Evolutionary Algorithms: Project assignment

Prof. Nick Vannieuwenhoven

October 20, 2025

All instances of "r0123456" in this document should be replaced with your own student number.

Read this assignment carefully before starting and closely follow its instructions.

1 Formal requirements

This document describes the project assignment for *Genetic Algorithms and Evolutionary Computing*.

- You will implement an evolutionary algorithm for the **traveling salesperson problem** in Python 3.12.3.
- The Python code that you submit will be executed and tested automatically, so it is imperative to follow the **exact** specifications and interface set out here.
- The choice of programming language is non-negotiable.
- Failure to adhere to the instructions in this assignment will result in a deduction of credit.

The project is essentially a **take-home exam**:

- The intermediate report and peer reviews comprise 10% of the final grade.
- The final report and final Python code comprise 60% of the final grade.
- The discussion of your project along with general questions about evolutionary algorithms at the exam will contribute the final 30% of your score for this course.
- If you do not hand in one of the deliverables below, the exam score will be NA.

There are **three deliverables** and likewise three deadlines to observe:

1. *November 7, 2025 at 9:00 CET*: The **intermediate report** of the group phase.
2. *November 17, 2025 at 9:00 CET*: The **peer review reports**.
3. *December 28, 2025 at 9:00 CET*: The final **Python code** and the **final report** of the individual phase.

These deliverables should be **uploaded to Toledo** by their respective deadlines. The reports must be in the **Portable Document Format**. The deliverables should be named as follows.

1. The reports must be named `r0123456_intermediate.pdf` and `r0123456_final.pdf` for the group phase report and final report respectively.
2. The peer reviews should be named `r0123456_peerHI.pdf` and `r0123456_peerAI.pdf` for the all-human and all-genAI reports, respectively.
3. The program code must be a **single Python source code file** named `r0123456.py`.

Templates of the deliverables that you must use are provided on Toledo. As the report templates will comprise a major portion of the score for this project, it is strongly advised to study the template carefully so you understand which aspects are critical in our assessment. The report templates also specify a minimum set of experiments that you must include and discuss. *The code template is discussed in section 5.*

The output of the Python code that you submit will be **graded automatically** by our testing framework. We will apply your code to five **benchmark problems**, similar to those that are posted on Toledo along with this assignment. These benchmark problems can have an **arbitrary problem size** between 100 and 1000. For each benchmark problem, your code will be given a **fixed time budget** of 5 minutes. To ensure a level playing field, your code will be **restricted to utilize** 2 **physical CPU cores**.[1] The grade you receive for the code will be partly based on the performance of your best solution relative to your peers.

The goal of this project is that you design, implement, and analyze an evolutionary algorithm yourself, hence **you cannot use existing Python evolutionary algorithms frameworks**. You are allowed to use any other package installable from `pip` that is not designed for (approximately) solving combinatorial optimization problems. You can use **numba**. When in doubt if your package is allowed, contact your teaching assistant.

[1] Hyperthreading and logical processors are disabled on the benchmark machine, so the total number of threads that can execute concurrently is 2.

2 Generative AI reporting requirements

General information about the use of genAI can be found under Practical Information: GenAI usage on Toledo.

If generative AI is used

- to generate ideas, or
- to generate any computer code relevant to solving the project,

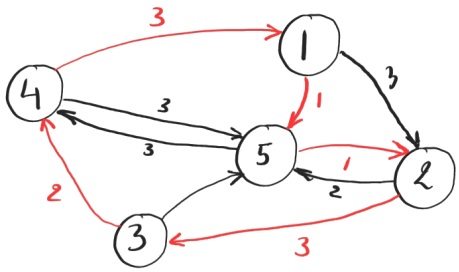
then the relevant sections of the final report template must be completed. In particular, you must:

- disclose the generative AI system used (e.g., ChatGPT, Copilot, Midjourney, etc);
- describe the intention and reason for employing generative AI for each specific instance you used it for;
- document all of the prompts used to generate the content;
- describe which postprocessing you applied to the generated content;
- critically reflect on the output generated and describe what you learned and in which aspects the result was useful and in what aspects it was not useful.

Failure to include this information will, in the phrasing of article 84 of the KU Leuven exam regulations, inhibit my "judgement of [your] own knowledge, understanding and/or skills."

