Laboratory 3

Variant 2

Group 12

By Andrea Amato and Dimitrios Georgouisis

Introduction

This report details the implementation and experimentation of a genetic algorithm (GA) to optimize the Rastrigin function, a common benchmark problem in evolutionary computation. The optimization process uses Rank Selection, a Gaussian mutation operator, and a random interpolation for crossover.

Implementation

The GA was implemented in Python, utilizing the NumPy library for efficient mathematical computations. We initialized the population with candidates randomly generated within the range $[-5, 5]$. Rank selection was used, which maintains diversity and mitigates the risk of premature convergence on local optima by systematically and proportionally rewarding fitter individuals while preventing their dominance. Crossover, guided by the `crossover_rate` parameter, was employed to mix genetic information from pairs of parents using a random interpolation technique as described by the formula:

$$\begin{cases} x_{o1} = \alpha x_{p1} + (1-\alpha)x_{p2} \\ y_{o1} = \alpha y_{p1} + (1-\alpha)y_{p2} \end{cases}, \begin{cases} x_{o2} = (1-\alpha)x_{p1} + \alpha x_{p2} \\ y_{o2} = (1-\alpha)y_{p1} + \alpha y_{p2} \end{cases}$$

where o_1, o_2, p_1, p_2 refer to offspring 1, offspring 2, parent 1, parent 2 respectively, and α is a random number. Mutation was applied gene-wise with a Gaussian distribution, with the `mutation_strength` parameter defining the standard deviation, to introduce variability and promote exploration.

Experiment 1: Finding genetic algorithm parameters

We experimented with various parameter settings to find an optimal configuration for our GA. We varied one parameter at a time while keeping the others at default values, i.e. the values we used in our first trial.
Table 1 summarizes the parameters tested and the resulting fitness values. For the random generation of the population in the following trials we used `seed = 27`.

Population Size	Mutation Rate	Mutation Strength	Crossover Rate	Number of Generations	Best Individual genes	Best individual fitness
100	0.01	0.2	0.9	100	$-9.948 \cdot 10^{-1}$, $2.552 \cdot 10^{-3}$	$9.963 \cdot 10^{-1}$
200	0.01	0.2	0.9	100	$1.683 \cdot 10^{-8}$, $-4.740 \cdot 10^{-8}$	$4.991 \cdot 10^{-13}$
500	0.01	0.2	0.9	100	$-1.771 \cdot 10^{-10}$, $2.305 \cdot 10^{-10}$	$< 10^{-15}$
100	0.02	0.2	0.9	100	$6.495 \cdot 10^{-8}$, $7.999 \cdot 10^{-9}$	$8.491 \cdot 10^{-13}$
100	0.04	0.2	0.9	100	$9.948 \cdot 10^{-1}$, $9.941 \cdot 10^{-1}$	1.990
100	0.01	0.1	0.9	100	$1.823 \cdot 10^{-3}$, $2.630 \cdot 10^{-3}$	$2.032 \cdot 10^{-3}$
100	0.01	0.5	0.9	100	$1.712 \cdot 10^{-2}$, $8.739 \cdot 10^{-3}$	$7.327 \cdot 10^{-2}$
100	0.01	0.2	0.8	100	$4.889 \cdot 10^{-3}$, $3.812 \cdot 10^{-4}$	$4.771 \cdot 10^{-3}$
100	0.01	0.2	0.7	100	$3.914 \cdot 10^{-5}$, $5.317 \cdot 10^{-5}$	$8.648 \cdot 10^{-7}$
100	0.01	0.2	0.9	200	$-9.948 \cdot 10^{-1}$, $9.548 \cdot 10^{-4}$	$9.951 \cdot 10^{-1}$
100	0.01	0.2	0.9	500	$-9.948 \cdot 10^{-1}$, $9.548 \cdot 10^{-4}$	$9.951 \cdot 10^{-1}$

Table 1 – Parameter settings and results

As it is noticeable from *Table 1*, the best fitness value was obtained in our third test, for which the algorithm achieved a value that fell below the threshold of numerical precision available in the computing environment

(Fig.1), indicating the algorithm's effectiveness in converging on the global optimum of the Rastrigin function, which it is located at the origin $(0, 0)$.

