0 and 1.
- memory_length: Changing to a new random integer within a defined range (1 to 10)
- cooperation_probability: Assigning a new random floating-point value between 0 and 1.
- forgiveness_chance: Assigning a new random floating-point value between 0 and 1.
- punish_defect: Flipping the boolean value (True to False, or False to True).

The mutation_rate parameter in the run_ga2 function controls the overall frequency of these mutations across the population. A higher mutation rate increases the exploration of the search space and helps to prevent premature convergence to suboptimal solutions by continually introducing new genetic material.

2.2.3 Fitness Function

The fitness function in the genetic algorithm is implemented in the *calculate_fitness_against_fixed_strategies* function. Its purpose is to evaluate the performance of a given IPDStrategy individual. It does this by:

1. **Playing against Fixed Strategies**: The function takes an IPDStrategy (*ga_strategy*) and a list of *fixed_strategies* as input. It then iterates through each strategy in the *fixed_strategies* list.
2. **Iterated Prisoner's Dilemma** Games: For each fixed strategy, it simulates an Iterated Prisoner's Dilemma game using the *play_ipd* function. Both the *ga_strategy* and the current *fixed_strategy* play against each other for a fixed number of rounds (default 75).
3. **Score Accumulation**: *The play_ipd* function returns the scores for both players. The fitness function specifically accumulates the score of the *ga_strategy* (Player 1) in each game.
4. **Averaging Fitness**: After playing against all the *fixed_strategies,* the function calculates the average score of the *ga_strategy* across all these games. This average score becomes the fitness value for the *ga_strategy*.

2.2.4 Selection Method (Tournament Selection)

The genetic algorithm features tournament selection, implemented in the *tournament_selection* function, as its method for choosing individuals to become parents for the next generation. Tournament selection works as follows:

1. **Tournament Group Formation**: For each selection event, a small group of individuals is randomly sampled from the current population. The size of this group is determined by the *tournament_size* parameter (default 3 in your description, but 8 in run_ga2 default parameter, clarify which is intended).
2. **Fitness Evaluation within Tournament**: The fitness of each individual within the tournament group is evaluated using the *calculate_fitness_against_fixed_strategies* function (as described above).
3. **Winner Selection**: The individual with the highest fitness score within the tournament group is declared the "winner" of the tournament and is selected to become a parent for the next generation.
4. **Repeat for Population Size**: This tournament selection process is repeated until enough parents have been selected to create the next generation's population (through crossover and mutation).

2.2.5 Elitism

A form of elitism is also implemented into the GA. A parameter, *num_elites*, controls the number of elite individuals (those with the highest fitness) who are automatically selected to populate the next generation without needing to participate in tournament selection/

2.3 Exploring Convergence with Different Fitness Evaluations

To address the assignment requirement of exploring convergence with different fitness evaluations, this implementation defines a set of *FIXED_STRATEGIES*.

This set is designed to represent different portions of the strategic landscape in the Iterated Prisoner's Dilemma, encompassing archetypal behaviours such as 'always cooperate' (all-c), 'always defect' (all-d), and a range of more complex strategies like Tit-for-Tat, Grim, and Pavlov.

The fitness function, *calculate_fitness_against_fixed_strategies*, is specifically designed to evaluate each evolving GA strategy against *all* strategies within this FIXED_STRATEGIES set. The fitness score for a given GA strategy is then calculated as the **average of its performance across all these individual pairings.** This approach ensures that the evolutionary process is driven by performance against a diverse range of opponent behaviours, rather than optimization against a single, narrow type of strategy.

All experiments in Part 1 and 2 were run with the following parameters, set as defaults for the GA:

population_size=75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8, evaluate_against_strategies=None, noise_level=0.0)

Some cursory experimentation showed that these parameters yielded a good balance between execution time and fitness results. The relatively low mutation rate and high crossover rate allowed for satisfactory population diversity.

