UNIVERSITY OF GALWAY

Project 1: Evolutionary Search (GAs)

Cathal Lawlor

March 11, 2025

1 GITHUB REPOSITORY

Github repository with code available here

2 IMPLEMENTATION DETAILS & DESIGN CHOICES

2.1 Tournament Selection and Mutation

I used tournament selection for selecting parents for crossover. For my testing, I used a value of 5 for k, as it was a good balance between exploration and exploitation. I used a mutation rate of 0.1, to allow for a good amount of exploration, but usually it did not result in significant deviations.
As I elaborate on later 3.1.1, I had to implement an evaluation function where the score was the average of the agent competing against all the fixed strategies, as tournament selection would always discount any algorithms who were competed on the `Always Defect` strategy.

2.2 Intialisation and Crossover

I used a random initialisation for the population, with each gene being a random float between 0 and 1, representing the probability of cooperating given a certain history permutation. I had a uniform crossover method, which randomly allocated a gene from either parent to the child.

2.3 Termination condition

I implemented a max number of generations to run, but I also added a clause, that if the algorithm hasn't improved by more than 0.015% in the last 60 generations, it will terminate early. This was to avoid the genetic algorithms getting stuck in local optima, or to stop when it had plateaued and there wasn't any changes.

3 PART 1: FIXED STRATEGIES

The agent had a memroy of 2, and only awareness of the enemys moves.

3.1 Fixed strategies

I trained the agent on the following fixed strategies:

- **Always Cooperate** - Would always cooperate no matter what the agent did.
- **Always Defect** - Would always defect no matter what the agent did.
- **Tit for Tat** - Would do what the agent did on the previous turn.
- **Adaptive** - Always cooperate, unless the agent defected more than twice in the last 5 moves.
- **Spiteful** - 30% chance of defection the first turn, and had a (0.05 x num_agent_defections) probability of defection for all other moves (ignoring the agents first move).

3.1.1 Generalised training on all fixed strategies

I trained a generalised agent on all the fixed strategies, with a generalised evaluation function. I did this as tournament selection would always discount any algorithms who were trained on the `Always Defect` strategy. Tournament selection wouldn't picked those agents, even if it was a perfect response to `Always Defect`, as it results in a really low payout relative to payouts other agents have when faced with 'easier' strategies.

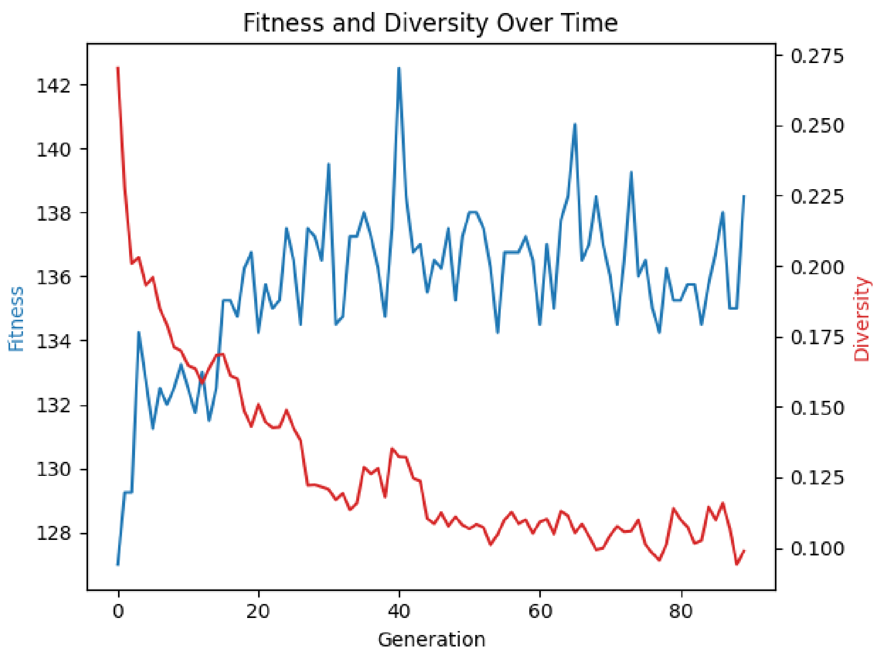

Figure 3.1: Generalised Trained Alogrithm Tested On All Fixed Strategies

(Fixed Strategy)	**Agent Score**	**Fixed Strat Score**
Always Cooperate	172	117
Always Defect	50	50
Tit-for-Tat	139	139
Adaptive Strategy	167	122
Spiteful Strategy	120	130