It was also evidenced how the first and last two configurations, which only differ by the number of generations used by the GA, were trapped in local optima (*Fig.2)*, and that altering any other parameter, which enhanced exploration, proved effective in enabling the algorithm to escape the local minima (*Fig.3)*.

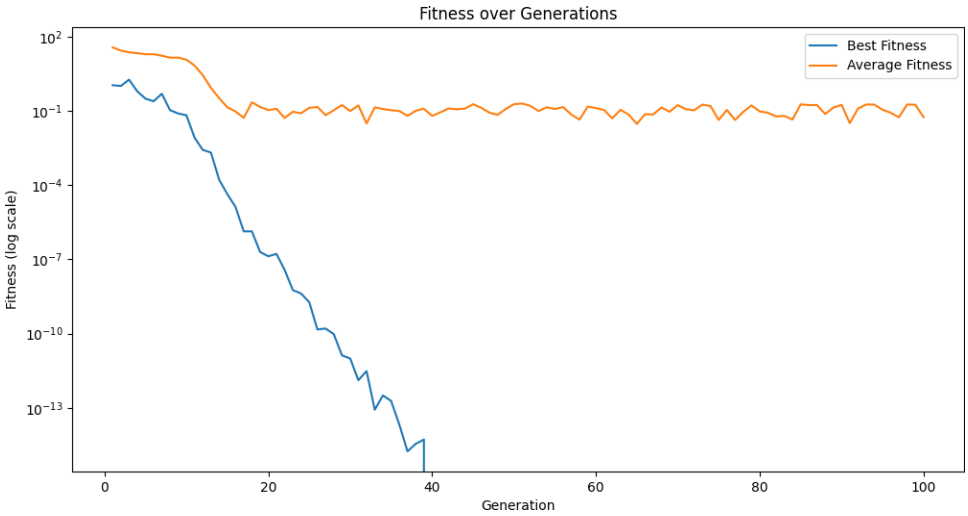

Fig.1 – The best fitness solution achieved, passing the limit of numerical precision available

