Assignment 1

COMP 9058 - Metaheuristic Optimisation

September 29, 2025

The assignment should be submitted on Canvas by **Sunday, October 19th, 23:59**. As per MTU regulations, submitting within 7 days of the deadline will result in a 10% penalty, between 7 and 14 days late will result in a 20% penalty, and later than 14 days will result in a 100% penalty applied.
The pdf MUST be named *Lastname_Firstname_StudentNumber_MH1.pdf.* Similarly your python script must be named
Lastname_Firstname_StudentNumber_MH1.py.
You need to write a report with a description of the experiments and an evaluation (and analysis) of the results. Use tables and figures where appropriate. As always, the random number generator must be seeded to your student number for your experiments (by assigning the variable *myStudentNum* in the TSP_student.py file). The main goal of this assignment is to assess your understanding of the content that has been discussed in the lectures.

1 Assignment Description

1.1 Genetic Algorithms - [80 Marks]

The Traveling Salesman Problem (TSP) is one which has commanded much attention in Artificial Intelligence because it is so easy to describe and so difficult to solve. The problem can simply be stated as: if a traveling salesman wishes to visit exactly once each of a list m cities (where the cost of traveling from city i to city j is c_{ij}) and then return to the home city, what is the least costly route the traveling salesman can take.

Simple GA

A simple genetic algorithm can be defined in the following 9 steps:

1. Create an initial population of P chromosomes (generation 0)
2. Evaluate the fitness of each chromosome
3. Select P parents from the current population via fitness-biased selection for the mating pool (i.e., the selection probability is dependent on the fitness).
4. Combine two randomly chosen parents from the mating pool using a crossover operation to create offspring
5. Process each offspring by the mutation operation, and insert the resulting offspring in the new population
6. Repeat steps 4 and 5 until P offspring are created
7. Replace the old population of chromosomes with the new population, keeping the elites.
8. Evaluate the fitness of each chromosome in the new population, updating current best.
9. Go back to step 3 if the number of generations is less than some upper bound. Otherwise, the final result is the best chromosome created during the search.

This assignment requires you to adapt a program that solves the TSP using Genetic Algorithms, you are given skeleton code containing the majority of the necessary functionality. Note the skeleton GA for TSP code provided is somewhat basic so as to allow understanding. Efficiency can be improved so consider the efficiency measures discussed in class.
You are given code implementing a GA for the TSP (TSP_student.py and TSP_Individual.py) with the following aspects:

1.1.1 Population initialisation

Random tour generation.

1.1.2 Selection

Truncation selection for selecting parents for crossover.

1.1.3 Crossover and Mutation

Crossover should be performed with crossover probability (P_C) (and if not performing crossover, the parents just become the children and get passed to mutation), and mutation should be performed with probability (P_M).
The default crossover operator to be used is the (permutation) OX Crossover which is implemented, the default mutation operator is the reciprocal exchange mutation operator which is implemented.
Implementation: You must implement the uniform crossover operator (function *uniformCrossover*) and the inversion mutation operator (function *inversionMutation*)

1.1.4 Replacement

Elitism with a parameter indicating the proportion of the population carried over, where 0 would mean no elite solutions and 1 would mean all the previous population are elite solutions.

I/O Specification

The following problem instances will be used to evaluate the performance of your algorithms:

- if the last digit of your id is < 3: inst-a.tsp, and inst-b.tsp
- if the last digit of your id is in the range 3-5 inclusive: inst-c.tsp, and inst-d.tsp
- if the last digit of your id is > 6: inst-e.tsp, and inst-f.tsp

1.2 NP-Completeness - [20 Marks]

Note this part is not a coding exercise nor one that requires solving the 3SAT instance. Simply present the steps used for converting the clauses and the final formula for the SAT to 3SAT reduction.
This problem concerns the proof of the NP-completeness of 3SAT.
Convert the formula F below into a 3SAT formula F', find a solution to F' and verify that this is a solution to F

1. If the last digit of your student id is less than 3 use
   $F = (q_1 \vee \neg q_2 \vee \neg q_3 \vee q_4 \vee \neg q_5) \wedge (q_1 \vee \neg q_3) \wedge (q_5)$

2. If the last digit of your student id is either 3, 4 or 5 use
   $F = (p_1 \vee \neg p_2 \vee \neg p_3 \vee p_4 \vee p_5) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_1)$

3. If the last digit of your student id is either 6 or 7 use
   $F = (\neg w_1 \vee w_2 \vee w_3 \vee \neg w_4 \vee w_5) \wedge (\neg w_1 \vee w_3) \wedge (\neg w_3)$

4. If the last digit of your student id is greater than 7 use
   $F = (z_1 \vee \neg z_2 \vee z_3 \vee z_4 \vee \neg z_5) \wedge (\neg z_1 \vee \neg z_3) \wedge (z_4)$

2 Submission, marking and academic integrity

The deliverable of this project will consists of a python code file(s) and a report. You must follow the correct scheme with last digit of id and first letter of first and last names, **you must state what you used** to avoid confusion. Marks will be lost if this is not done correctly, similarly the **random seed must be set and must be your student id.**

2.1 Submission:

This assignment is due by 23:59 on Sunday, Oct 19th, 2025. You must submit the pdf via the Turnitin assignment submission, and the pdf should include your work for Parts 1 and 2 of the assignment (i.e. everything except the assignment 1 implementation), and the python file separately via the code submission Your pdf **MUST** be named Lastname_Firstname_StudentNumber_MH1.pdf (e.g. Grimes_Diarmuid_R1234_MH1.pdf). Similarly your python file MUST be named *Lastname_Firstname_StudentNumber_MH1.py.*

2.2 Rubrics:

Part 1: GA

Implementation

Pay careful attention to the instructions, your implementation must be based on the exact specification in this document, including naming conventions, and command line format of arguments. Again, what needs to be implemented is

- Change *myStudentNum* variable so it is set to your student number (removing leading R000 part, so if student number is R00123, then you would set the variable to 123).
- Uniform crossover operator.
- Inversion mutation operator.
- Add metrics.