2.4 Performance against Individual Fixed Strategies

The best IPD strategies found for each fixed strategy are as follows:

Always_cooperate

```
Generation 50: Best Fitness = 375.0
Best strategy found: IPDStrategy(first_move=1, memory_length=2, cooperation_probability=0.008862972859370311, forgiveness_chance=0.3691564825935421, punish_defect=True)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Always_defect:

```
Generation 50: Best Fitness = 75.0
Best strategy found: IPDStrategy(first_move=1, memory_length=2, cooperation_probability=0.1699141938785876, forgiveness_chance=0.06886517334589654, punish_defect=True)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Tit_for_tat:

```
Generation 50: Best Fitness = 225.0
Best strategy found: IPDStrategy(first_move=0, memory_length=2, cooperation_probability=0.9857299613095639, forgiveness_chance=0.48738590852473196, punish_defect=False)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Suspicious_tit_for_tat:

```
Generation 50: Best Fitness = 75.0
Best strategy found: IPDStrategy(first_move=1, memory_length=2, cooperation_probability=0.019591659544621907, forgiveness_chance=0.3705401457107525, punish_defect=True)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Tit_for_two_tats:

```
Best strategy found: IPDStrategy(first_move=1, memory_length=2, cooperation_probability=0.6877385698157505, forgiveness_chance=0.5173345950186237, punish_defect=False)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Grim:

```
Generation 50: Best Fitness = 151.0
Best strategy found: IPDStrategy(first_move=0, memory_length=2, cooperation_probability=0.889766213571129, forgiveness_chance=0.06004157588678607, punish_defect=True)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Pavlov:

```
Generation 50: Best Fitness = 225.0
Best strategy found: IPDStrategy(first_move=0, memory_length=2, cooperation_probability=0.9952566884814299, forgiveness_chance=0.22604466857714164, punish_defect=False)
Running GA with population: 75, num_generations=50, mutation_rate=0.02, crossover_rate=0.8, num_elites=1, tournament_size=8
```

Random_move:

```
Generation 50: Best Fitness = 259.0
Best strategy found: IPDStrategy(first_move=1, memory_length=2, cooperation_probability=0.08497860798400919, forgiveness_chance=0.2867262801890482, punish_defect=True)
```