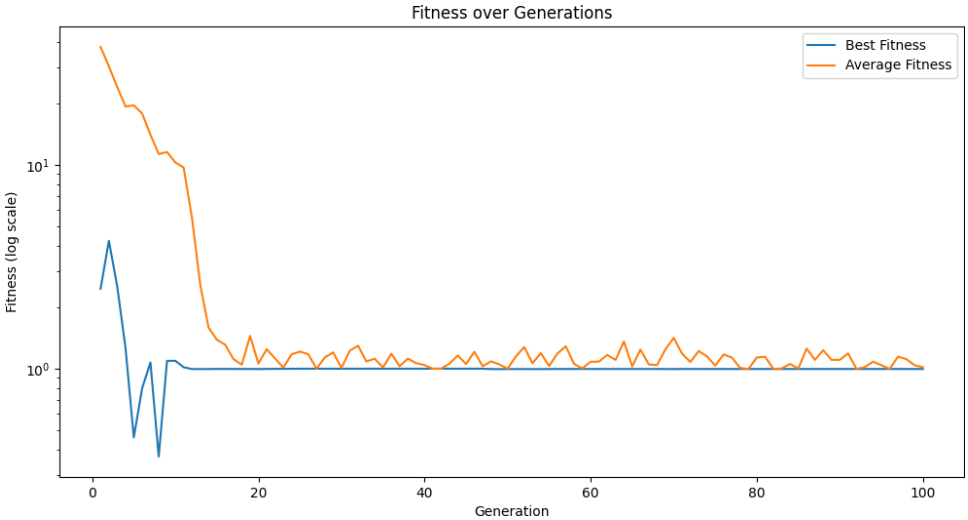

Fig.2 – First configuration leads the GA into getting trapped in local optima

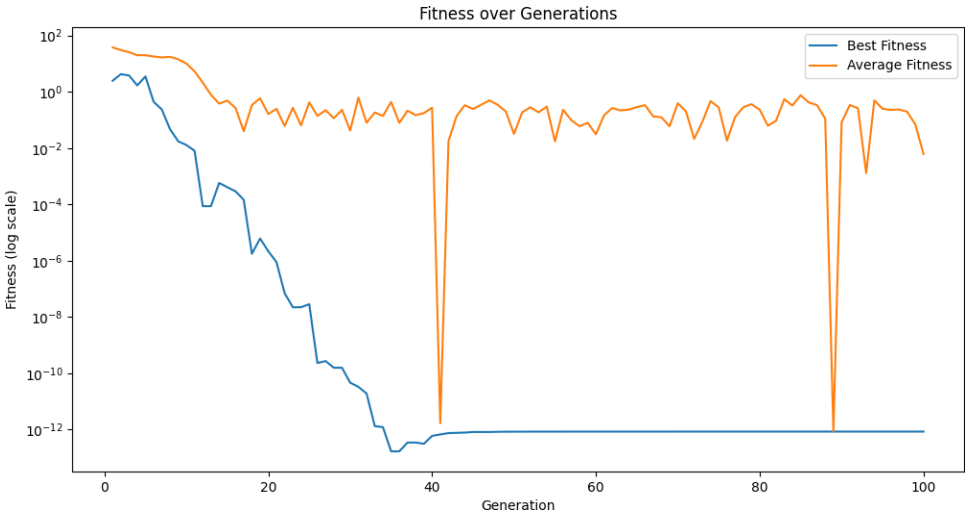

Fig.3 – Increasing mutation rate in the fourth configuration allows the GA to jump out of local optima

Experiment 2: Randomness in genetic algorithm

To assess the impact of randomness, the GA was run with the best parameter set from Experiment 1 across five other different random seeds (seed 27 was kept for comparison). The robustness of the GA was evaluated based on the variability of the results. The outcomes are displayed in *Table 2*.

Seed	Best individual genes	Best individual fitness	Average fitness value (last population)
27	$-1.771 \cdot 10^{-10}$, $2.305 \cdot 10^{-10}$	$< 10^{-15}$	$5.591 \cdot 10^{-2}$
18	$-6.359 \cdot 10^{-11}$, $-1.217 \cdot 10^{-11}$	$< 10^{-15}$	$2.060 \cdot 10^{-1}$
42	$-7.802 \cdot 10^{-4}$, $-3.364 \cdot 10^{-4}$	$1.432 \cdot 10^{-4}$	$4.488 \cdot 10^{-2}$
80	$7.311 \cdot 10^{-12}$, $8.504 \cdot 10^{-10}$	$< 10^{-15}$	$1.632 \cdot 10^{-1}$
33	$5.151 \cdot 10^{-5}$, $-1.827 \cdot 10^{-5}$	$5.926 \cdot 10^{-7}$	$2.643 \cdot 10^{-2}$
58	$9.288 \cdot 10^{-11}$, $1.605 \cdot 10^{-10}$	$< 10^{-15}$	$6.325 \cdot 10^{-2}$

