Habib University
CS 451 - Computational Intelligence
Spring' 2026
Assignment 1 – Evolutionary Algorithm

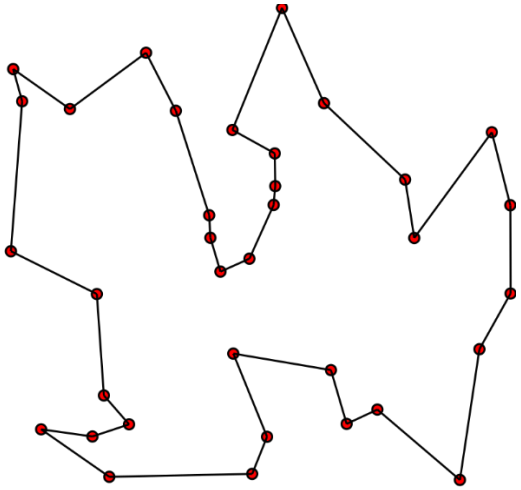
Objective:

The purpose of this assignment is to provide students an insight of global optimization using Evolutionary Algorithms (EA). This exercise will enable them to address some popular computing problems that can be mapped to several real-world problems and are known to be computationally hard. The process will expose them to the overall process of EA, its underlying challenges and the impact of different parameters and selection schemes.

In this assignment, you will implement the standard process of EA to address the following two benchmark problems. You will also plot graphs to show EA progress and its convergence with a concluding analysis. Specific details of the EA process are given in Appendix A.

1. Travelling Salesman Problem (TSP):

TSP asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"



There are multiple TSP problem instances available online. You will use the dataset of **Qatar** that contains **194 cities**. The optimal tour reported so far for this dataset is of length ***9352*** (and the ***best that any CI student has achieved in past iterations is 9573***). The dataset and its related details can be found at: http://www.math.uwaterloo.ca/tsp/world/countries.html (look for Qatar dataset on this page).

2. Helping the RO: An Exam Scheduling Challenge

The Registrar's Office at Habib University currently uses UniTime for class and exam timetabling. Despite this, many students remain dissatisfied with their schedules—due to clashes, uneven exam loads, inconvenient timings, and other constraints and preferences.

Now, the ball is in *your* court.

Your task is to design an **exam timetabling solution using evolutionary algorithms** that satisfies as many hard constraints as possible while maximizing adherence to soft constraints and student and faculty preferences. The goal is not merely to produce a feasible timetable, but to generate one that is as optimal and student-friendly as possible.

Before the Registrar's Office can trust you with real institutional data, you must first demonstrate your computational optimization skills on a well-established benchmark problem. For this assignment, you will work with a **publicly available exam timetabling dataset provided by Purdue University**.

Multiple benchmark instances and a detailed problem description are available in the Unitime Exam Timetabling benchmark datasets. For this assignment, you will use the **"Spring 2012"** data instance with the **"production"** configuration**.**

Your solution should clearly define:

- the representation of a timetable (chromosome design)
- the fitness function, including how constraints and preferences are modeled
- the evolutionary operators used (selection, crossover, mutation)
- any constraint-handling or repair mechanisms
- experimental results and analysis of solution quality

Grading:

The grading will be based on the following components:

Problem Formulation (for both problems) Including Chromosome representation, Fitness function, and Crossover and Mutation operators	20% (5 + 15)
EA Implementation (Overall EA Process, Selection Schemes) (Correctness of code, Proper Structuring of code, Generic Implementation/Code reusability)	30%
Fine-tuning of parameters and final solution	15 %
Gathering statistics and plotting graph	15 %
Analysis and Findings	20 %

Submission:

The assignment will be done in pairs. Please sign up for your Assignment 1 team on Canvas. You will submit your code and the pdf file describing (for both problems):

- Complete Problem Formulation
- Plots of Avg. BSF and Avg. ASF for various combinations of schemes, The two plots for each combination should be shown side by side for comparison. See Appendix A for details on graph plotting.
- Finally, you will present your overall analysis of various settings and the configuration that worked best for you.

Appendix – A

EA Implementation:

Your EA will perform the following cycle:

- Step 0: Analyse the problem that you are addressing and decide your chromosome representation and fitness function.
- Step 1: Initialize the population randomly or with potentially good ***solutions***.
- Step 2: Compute the ***fitness*** of each individual in the population.
- Step 3: Select parents using a ***selection procedure***.
- Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
- Step 5: Compute the ***fitness*** of the new offspring.
- Step 6: Select members of population to die using a ***selection procedure***.
- Step 7: Go to Step 2 until the termination criteria are met.

The following selection schemes will be implemented:

- Fitness Proportional Selection
- Rank based Selection
- Binary Tournament
- Truncation
- Random

Following parameters will be configurable in your program with some default values given below that you are free to change and observe the behavior of your algorithm:

- Population size: 30
- Number of offspring to be produced in each generation :10
- No. of generations: 50
- Mutation rate: 0.5
- No of Iterations: 10

EA Evaluation:

In each generation, you will note the average fitness (avg) of the generation and the best fitness so far (BSF). Suppose you run the process for 40 generations. You will have following observations:

Generation #	Best Fitness so far (BSF)	Average Fitness so far (ASF)
1		
2		
..		
..		

..		
40		

You need to repeat this exercise (for a single combination of parent and survival selection schemes) K times where K is your 'number of iterations'. Each run will start with a new randomly initialized population.

You will have the following table after K runs:

Generation #	Run # 1 BSF	Run # 2 BSF	..	..	Run # 10 BSF	Average BSF
1						
2						
..						
..						
..						
40						

A similar table will be obtained for average 'average fitness'.

- Once you have calculated (a) average best-so-far values and b) average average-fitness, you need to plot these values against generation # and analyze their pattern.
- You will repeat the process for different combinations of parent and survival selections such as (not limited to):
  - FPS and Truncation
  - Binary Tournament and Truncation
  - Truncation and Truncation
  - Random and Random
  - RBS and Random

Finally, you will provide your analysis on the behavior of EA under different parameters and selection schemes. Which scheme did work best in your case and which one performed worst?