3 Problem statement

The **traveling salesperson problem** consists of minimizing the length of a cycle that visits all vertices in a weighted, directed graph exactly once. The length of a cycle is defined as the sum of the weights of all directed edges constituting the cycle. For example, the cycle defined by the red edges in the figure below has length 10:



Note that this picture includes vertices that are not directly connected by one another, such as 1 and 3. This is modeled as an edge weight equal to ∞.

Formally, let (V, E) be a weighted, directed graph with n vertices V and directed edges $E \subset V \times V$. An edge from $v_i \in V$ to $v_j \in V$ is denoted by $e_{i,j} = (v_i, v_j)$. The weights, or distances, of the edges are given by a **nonsymmetric distance matrix** $D \in \mathbb{R}^{n \times n}$: $d_{i,j}$ represents the weight of $e_{i,j} = (v_i, v_j)$. Note that $d_{i,j}$ can be different from $d_{j,i}$. The length of the cycle (or cyclic permutation) $\pi = (\pi_1 \pi_2 \pi_3 \cdots \pi_n)$ is then defined as

$$\text{length}(\pi) := d_{\pi_n, \pi_1} + \sum_{i=1}^{n-1} d_{\pi_i, \pi_{i+1}}.$$

The goal of the traveling salesperson problem is to solve the optimization problem

$$\min_{\pi \in C_n(V)} \text{length}(\pi),$$

where $C_n(V)$ is the set of all $(n-1)!$ cyclic permutations on the n vertices V. The objective function that we will use to evaluate your project is the length of the cycle.

4 Work plan

The project consists of two phases: a **group phase** in groups of 3 (or 4) that takes place during four exercise sessions, and a more substantial **individual phase** that you complete individually.

Collaboration is allowed only during the group phase and only with your group members. The goal of the group phase is to collaborate in diverse teams, to optimize your understanding of genAI capabilities, and create a minimal working evolutionary algorithm utilizing only the basic components (initialization, selection, variation, elimination) that you can extend, improve, and optimize in your individual phase.

4.1 Group phase (10h)

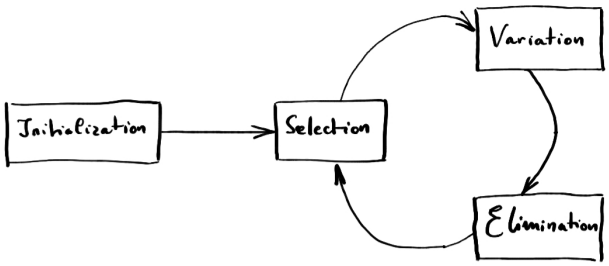
During the group phase you work in your group to design and implement a basic evolutionary algorithm. The composition of the groups will be communicated in the first exercise session.

In some of the exercise sessions you can use generative AI. You must use Copilot, which you can access from `https://copilot.cloud.microsoft/`. **You must log in under your KU Leuven account**, so that information you enter will be treated confidentially and is not used for further training of the model. Only under these conditions, you may upload the assignment and templates to Copilot.

4.1.1 Exercise session 1 (human)

You cannot use generative AI during this exercise session.

Think about all the components that you need for a basic evolutionary algorithm:



Discuss and decide which representation to use, what selection and elimination to perform, and what variation operators could be designed. Try to reason how these four components would interact with one another. You may want to look up more information about the problem online, using non-genAI solutions.

Look at the intermediate report and peer review templates to see which information you will need to record.

4.1.2 Exercise session 2 (genAI)

You must use Copilot generative AI during this exercise session.

Make (a) prompt(s) to let Copilot generate a representation, the objective function `length`, a simple population initialization, a selection mechanism, one mutation operator, one recombination operator, an elimination mechanism, and a stopping criterion that corresponds to the design that you developed in the first exercise session. Make sure to let Copilot **conform to the code template** that is provided on Toledo and further described in section 5. Test your basic evolutionary algorithm on the smallest benchmark problem on Toledo. Let Copilot fix any programming errors it may have generated.

Record the prompts that you used to generate your solution; they will need to be reported.

4.1.3 Exercise session 3 (hybrid)

You may use Copilot generative AI during this exercise session.

Write the group phase project report using the template, performing the numerical experiments as required. You can use Copilot as you see fit.