Table 2 – Results obtained for different seeds

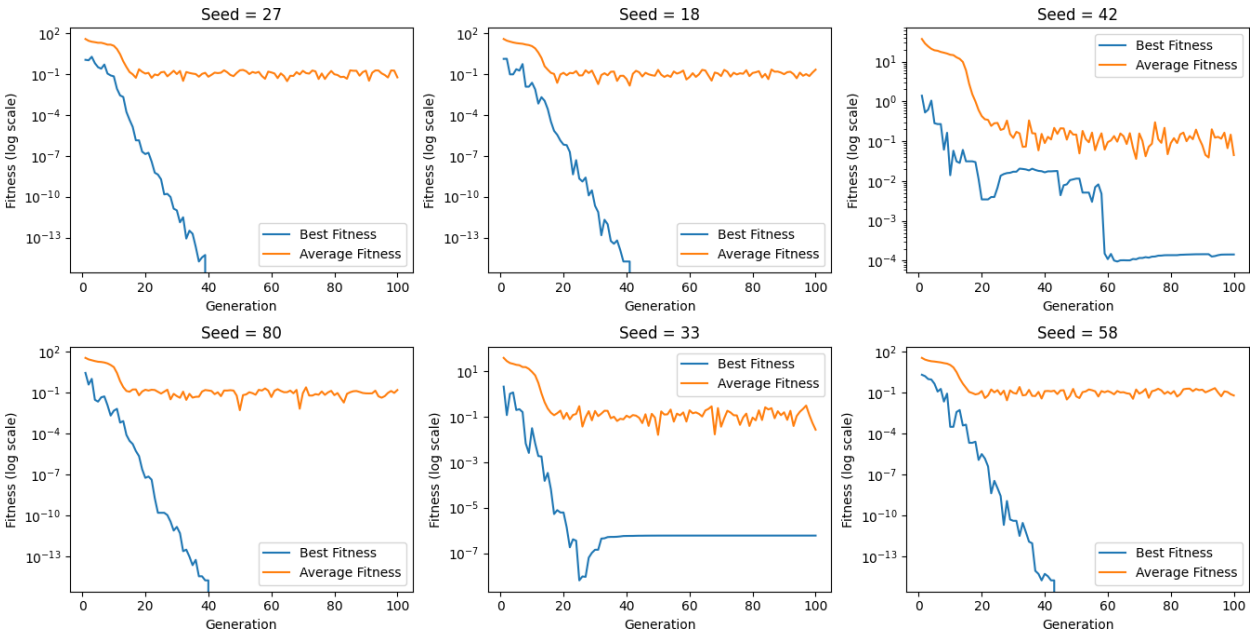

Fig.4 – Performance of the GA for different seeds

The above table shows how seeds 18, 80 and 58 all achieved fitness values below the numerical precision threshold, indicating that the algorithm consistently found solutions close to the global optimum across these runs. The other solutions, for seed 42 and 33, while not as optimal, still demonstrate a good approximation, showcasing the robustness of the algorithm against randomness.

We have determined that the run with seed 18 resulted in the most optimal fitness. This is evidenced by the proximity of the fittest individual's genes to the origin, $(0, 0)$, which is the known global minimum for the Rastrigin function.

Aggregating the results across all seeds *(Figure 5)*, we derived an average best fitness value of $2.397 \cdot 10^{-5}$, offering a collective measure of the GA's performance that still reflects the general efficacy of the genetic algorithm in navigating the solution space. The observed standard deviation of approximately $2.520 \cdot 10^{-1}$ suggests a moderate level of variability in performance across different seeds. This indicates that there is a noticeable influence of the initial seed on the final outcome.

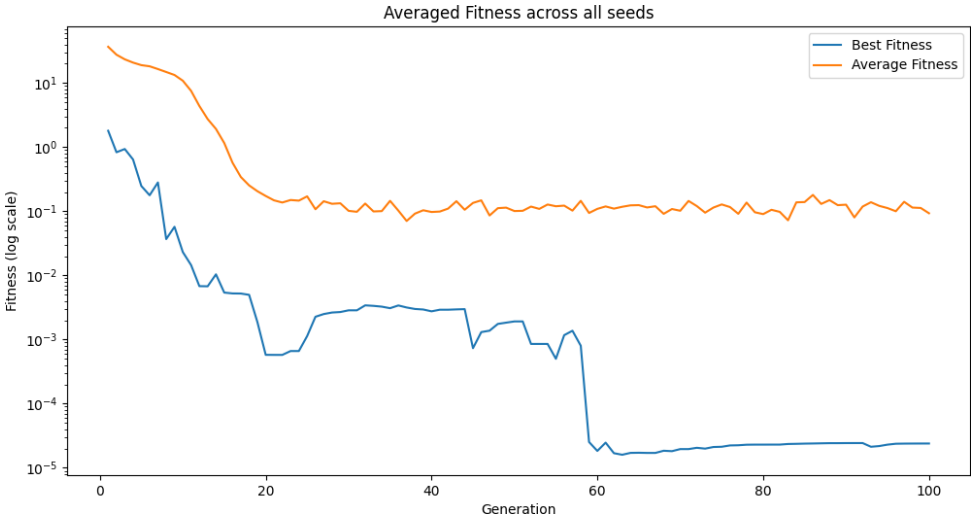

Fig.5 – Aggregated fitness across all seeds

We ran again the algorithm with reduced population size— 50%, 25% and 10% of the original value—to assess the correlation between this parameter and the GA's performance *(Figure 6)*. The results, which were averaged across all seeds, are presented in *Table 3*.