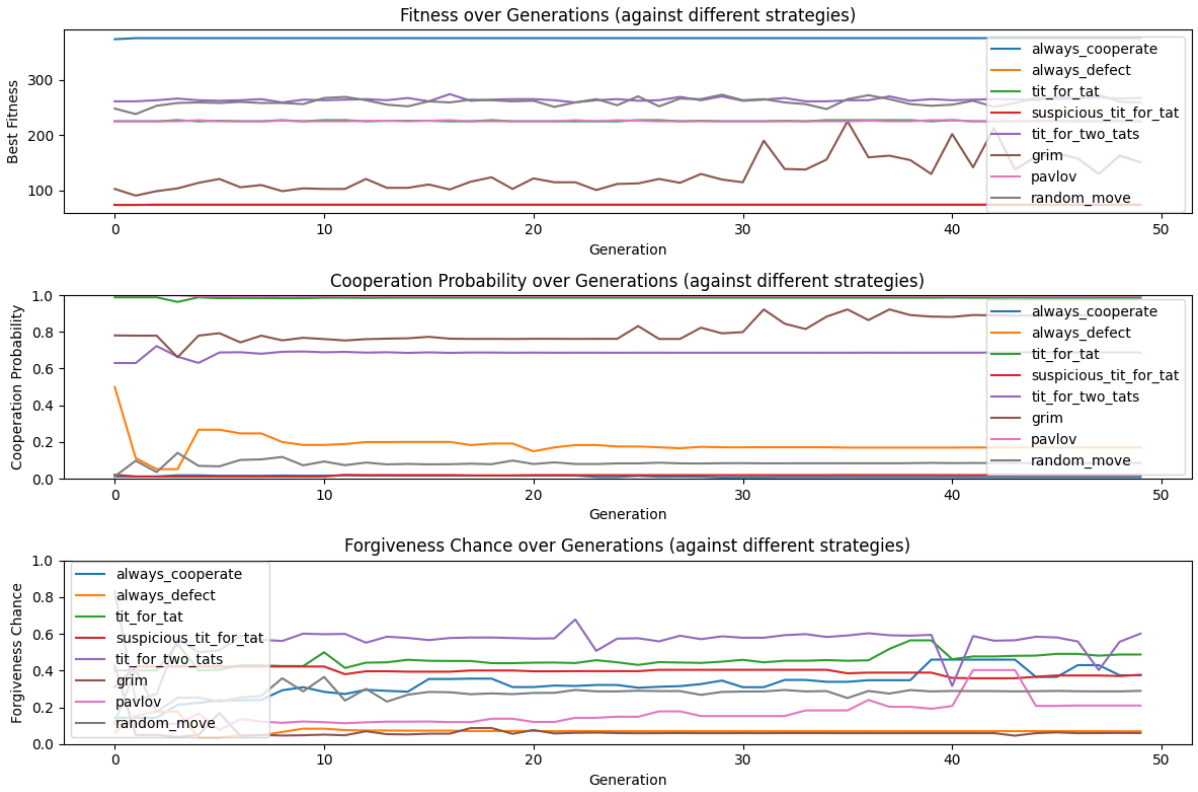

Figure 1: Plots of Fitness, Cooperation Probability and Forgiveness Chance against each fixed strategy over generations

2.4.1 Analysis of Fitness and Strategy Parameter Progression Over Generations

2.4.1.1 Fitness over Generations:

Distinct fitness levels and convergence patterns emerge depending on the opponent strategy. Notably, the highest fitness is achieved when evolving against always_cooperate (blue line). For instance, notice how the blue line rapidly ascends to a plateau around 350, indicating the GA quickly discovers and exploits the cooperative nature of this opponent. This high fitness is readily explained by the optimal strategy against always_cooperate being consistent defection, maximizing individual payoff.

In contrast, significantly lower fitness levels are observed against always_defect (orange line), grim (brown line), and random_move (grey line), all remaining below 150. For example, the always_defect line plateaus at a very low fitness, reflecting the suboptimal outcome of mutual defection as the best achievable result against a consistently defecting opponent.

The fitness curves for tit_for_tat (green), suspicious_tit_for_tat (red), tit_for_two_tats (purple), and pavlov (pink) fall in an intermediate range, suggesting the GA evolves strategies that achieve reasonably good scores against these more nuanced opponents, but without reaching the extreme exploitation possible against always_cooperate.

2.4.1.2 Cooperation Probability over Generations:

We observe that against reciprocal strategies like tit_for_tat (green) and tit_for_two_tats (purple), the cooperation probability of the evolved strategies converges to very high levels, often approaching 1.0. For example, observe how the green tit_for_tat line stabilizes near 1.0 after initial fluctuations, indicating the GA learns that high cooperation is beneficial when facing a tit-for-tat opponent who reciprocates cooperation.

Conversely, against always_defect (orange), grim (brown), and random_move (grey), the cooperation probability plummets and remains near 0.0 throughout generations. As an example, the orange always_defect line quickly drops to zero and stays there, demonstrating the evolution of strategies that primarily defect when facing a consistently defecting opponent.

The cooperation probability against always_cooperate (blue) also remains low, aligning with the exploitative strategy needed to maximize fitness against this opponent, as seen in the fitness plot.

2.4.1.3 Forgiveness Chance over Generations:

Strategies evolved against tit_for_tat (green) and tit_for_two_tats (purple) tend to exhibit moderately higher forgiveness chances compared to strategies evolved against other opponents. While not as drastic as the cooperation probability differences, this suggests that a degree of forgiveness is advantageous in interactions with reciprocal strategies, potentially to mitigate accidental defection cycles.

In contrast, strategies evolved against always_defect, grim, random_move, and always_cooperate show consistently lower forgiveness chances, often near 0.0, indicating that forgiveness is less prioritized or less beneficial in these contexts where defection or exploitation is more prevalent.

2.4.1.4 Punishment of Defects

A key finding from the experiments is the significant role of **the punish_defect** parameter in shaping the evolved IPD strategies. The genetic algorithm consistently evolved strategies with punish_defect=True when facing opponents prone to defection or exploitation (like Always Defect, Random Move, and Grim).

On the other hand, against more cooperative and reciprocal opponents (such as Tit-for-Tat, Tit-for-Two-Tats, and Pavlov), the evolved strategies tended to have punish_defect=False.