Table 3.1: Generalised agent performance against each fixed strategy

With the stopping condition in place, the evolution stopped early at generation 89, as the fitness had plateaued, and the diversity of the population had converged.
As we can see, both in the graph3.1 and the results table3.1, the agent performed well against all the fixed strategies, except for the always cooperate strategy.
It isn't a surprise it struggled with the always cooperate strategy, as the agents training on the adaptive, spiteful and tit-for-tat strategies will have biased it's strategy for when an opponent has played `CC`. We can see this in the table3.2 representing the probability of the agent complying based on the opponent history. If the memory was increased, it would almost certainly be able to differentiate the always cooperate strategy from the others.
The agent had a average fitness of 135.71, when averaged over all the fixed strategies (which however, isn't normalised for the different max possible results for each strategy), the table above, is a much better representation of the agent's performance.

(History permutation)	Probability of Cooperate
""	0.49
C	0.86
D	0.11
CC	0.74
CD	0.94
DC	1.0
DD	0.0

Table 3.2: Best generalised agent strategy genome

3.1.2 Specific agents against other fixed strategies

For fun, for each fixed strategy, I trained an agent against only that specific strategy, and then had the best evolved agent play all of the other strategies.
Some notable results: unsurprisingly, the `Always Cooperate` strategy performed well when it was tested against itself and the and `Always Defect` strategy, as it always was going to select the highest reward option, being always defect. With this, it held it's own though against the other strategies.
The `Always Defect` strategy performed the worst, as it only knew what to do when the other agent defected twice in a row. The only wins it had were when the other potential combinations were used (e.g. DC, CC) and it happened to have a random value initialised that proved to be lucky.
The adaptive and tit for tat strategies are interesting to look at in how they performed. `Tit-for-Tat` was quite compliant, and matched equal scores, except for when it was confused by `Always Defect` (as it not had any training experience with repeated `DD`). `Adaptive Strategy` did well with `Always Cooperate`, but lost miserably out to `Always Defect`. Otherwise, it matched well for the other two tests.

4 Part 2: Extension

Disclaimer: I didn't read the assignment spec initially and actually made a much more complex genome for part 1, and actually didn't have too much to extend for part 2. (I had a memory of 2, and probabilities rather than binary decisions) Here is what I have changed:

- Increased memory from 2 to 6.
- Good Samaritan Award
- Coevolution

4.0.1 Memory increase

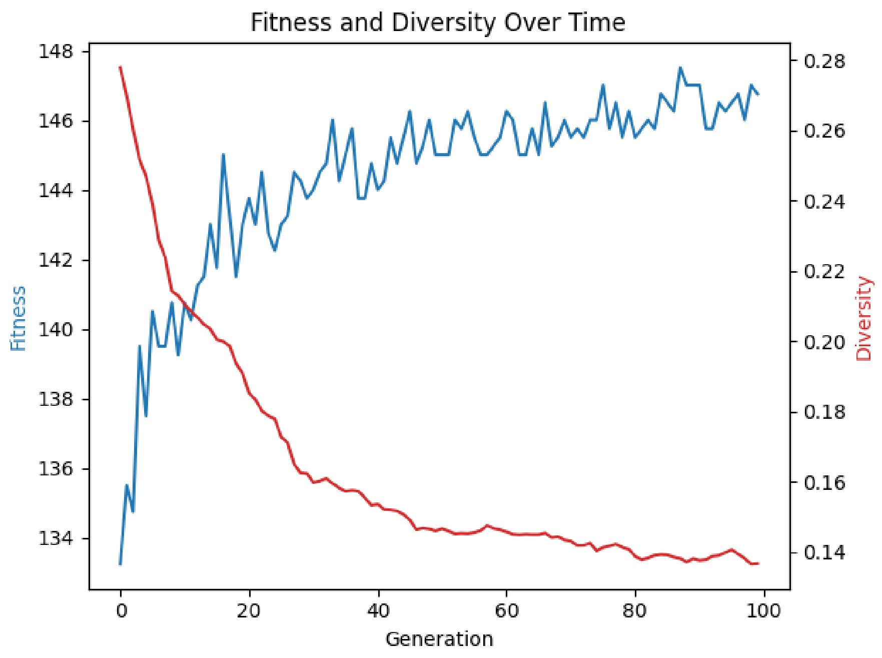

Figure 4.1: Generalised agent with more memory against Tested On All Fixed Strategies