Population	Best Fitness	Average Fitness (last population)	Standard deviation of best fitness
500	$2.397 \cdot 10^{-5}$	9.328	$2.520 \cdot 10^{-1}$
250	1.662	2.464	$5.109 \cdot 10^{-1}$
125	1.692	2.253	$7.245 \cdot 10^{-1}$
50	$8.503 \cdot 10^{-1}$	9.450	1.241

Table 3 – Aggregated results with varying population sizes

Observations here indicate that maintaining a large population of 500 helped guiding the algorithm towards the global minimum, while smaller populations appeared to confine it within local optima. Moreover, the increased standard deviation underscores the algorithm's fluctuating stability across various seeds, implying that a limited candidate pool reduces genetic diversity and, by extension, the algorithm's ability to reliably identify the global optimum.

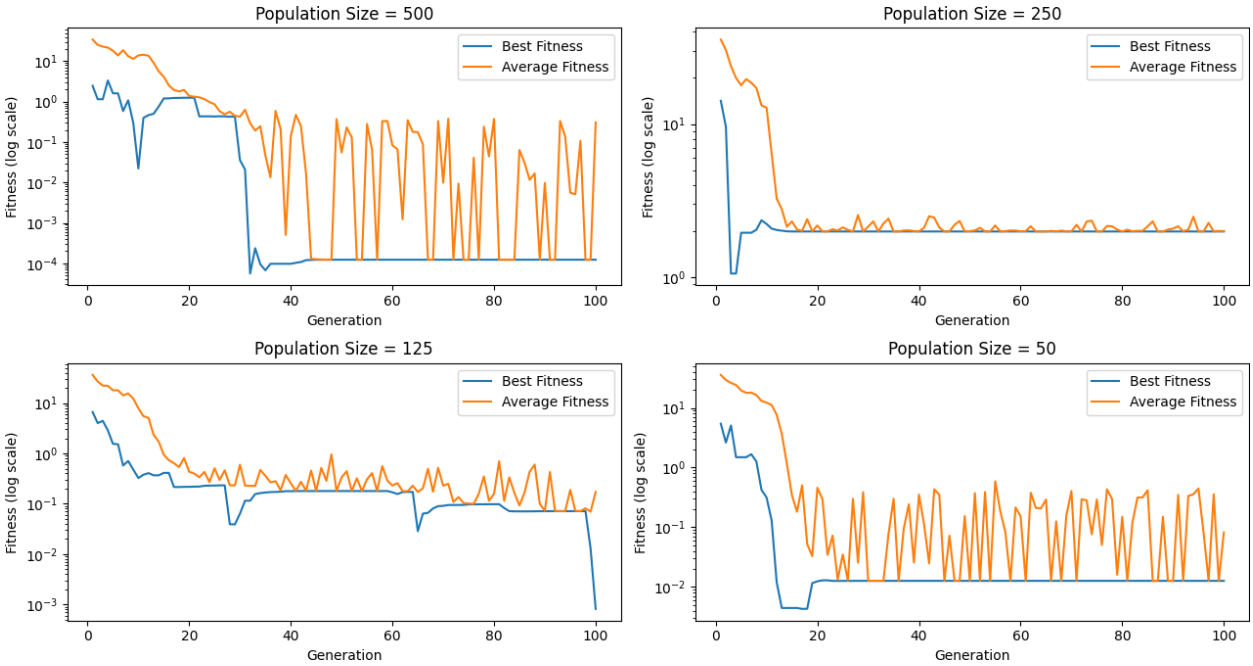

Fig.6 – Aggregated results with different population sizes

Experiment 3: Crossover impact

For this experiment we use the following configuration:

Population Size	Mutation Rate	Mutation Strength	Crossover Rate	Number of Generations	Seeds
500	0.01	0.2	0.3, 0.5, 0.75, 0.9, 1.0	100	27, 18, 42, 80, 33, 58

Table 4 – Information about our GA model's configuration for experiment 3

The crossover rate's impact on the GA's performance was investigated by varying it and observing the changes in the fitness values. Before creating the following plots we averaged our best and average fitness values across all seed values, so as to not draw misleading conclusions due to the behaviour of our algorithm under specific circumstances/seeds.