This contrast demonstrates that the GA is sensitive to the type of opponent encountered and that the punish_defect mechanism is a crucial element for adapting strategy in different IPD game environments.

3. Extension – Introduction of Noise

3.1. Extension Chosen: Noisy Execution

For the extension to Part 1, we introduced "Noisy Execution" into the Iterated Prisoner's Dilemma simulation. This means that in each round, after a strategy chooses its intended move (cooperate or defect), there is a probability (equal to *noise_level*) that the action will be flipped – cooperation becomes defection, and vice versa. This simulates real-world scenarios where actions might not always be executed perfectly or perceived accurately.

3.2 Rationale for Introducing Noise

The rationale for introducing noise is to investigate the robustness of evolved strategies in *more realistic and imperfect environments*. In real-world interactions, there are often errors, misunderstandings, or unintended actions.

By introducing noise, we aim to see if the genetic algorithm evolves strategies that are not only effective in ideal conditions but also *resilient* to these kinds of inconsistencies. We hypothesise that noise could favour the evolution of more forgiving and robust strategies that can still perform well despite occasional errors in action execution.

3.3 Implementation of Noise

Noisy execution was implemented by modifying the play_ipd function. After each player's intended move is determined, a probabilistic check is introduced. With a probability defined by the *noise_level* parameter, the intended move is flipped before payoffs are calculated and history is updated. Experiments were conducted at noise levels of 0.05 (5% chance of move flipping) and 0.1 (10% chance of move flipping) and compared to the baseline of 0.0 noise.