I increased the memory from 2 to 6, as I wanted to see how the agent would perform with a larger memory. It had a big impact on the agents performance, as it was able to differentiate between strategies much better, and was able to respond to the `Always Cooperate` strategy much better, as it wasn't being confused by other strategies like the `Spiteful` or `Adaptive` strategies. It was able to differentiate between the `Always Cooperate` strategy and the

(Fixed Strategy)	**Agent Score**	**Fixed Strat Score**
Always Cooperate	248	3
Always Defect	49	54
Tit-for-Tat	130	130
Adaptive Strategy	140	110
Spiteful Strategy	124	124

Table 4.1: Generalised agent with more memory, 100 generations

SOME NOTABLE PIECES FROM THE GENOME We can see how the algorithm was able to differentiate what to play for the `Always Cooperate` strategy, with a representation of `History: CCCCCC, Strategy Value: 0.0`, where it was always going to defect. This had it score 248, versus the 172 with the shorter memory.
For example then, with the `Spiteful Strategy` having a higher chance of initially defecting with a 0.3 probability (spiteful doesn't take into account the first agent move), the agent has learnt that if it keeps it's nose clean, the spiteful algorithm will keep cooperating. At 5 moves, it has `History: DCCCC, Strategy Value: 0.91`, basically developing a strategy to try to keep working with spiteful, rather than have spiteful increase it's chance of defection (which spiteful will do for each agent defection).

The first run 4.1, only had 100 generations to evolve with. Interestingly, when I removed the early stopping condition and had the agent evolve for 300 generations, it did not make any significant improvement, and performed worse in some instances - for example, against the spiteful algorithm.

(Fixed Strategy)	Agent Score	Fixed Strat Score
Always Cooperate	244	9
Always Defect	50	50
Tit-for-Tat	138	133
Adaptive Strategy	157	102
Spiteful Strategy	91	126

Table 4.2: Generalised agent, more memory, 300 generations of evolution.

4.1 Good Samaritan Award

I added an award to the agent, that will reward it with the `num_round x 3.5` of the score, if it only has only cooperated on every turn. This would also happen in only 55% of the cases.
This means, it will have it will have the slightly better than the max possible reward. I did this, as I was interested in what would the agent do, and how the diversity would hold up.