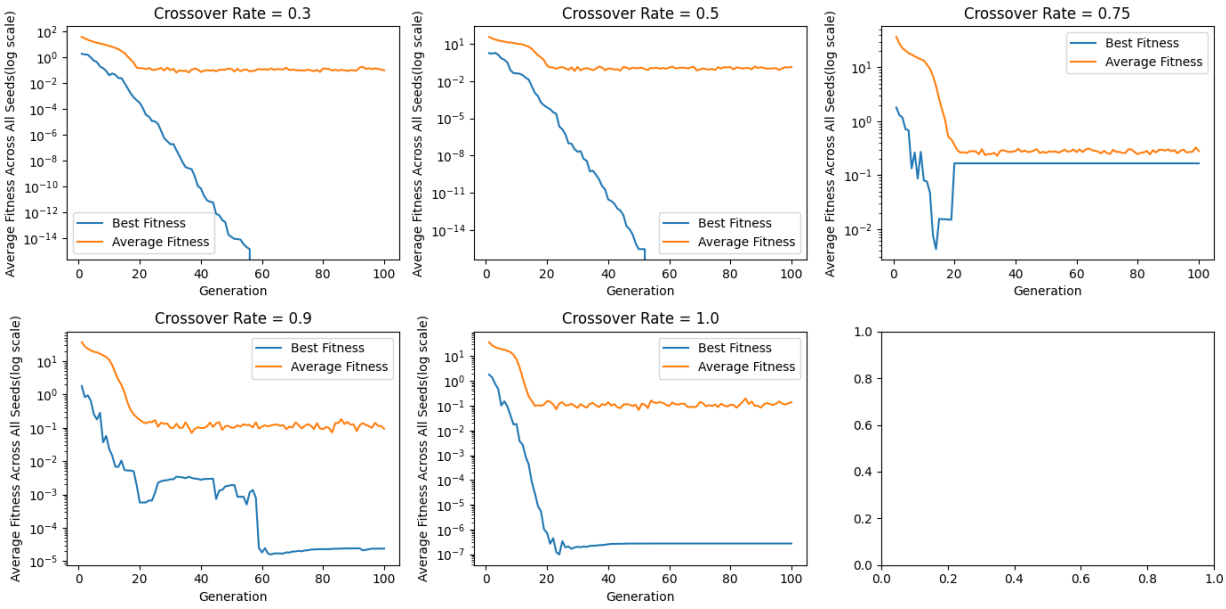

Fig.7 – Different crossover rates result in different ways of development in fitness behaviors.

Crossover Rate	Best Fitness (Final)	Average Fitness (Final)
0.3	0	0.096
0.5	0	0.14
0.75	0.166	0.28
0.9	2.4 * 10^(-5)	0.093
1.0	2.74 * 10^(-7)	0.138

Table 5 – Numerical values of interest for different crossover rates

We can make some quite interesting observations from the above image. Firstly, low crossover rates seem to converge quite early to the best solution, while the overall population fitness does not seem to improve from that point on. We may attribute this result to the fact that low crossovers mean that good solutions are less likely to be altered by individuals that are far off the optimum point.

The middle crossover rate value did not manage to approach the optimum point but kept the average and best fitness quite close.

Lastly, large crossover values need a few generations before reaching convergence, but they manage to give good results. Because of the higher crossover rate, they produce more points that can be searched and thus

search a larger part of the search space. This helps them avoid local optima (the 0.75 value seems to get stuck at a local optimum) and also find a good solution while searching a bigger part of the search space.

Finally, we should mention that all crossover rates that lead to optimal solutions also give quite a low average fitness value for the whole population, which is expected, since individuals keep getting closer to the optimum and crossovers between them will just create offspring that are also close to good positions.

Experiment 4: Mutation and convergence

For this experiment we use the following configuration:

Population Size	Mutation Rate	Mutation Strength	Crossover Rate	Number of Generations	Seeds
500	0.01, 0.02, 0.04	0.2, 0.4	0.9	100	27, 18, 42, 80, 33, 58

Table 6 – Information about our GA model's configuration for experiment 4

The mutation rate and strength's impacts on the GA's performance was investigated by varying them and observing the changes in the fitness values. Before creating the following plots, we averaged our best and average fitness values across all seed values, so as to not draw misleading conclusions due to the behaviour of our algorithm under specific circumstances/seeds.

