1. **For the same rule, deflate values are as follows:**

    Example rule 1 = [[(0.05, 0.815, 1), (0.661, 0.777, 1)], [(0.194, 0.773, 0), (0.726, 0.828, 1), (0.415, 0.85, 1), (0.482, 0.748, 1), (0.708, 0.821, 0), (0.448, 0.693, 1), (0.131, 0.418, 1), (0.117, 0.578, 0)], [(0.051, 0.097, 0), (0.314, 0.625, 0)]]

    118128
    111509
    115879
    108970
    113877
    113822
    118746
    112141
    112643
    109045

    Example rule 2 = [[(0.859, 0.936, 1), (0.26, 0.63, 0), (0.664, 0.827, 0), (0.341, 0.891, 0)], [(0.721, 0.742, 1), (0.412, 0.583, 1), (0.717, 0.773, 1), (0.789, 0.881, 1)], [(0.467, 0.967, 1), (0.483, 0.569, 1), (0.148, 0.28, 1), (0.456, 0.614, 1), (0.232, 0.761, 1), (0.642, 0.927, 1)]]

    113270
    113395
    108466
    110480
    104928
    112743
    114889
    110058
    114020
    108853

    All values have a significance of 4 digits which is not a large deviation that we have seen in the curves before. So it is not the deflate values!
    This proves the deflate calculations are fine (that is, our mnca.py, which is returning the deflate values, is fine) OK!

2. Functions of the Genetic Algorithm are also working okay, as we already saw in the exhaustive debugging: https://github.com/s4nyam/emnca/tree/main/emnca/emnca/outputs/outs_12march

3. What I think personally is that the elite is getting lost for some reason.

4. Figured out the problem, the problem was actually indeed that we were loosing elites. After debugging I knew that elites were getting lost. For initial 1-2 generations it was difficult to identify but actually the list of

    ```
        best_individual =
    copy.deepcopy(population[best_individual_index]) # elite chromosome
    ```

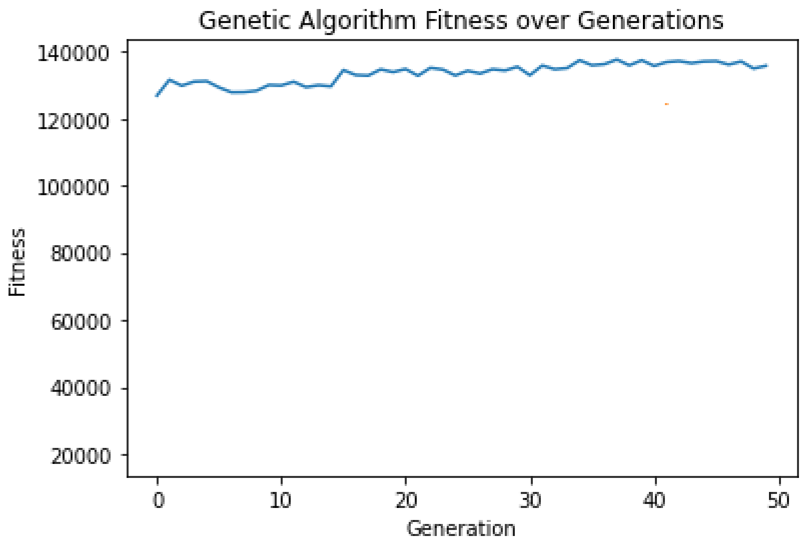
    This is how I am preserving elite now.

5. This version of code is stable for now! https://colab.research.google.com/drive/1iJZ5acoOOLxmsxcd-l0WN0CYmTk4Bm8C#scrollTo=8amjHGnKpL4K

6. It takes **3.8 hours** to run complete simulation from end to end. For “p=10 and g=50”

7. Curves show that best fitness has become stable BUT! Average and worst fitnesses also needs deepcopy! But introducing deepcopy for worst and average means that we want to preserve those individuals for future generations too? We do not want that right? Instead it should have happened through the evolution that the population becomes closer to the so-called “convergence”

8. The graph looks like for best plot:



9. Yes, it is indeed true that LSTM could not produce interesting results. I tried for just out of curiosity to make it learn more and more via curriculum learning but looks like did not work out. (In fact random rules were better than this) It is known to be quite difficult to learn interesting rules with NNs, see here for an example https://arxiv.org/pdf/2009.01398.pdf and https://colab.research.google.com/drive/1_9bAuVnxepO9q5taOZ_EwttstIRg9DqN?usp=sharing (# FOLLOW ONLY GLIDER RULE / REPLICATE GoL may be for dataset, manageable for few steps, may be we could extednt to MNCA)

10. A simple step wise approach of Genetic Algorithm:
    a. For each generation
    b. Fitness of each individual in the population is calculated using a function called calculate_deflate
    c. Best individual in the population is identified and preserved for historical purposes, along with its fitness value, which is stored in the best_fitness_history
    d. The average fitness and worst fitness are calculated and stored in the average_fitness_history and worst_fitness_history lists, respectively
    e. The best individual in the population is preserved using elitism, which involves copying the best individual to the next generation without any modification
    f. The rest of the individuals in the next generation are created using a combination of selection, and mutation. NO CROSSOVER
    g. The selection process uses a roulette wheel selection method, where the probability of an individual being selected for the next generation is proportional to its fitness value
    h. The selected individuals are then mutated to create new individuals, which are added to the next generation.
    i. Finally, the current population is updated with the new generation, and the process is repeated for the next generation

11. A small note for deepcopy:
    a. In the code, deepcopy is used to create a new copy of the best individual in the population, which is then added to the new generation without any modification (i.e., using elitism).
    b. Without using deepcopy, only a reference to the best individual would be copied to the new generation, and any changes made to the individual in the new generation would also affect the original individual in the old generation.
    c. This would lead to unexpected behavior in the algorithm, as the best individual in the old generation could be changed by the mutation process in the new generation, causing the algorithm to not converge to the optimal solution.
    d. Therefore, using deepcopy is crucial to ensure that the best individual is preserved accurately and independently in the new generation, and any changes made to the individual in the new generation do not affect the original individual in the old generation.

DEGREE OF HEIRARCHY - MELANIE MITCHELL BOOK
How do we quantify complexity at hierarchy level.
COARSE GRAINED as hierarchical
Evolving with coarse graining and relating - DEGREE OF HEIRARCHY - MELANIE MITCHELL BOOK