Each group member must hand in this report on Toledo by November 7, 2025 at 9:00 CET. No code must be submitted.

4.1.4 Exercise session 4 (hybrid)

You must use Copilot generative AI during this exercise session.

You will receive the intermediate reports from two groups at the start of the exercise session. One is marked for human review and one is marked for genAI review.

First, read the report marked for human review and discuss the evolutionary algorithm presented therein with your group. What are the strong and weak points? Do you think the components would interact well with one another? Can you think of a suggested improvement? Write down your conclusions in a **peer review report using the template on Toledo**.

Second, generate the genAI review report by uploading the other group's report along with the peer review report template (as .txt file) and ask Copilot to generate and complete the review template. Download the solution and compile this LaTeX document. This is the genAI "peer" review report.

Each group member should hand in both peer reviews on Toledo by November 17, 2025 at 9:00 CET. In the following week, you will receive two peer reviews about your own group report. You can use the insights from these peer reviews to improve your evolutionary algorithm in the individual phase. In the final report, you will be asked to reflect on the relative merits of the human and genAI peer reviews that you received.

4.2 Individual phase (40h)

In the individual phase you continue from the code that you prepared in the group phase with your group. From this point onwards, the project is an individual assignment. **Cooperation is forbidden in this phase.** Generative AI can be used as explained on Toledo and subject to the reporting requirements discussed in section 2.

The goal of this phase is that you develop an advanced, optimized evolutionary algorithm for the traveling salesperson problem. You could implement additional or more advanced initialization heuristics, selection schemes, elimination mechanisms, variation operators, local search operators, population enrichment schemes, parameter self-adaptivity, optimized hyperparameter search, diversity promotion, and so on. You should also write an individual final report using the final report template.

The Python code and final report should be turned in via Toledo by December 28, 2025 at 9:00 CET.

5 Code template

The code template from which you start is as follows.

```
import Reporter
import numpy as np

# Modify the class name to match your student number.
class r0123456:

  def __init__(self):
    self.reporter = Reporter.Reporter(self.__class__.__name__)

  # The evolutionary algorithm's main loop
  def optimize(self, filename):
    # Read distance matrix from file.
    file = open(filename)
    distanceMatrix = np.loadtxt(file, delimiter=",")
    file.close()

    # Your code here.
    yourConvergenceTestsHere = True
    while( yourConvergenceTestsHere ):
      # Your code here.
      meanObjective = 0.0
      bestObjective = 0.0
      bestSolution = np.array([0,1,2,3,4,5,6])



      # Call the reporter with:
      #  - the mean objective function value of the population
      #  - the best objective function value of the population
      #  - a 1D numpy array in the cycle notation containing the best solution
      #    with city numbering starting from 0
      # Leave the next three lines as they are.
      timeLeft = self.reporter.report(meanObjective, bestObjective, bestSolution)
      if timeLeft < 0:
        break

    # Your code here.
    return 0


if __name__ == '__main__':
  # Write any testing code here.
  return 0
```

You should **replace the class name** with your student number. You can add additional code to the constructor.

The class should have at least one method, `optimize`. The testing framework will call `optimize` with one argument, a string containing the filename of the traveling salesperson problem instance to optimize. You are allowed to add optional arguments to this method. The main loop of your evolutionary algorithm should be implemented by this method. The first three lines of this method should read the matrix, as in the template. The last three lines of the main evolutionary algorithm loop should report your results using the Reporter class provided on Toledo as in the template. **Do not modify the reporter.** The `float` returned by `Reporter.report` method indicates the amount of time in seconds that is left. These last three lines of the while loop must not be modified (it is allowed to increase 0 to a different positive number in the if-test). The stopping criterion `yourConvergenceTestsHere` can be implemented as you see fit.

Everything else you can implement as you desire (extra classes, methods, attributes, etc), subject to the constraint that you can use only 1 module (code file).

The testing framework will call your code as in the following example:

```
import r0123456

filename = "./tour50.csv"
a = r0123456.r0123456()
a.optimize(filename)
```

Your code will be executed in Python 3.12.3 on two physical CPU cores of a machine containing an Intel Core i9-10885H CPU (2.4 GHz base, boost up to 5.3GHz) and 128GB of main memory.