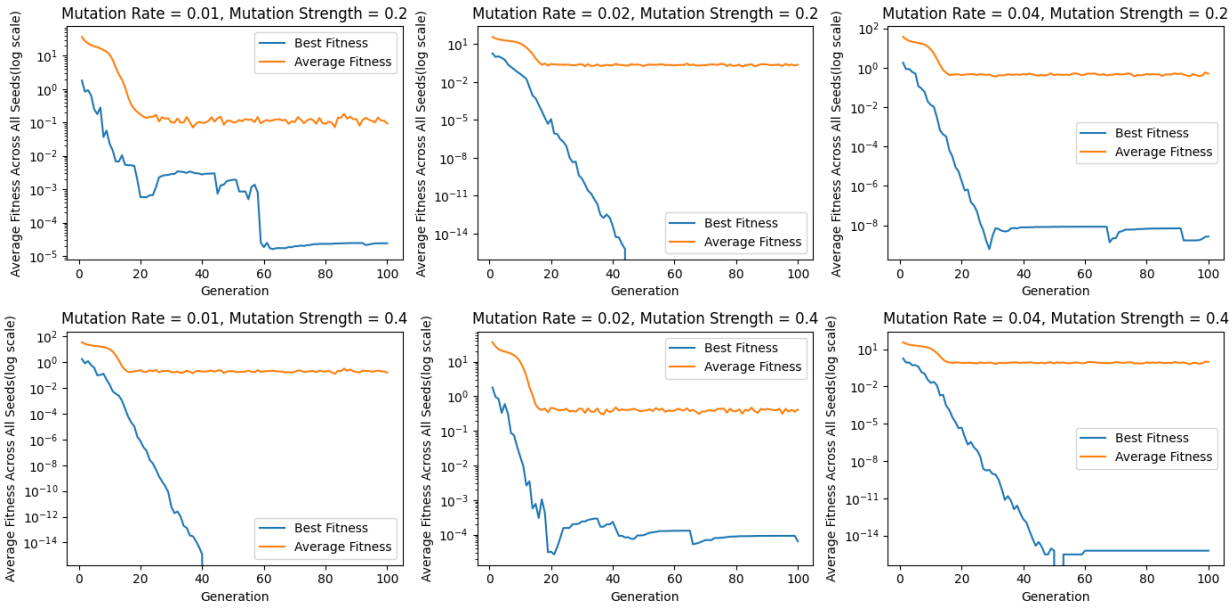

Fig.8 – Constant mutation strength on rows, constant mutation rate on columns.

Mutation Rate	Mutation Strength	Best Fitness (Final)	Average Fitness (Final)
0.01	0.2	2.4 * 10^(-5)	0.093
0.01	0.4	0	0.161
0.02	0.2	0	0.229
0.02	0.4	6.4 * 10^(-5)	0.411
0.04	0.2	2.7 * 10^(-9)	0.501
0.04	0.4	5.9 * 10^(-16)	0.963

Table 7 – Numerical values of interest for different mutation rates and strengths

In general, a higher mutation strength seems to lead to a better solution but with higher average fitness for the final population. A high mutation strength means that when mutations occur, they are more severe thus leading to better exploration of the search space. Increasing the mutation rate does not necessarily lead to better results.

We didn't use any termination criteria, because we wanted to observe the evolution of the algorithm through all 100 generations. It seems that medium values of mutation rate and mutation strength yield the best results. Also, we may expect our algorithm to cover a good portion of the search space without the risk of missing solution due to being trapped to local optima, because of sparce and light mutations or due to constantly moving around the search space because of a high frequency of severe mutations.

Conclusion

Our experiments have methodically refined a genetic algorithm for optimizing the Rastrigin function. Initial testing found a parameter set capable of reliably approaching the global optimum. Subsequent runs confirmed the solution's consistency, yet also highlighted a degree of dependency on the seed's initial value. A larger population size proved pivotal for the algorithm in avoiding local optima and maintaining a trajectory toward the global minimum. Further analysis showed that careful calibration of crossover and mutation rates is key to steady convergence and balance between exploration and exploitation. Ultimately, during our experiments we learned the critical role that randomness plays in the genetic algorithm's search efficacy, which underscores the importance of strategic parameter tuning, demonstrating that the full potential of genetic algorithms in complex optimization problems is achieved through meticulous experimentation and adjustments.