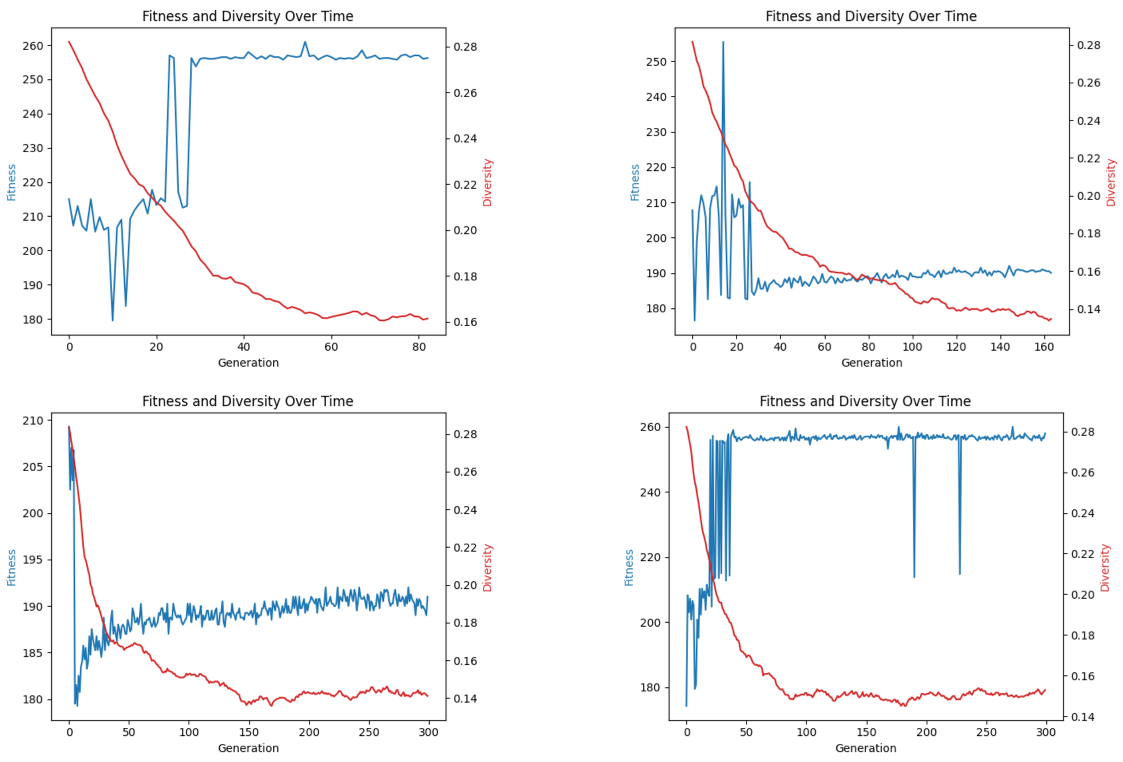

Figure 4.2: Good Samaritan Award Testing

With the good samaritan configuration, I evolved the agent for 300 generations, with a memory of 6. As you can see in the above figure 4.2, from 4 runs, twice did the agent evolve to find the higher reward system, and twice did it actually find it, but then evolve to not take advantage of it.
Additionally you can also see where the agent found the higher reward system, accidentally evolved to not take advantage of it, but quickly re-evolved to take advantage of it again.
The diversity score of the populations isn't noticably any different to other runs without this 'good samaritan' award, where as per usual, the diversity converges at around generation 40. From this, the reward system is a highly divise factor in the population, and the agent will either quickly adapt to it or shun it.

5 Coevolution

With the two other enhancements, I wanted to see how the agent would perform in a coevolutionary setting.
I firstly ran the coevolutionary training with just the increased memory, where it initially plunges in performance, but slowly recovers back to a reasonable score. As the agent irons out the random noise in it's initial population, it starts to perform better where it will sometimes give itself a higher reward (by accident probably), which is worth more than just continually defecting.

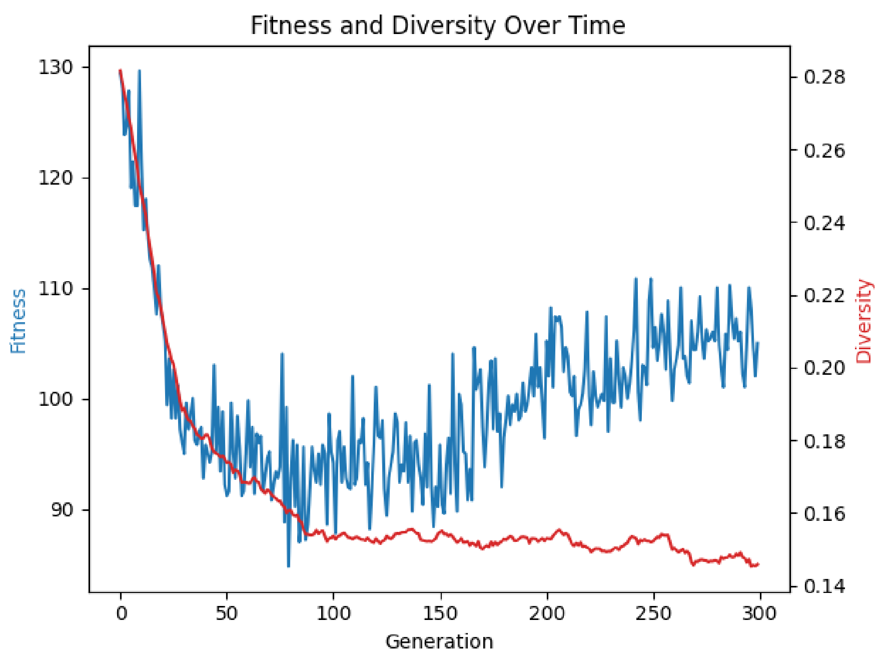

Figure 5.1: Coevolutionary training with increased memory

I then ran the coevolutionary training with the good samaritan award, and the increased memory. The agents did not engage with the good samaritan award with it's initial implementation at all.
I removed the probability element, awarding it everytime that the agent had a run with no more than 3 defections. This still had little impact, so I decreased the round count from 50 to 15.
Only then did the agents start to engage with the good samaritan award.