The main file must follow the same naming convention (Lastname_Firstname_StudentNumber_MH1.py) and be created such that it can be run from the command line with parameters passed in for each parameter as follows:

```
python Grimes_Diarmuid_R012345_MH1.py inst_file nRuns nIters popSize xoverH  Pc mutH Pm  elites trunk
```

where

- *inst_file* is the instance file to run on (including path to file).
- *nRuns* is the number of runs of the GA to be performed on the instance (you must run at least 10), default 10.
- *nIters* is the number of generations to be produced in a run of the GA, default 1000.
- *popSize* is the number of chromosomes in the population default 100.
- *xoverH* is the crossover operator (0 for ox crossover, 1 for uniform), default 0.
- *Pc* is the crossover probability, default 0.9.

- *mutH* is the mutation operator (0 for reciprocal exchange, 1 for inversion), default 0.
- *Pm* is the mutation probability (note here it is the probability of performing the mutation operator *per chromosome*, not per gene), default 0.2.
- *elites* is the proportion of the population that are elite (value between 0 and 1), default 0.1.
- *trunk* is the proportion to be used for truncation selection (similarly value between 0 and 1), default 0.5.

The algorithm is logically well designed and efficient without inappropriate design choices (e.g., unnecessary computations per iteration). Code is well commented.	The algorithm always works properly and meets the specification. Code is clean, understandable and well organized.	The algorithm works properly in limited cases	The algorithm is incorrectly implemented
(22-30 Marks)	(13-21 Marks)	(5-12 Marks)	(0-5 Marks)

Advice: *Use small toy instance as discussed in the first lab, for testing and debugging.*

Part 2: NP-Completeness

Rubric (Solution and verification aspect only refers to SAT to 3SAT):

Reduction is logically well designed, documented and explained. Solution is correct.	Reduction has slight logic errors that do no significantly affect the results. Solution has slight errors	Reduction has significant logic errors. Solution has errors.	Reduction is completely incorrect. Solution completely incorrect.
(16-20 Marks)	(11-15 Marks)	(6-10 Marks)	(0-5 Marks)

Evaluation

Your evaluation should involve, and discuss, the following experiments (run with default values unless otherwise stated):

- Impact of crossover operator.

- Impact of mutation operator.
- Impact of crossover probability (test 2/3 different values including the default of 0.9).
- Impact of mutation probability (test 2/3 different values including the default of 0.2).
- Impact of elitism (test the default elitism of 0.1 and no elitism).
- Impact of population size and number of generations (test 2/3 different values including the default of 100 for population and 1000 for generations, varying iterations at the same time to keep total chromosomes generated constant).

The **default settings** are given as follows (using the command line arguments listed above in that order):

```
python  Grimes_Diarmuid_R001234567_MH1.py inst_file  10 1000 100  0 0.9  0 0.2 0.1 0.5
```

This will perform 10 runs, each with the following GA settings

- 1000 generations
- population size 100
- ox crossover
- P_c 0.9
- reciprocal exchange mutation
- P_m 0.2
- Elitism 0.1
- Truncation 0.5

You should evaluate in terms of both solution quality and runtime, including the best and average for solution quality across runs.
You should also analyse the evolution of the solution quality across generations for a sample run of the default settings.

Excellent presentation, depth and insight analysis of the empirical results.	Good presentation of the results (e.g., describing the results with well structured tables).	Incomplete and/or unclear presentation of the results.	The results are poorly presented and interpreted, does not demonstrate understand-ing.
(41-50 Marks)	(26-40 Marks)	(11-25 Marks)	(0-10 Marks)

2.3 Academic Integrity

This is an **individual assignment**. The work you submit must be your own. In no way, shape or form should you submit work as if it were your own when some or all of it is not, including ChatGPT which you must explicitly state if used and where, bearing in mind that the purpose of this assignment is to assess your understanding of the course content. Any online source that is used must be cited, and a full citation given, e.g. do NOT give "stackoverflow", but the full citation, e.g. https://meta.stackoverflow.com/questions/339152/plagiarism-and-using-copyin -and-submitting-it-in-an-as. If you are unsure on whether something should be cited, general rule of thumb is to err on the side of caution and include the citation. You can also ask me via email
Collusion: Given how much freedom there is in the assignment, everybody's work will be different. It will be obvious if there is collusion. ***All parties to collusion will be penalized.***
Deliberate plagiarism: You must not plagiarise the programs, results, writings or other efforts of another student or any other third-party. Plagiarism will meet with severe penalties, which can include exclusion from the Institute.
Your report and code will be checked for signs of collusion, plagiarism, falsification and fabrication. You may be called to discuss your submission and implementation with me and this will inform the grading, any penalties and any disciplinary actions.