Introduction to Artificial Intelligence
Warsaw University of Technology, Summer 2025
Lab 3: Evolutionary and Genetic Algorithms

Filip Szatkowski

Task description

Write a Python program to optimize a given 2D function with a basic genetic algorithm. The optimized function and parent selection method depend on the variant of the lab:

1. Optimize Rosenbrock function using Tournament Selection:
$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2. \tag{1}$$
Initialize x and y from range $[-5, 5]$.
2. Optimize Rastrigin function using Rank Selection:
$$f(x, y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y)). \tag{2}$$
Initialize x and y from range $[-5, 5]$.
3. Optimize Bukin function using Tournament Selection:
$$f(x, y) = 100\sqrt{|y - 0.01x^2|} + 0.01|x + 10|. \tag{3}$$
Initialize x from range $[-15, 5]$ and y from range $[-3, 3]$.
4. Optimize Booth function using Roulette Wheel Selection:
$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2 \tag{4}$$
Initialize x and y from range $[-5, 5]$.
5. Optimize Styblński-Tang function using Rank Selection:
$$f(x, y) = \frac{1}{2}\left(\frac{x^4}{2} - 16x^2 + 5x + \frac{y^4}{2} - 16y^2 + 5y\right) \tag{5}$$
Initialize x and y from range $[-5, 5]$.

Algorithm details

Use the Gaussian operator for mutation and random interpolation for crossover (eg: $x_o = \alpha * x_{p1} + (1 - \alpha) * x_{p2}$, $y_o = \alpha * y_{p1} + (1 - \alpha) * y_{p2}$, where o, $p1$, $p2$ refer to offspring, parent 1 and parent 2, and α is a random number; this version of crossover operator is practically used for real-valued problems). Apply crossover and mutation to a randomly selected part of the population, and keep the other part of the population.

Hint: All of those functions have known global minima, so please check if your algorithm converges to the values close to those minima.

Instructions

The code for the functions and the template for the genetic algorithm is provided at `https://github.com/fszatkowski/EARIN_Lab3_template`. For coding the algorithm, you can use Python primitives or `numpy` package (all mathematical operations written in `numpy` should be significantly faster). Fill out the code in `genetic_algorithm.py` to obtain a working genetic algorithm. You should also add the code for plotting the results of the run (eg. the best fitness value from each generation; use a log scale for the y-axis). You can add other files and use any Python packages you like, but please add the packages to `requirements.txt`.

For the report, run and document the following experiments:

1. **Finding genetic algorithm parameters.** Run the algorithm with different parameters (population size, mutation rate and strength, crossover rate, and number of generations) until you find a set of parameters that obtains a good fitness value. Present the tested combinations in a table alongside the solutions obtained by the algorithm.

2. **Randomness in genetic algorithm.** Run the algorithm with the best parameters found in point 1. using 5 different random seeds. Report the best solution across all seeds and its fitness value, and the average fitness value and its standard deviation across all seeds. Rerun the algorithm with decreasing population size (eg. use around 50%, 25%, and 10% of the original population), and present the results in a table.

3. **Crossover impact.** Run the algorithm a few times, changing the crossover rate. Plot the best and average fitness across the generations. Report the results averaged across more than one seed.

4. **Mutation and the convergence.** Run the algorithm increasing the mutation rate and mutation strength. Plot the best and average fitness across the generations. Report the results averaged across more than one seed.

Submission guidelines

1. The report should include a short description of the task (no more than a few sentences).

2. The solution must be implemented in Python, based on the provided template.

3. Please ensure that your code adheres to basic standards of coding (PEP8). If possible, use comments and unit tests.

4. The code should be self-documenting and easily readable. Do not explain the code in detail in the report, only the details of your implementation of the algorithm.

5. The report should focus on the experiments done with the algorithm. Please document your thought process when setting the parameters, and discuss the results and your insights.

Rules

To pass the lab, it is required to submit both the code and the final report and discuss your solution during the online assessment. The online assessment will take place during the labs and should take around 15 minutes. Please notify me on Teams when you submit the solution to schedule the exact time for the meeting. You should submit the code and a PDF report to the designated Gitlab repository at least a day before the online assessment of the exercise. Programs delivered after the deadline will not be assessed. If you have any questions, please contact me via MS Teams.

Assessment Criteria

You can get $[0, 5]$ points for the lab. The following criteria will be used to evaluate your work:

- Proper implementation of the algorithm: 1 point.
- Report: 1 point.
- Online assessment: 3 points.