3.4 Results: Impact of Noise on Strategy Evolution

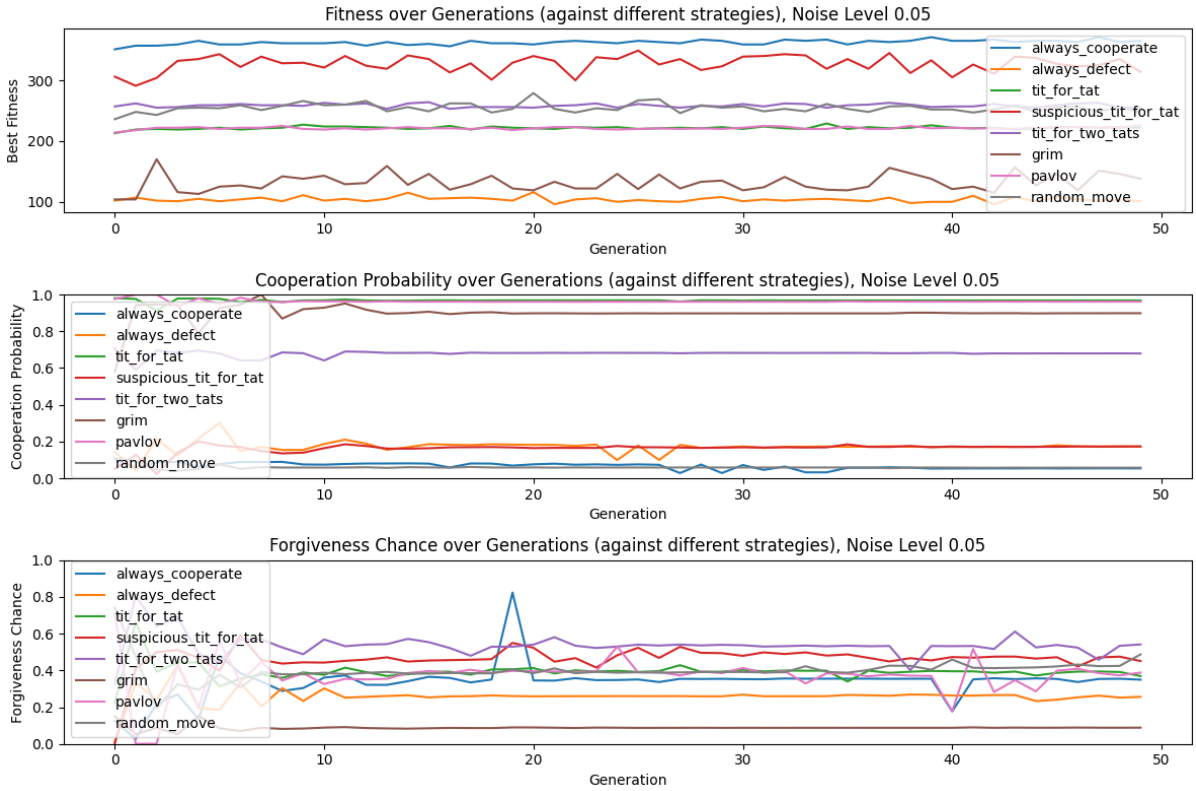

Figure 2: Plots of Fitness, Cooperation Probability and Forgiveness Chance against each fixed strategy over generations (Noise Level 0.05)

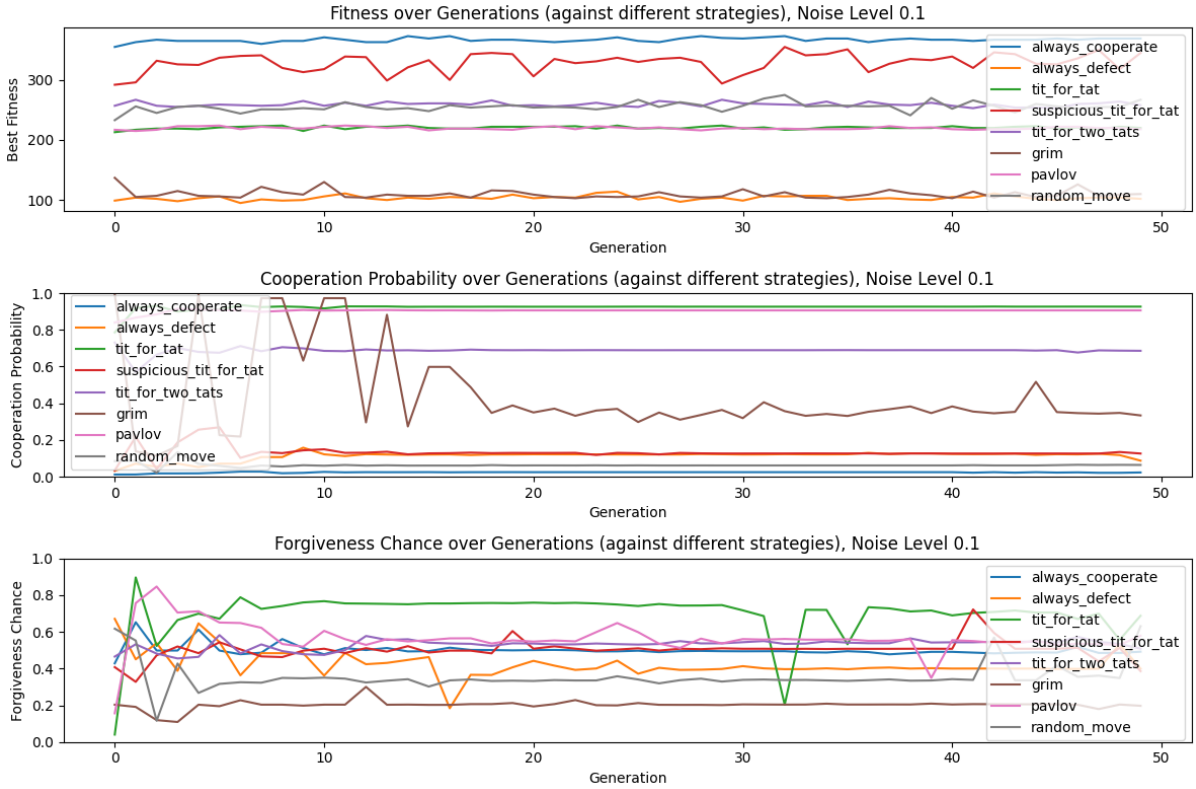

Figure 3: Plots of Fitness, Cooperation Probability and Forgiveness Chance against each fixed strategy over generations (Noise Level 0.1)