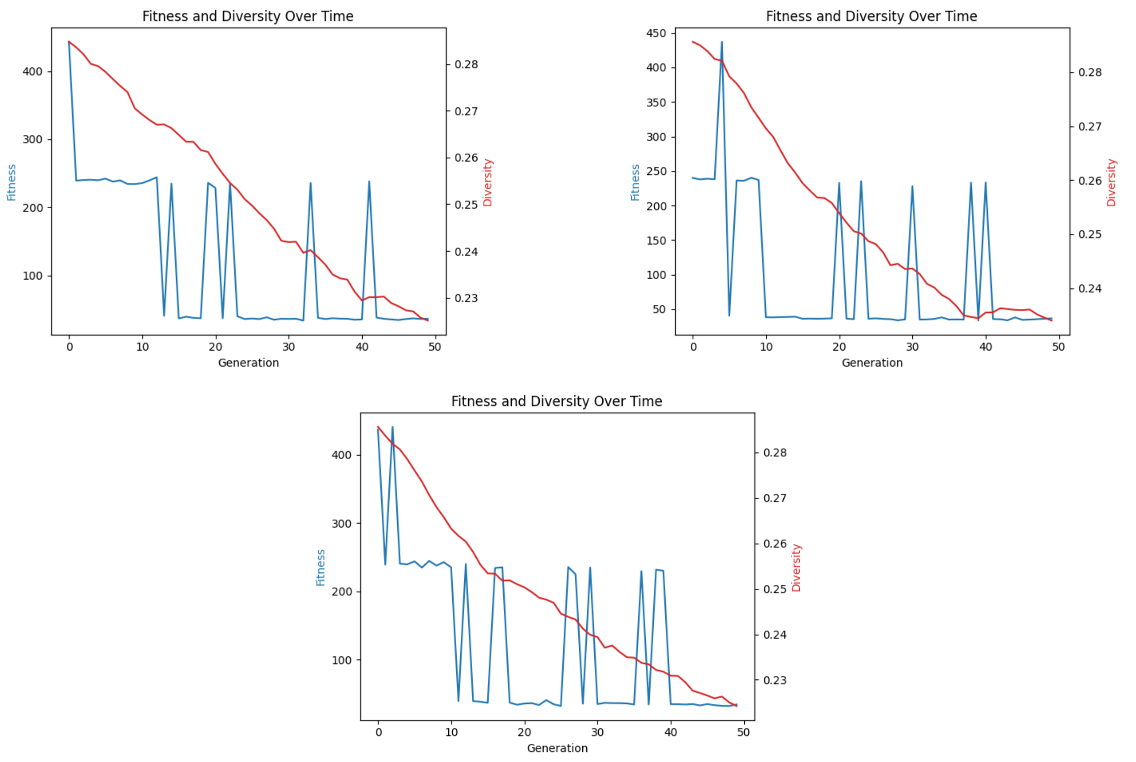

Figure 5.2: Coevolutionary training with good samaritan award

Even after all the changes, the agents still usually didn't engage with the good samaritan award on every run, and would evolve away from an initial position of achieving it for around 10 generations.
I guess from this, my agents are just too selfish to be good samaritans ;)

5.1 Coevolution against each other

Running without all the good samaritan changes, I pitted 5 coevolutionary agents against each other. Surprisingly, I found that the agents did not just always defect against one anohter, and had quite close scores. The agents were able to find a balance between defecting and cooperating, resulting in higher payoffs overall and not just getting the worst score from both defecting. The scores in order of the agents were: 321, 321, 318, 314, 298.

6 Conclusion

I had good fun with this mini project.
Coevolutionary training was interesting, as it was able to find a balance between themselves to result in higher payoffs, even if it meant losing out to itself for the odd turn.
The effects of the memory size difference also were insightful, as it opened up a lot more nuanced strategies. If I had more time, I would have liked to have seen how the agents would have performed with an awareness of it's own moves, and not just the opponents. (although, you will have to make sure the memory doesn't become too large, as it will scale really badly with the number of possible history permutations). This does lead into an idea space that basically is one of using a neural network to approximate the strategy, (which would be interesting in it's own right).