Exercise 2: Genetic Algorithms

Due date: *October 23rd, 2025 (push to GitHub repo before the lecture)*

General Requests

- add the required implementations to the GitHub repository of the first exercise
- add any novel relevant packages to your `uv` environment, as had (again, more information here)
- use `streamlit` again for summarization and documentation (more information here). use your `pages` folder setup (info here once more) to add a new page for the current exercise to your existing streamlit app from last exercise

2.1 Genetic Algorithms

Last exercise, we implemented a basic Hill Climbing Algorithm, which uses *one single* solution candidate and iteratively improves it by applying small random changes (mutations). In this exercise, we will implement a *population-based* optimization algorithm now - a Genetic Algorithm (GA) [1, Chapter 4][2, Chapter 2][3, Chapter 3] - to optimize coffee brewing in a dummy problem.

- use this dummy fitness function to evaluate the quality of your coffee based on 4 parameters (roast: int $[0, 20]$, blend: int $[0, 100]$, grind: $[0, 10]$, and brew_time: float $[0.0, 5.0]$). it will return a quality (fitness) value between 0 and 100. think about how to encode this problem (how would the chromosomes of each individual look like in terms of structure? take care, one of the parameters is continuous, whereas the others are integers - what encoding type could work well for this 'mixed' problem?), and how to adapt the fitness function to be able to 'digest' your chandidate chromosomes as inputs.
- implement a genetic algorithm. it should take a starting population (m individuals of dimension n), a fitness function, a termination criterion, a mutation function, and a crossover function as inputs. randomly sample two individuals as 'parents' for crossover until you generated m children (or implement one of the discussed parent selection schemes[1]), bit flip mutation, and single-point crossover to optimize your dummy 'mug of coffee' [2].
- use an adaptation of this demo visualization function to generate a contour plot of two choice parameters of the four to make everything easier to comprehend (even if it's a 4D-problem). you can of course also implement some visualization that even shows three of the four dimensions - whatever you want to use is fine.
- (optional) find an optimum solution with your Hill Climbing algorithm from last time as well. how does it compare to the genetic algorithm?

2.2 Analysis and Reporting

Add a streamlit page on Genetic Algorithms to the documentation you created for the last exercise. Your final documentation webpage should contain the following sections:

1. **Introduction** - General overview of GAs: basics, strengths/weaknesses, motivation (natural evolution), operations, etc.
2. **Methods** - Describe your implementation: encoding, specific strategies/operations, relevant parameters/functions, critical design choices, etc.
3. **Results** - Display the results obtained: best individual you found, fitness over generations (or evaluations, etc.), computation time, fitness of the best obtained individual, a visualization of how the fittest individual changes across generations (update the contour plot with the current best solution or similar), etc.
4. **Discussion** - Analyze your findings: expected versus unexpected results, solution quality, efficiency, variations with parameter choices, limitations, possible improvements, etc.

The documentation doesn't need to be absurdly long, but it should give a comprehensive overview of the topic that you can use later on to study for the exam.

[1] Very important: Keep the population count constant!

[2] Good first choices for population size and mutation rate might be population_size $\in [20, 100]$ and mutation_rate $\in [0.01, 0.1]$, but this depends on the specific problem at hand.

References

[1] A. Eiben and J. Smith, Introduction to Evolutionary Computing. Springer, 2003.

[2] X. Yu and M. Gen, Introduction to Evolutionary Algorithms. Springer, 2010. [Online]. Available: https://jainakshay781.wordpress.com/wp-content/uploads/2017/12/introduction_to_evolutionary_algorithms-184996128x.pdf

[3] D. Simon, Evolutionary Optimization Algorithms. Wiley, 2013. [Online]. Available: https://research-1ebsco-1com-1195qzf320241.perm.fh-joanneum.at/c/kofjhs/search/details/4sh2uyq6wn?db=nlebk&db=nlabk