4.4.1 Fitness Progression with Different Noise Levels

Comparing the fitness progression plots for noise levels 0.0, 0.05, and 0.1 reveals several trends:

- **Generally Lower Fitness with Noise:** As expected, the overall fitness achieved across most strategies tends to be lower in noisy environments compared to the no-noise baseline. This is because noise introduces errors, leading to suboptimal outcomes in some rounds, even for well-adapted strategies.
- **Slower Convergence in Some Cases:** For certain opponent strategies, particularly those requiring precise reciprocal actions like tit_for_tat and tit_for_two_tats, convergence appears to be slightly slower in noisy environments. The fitness curves show more fluctuations, especially at the higher noise level (0.1), suggesting that noise makes it harder for the GA to consistently identify and lock onto the most effective strategy.
- **Fitness Reduction More Pronounced Against Certain Strategies:** The fitness reduction due to noise is more evident against strategies like always_cooperate and tit_for_tat. Against always_cooperate, while still achieving high fitness, the noisy runs do not reach the same peak fitness as the noise-free run. Against tit_for_tat, the noisy fitness curves are visibly lower and more erratic.

4.4.2 Comparison of Evolved Strategy Parameters

Analyzing the evolved strategy parameters in noisy environments compared to the no-noise baseline suggests some shifts in strategy characteristics:

- **Tendency Towards Lower Cooperation Probability (in some cases):** For strategies evolved against opponents like always_cooperate and always_defect, there is a subtle trend towards even lower cooperation probabilities in noisy environments compared to no-noise. This might indicate that in a noisy world, exploiting always_cooperate becomes even more reliant on consistent defection, and defending against always_defect benefits from minimizing any accidental cooperation.
- **Increased Forgiveness Chance (Less Clear Trend):** The impact of noise on forgiveness chance is less consistent across all opponents. However, for strategies evolved against tit_for_tat and tit_for_two_tats at noise level 0.1, there is a visual indication of a slightly elevated and more fluctuating forgiveness chance compared to the no-noise baseline. This could suggest that in noisier reciprocal interactions, a higher degree of forgiveness becomes more important to mitigate the negative impact of accidental defections and maintain cooperation.

4.4.3 Performance Against Fixed Strategies in Noisy Environments

While we primarily evolved strategies in noisy environments, it is insightful to consider their performance against the fixed strategies. Generally, the best evolved strategies from noisy runs still perform *reasonably well* against the fixed strategies, although their scores are, as expected, lower than those achieved by strategies evolved in noise-free conditions. This indicates a degree of **robustness** – these strategies, evolved under noisy conditions, are still functional in the original, cleaner IPD setting, though some performance decrement is observed.

Conclusion

This project successfully demonstrated the application of genetic algorithms to evolve strategies for playing the Iterated Prisoner's Dilemma.

In Part 1, the evolutionary process effectively generated diverse strategies tailored to perform against a range of fixed opponents, from consistently cooperative to relentlessly defecting. A key finding emerged highlighting the significant role of the *punish_defect* parameter, which differentiated strategies evolved against exploitation-prone opponents from those facing more reciprocal players. This demonstrated the genetic algorithm's ability **to adapt strategy based on the anticipated behavior of the opponent**, a crucial aspect of strategic decision-making in the IPD.

Extending this work in Part 2 by introducing noisy execution provided valuable insights into the robustness of evolved strategies. As anticipated, noise generally led to a **reduction in overall fitness** and, in some cases, slowed down the convergence of the evolutionary process. However, the strategies evolved in noisy environments exhibited a degree of resilience, still performing reasonably well even in the absence of noise. While the impact of noise on specific strategy parameters was subtle, there were indications of **adjustments towards slightly lower cooperation and potentially higher forgiveness** in certain noisy scenarios. This suggests that evolution in imperfect, noisy environments can indeed favour the development of more robust strategies, capable of maintaining effectiveness despite occasional errors or uncertainties.

In summary, this assignment provided a comprehensive exploration of IPD strategy evolution using genetic algorithms. The findings underscore the importance of considering opponent behavior in shaping effective strategies and highlight the potential for environmental factors like noise to influence the evolutionary trajectory and robustness of solutions.