INTRODUCTION TO ARTIFICIAL INTELLIGENCE
Lab 4 – Evolutionary algorithms

This script briefly delineates **Evolutionary Algorithms** for optimization. Before doing this, let's consider a representative combinatorial optimization problem – **Travelling Salesman Problem (TSP).**

1) Travelling Salesman Problem (TSP): Given are M cities. The distances between each pair of cities are known. The objective is to traverse through each city once and return to the starting point. In other words, the goal is to find a graph cycle that goes through each node. Consider the example below. There are 5 cities: A, B, C, D, E, and a distance matrix. A solution is here represented as a **permutation of city indices**. The given sequence of indices shows how the nodes (cities) are visited in the graph. Using the distance matrix, we can compute the total distance. It is an objective function that has to be minimized in the TSP problem. In other words, one has to find a permutation of indices that gives the minimum total distance. The complexity of this problem is M! (as there are precisely M! permutations).

Distance matrix:

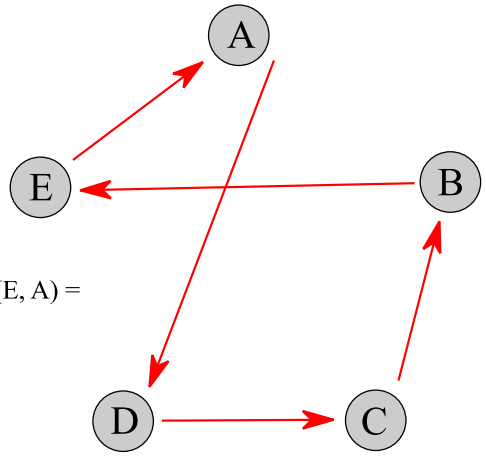
	A	B	C	D	E
A	0	2	4	2	5
B	-	0	3	1	3
C	-	-	0	4	1
D	-	-	-	0	3
E	-	-	-	-	0

Example solution in graph representation

note that the distances between the nodes do not necessarily reflect the actual distances between the cities

Solution = [A, D, C, B, E]

Distance = d(A, D) + d(D, C) + d(C, B) + d(B, E) + d(E, A) =
= 2 + 4 + 3 + 3 + 5 = 17



2) Evolutionary algorithms: Evolutionary algorithms mimic natural evolution to solve optimization problems. They include such elements as, e.g., population, mating pool, reproduction operators, or the survival of the fittest rule. These algorithms are heuristics, which means that they include some randomness in the search process. They do not guarantee to construct an optimal solution as ensured by exact methods for optimization. Nonetheless, exact methods are often inefficient when the problem complexity is large. In turn, heuristics usually can often generate solutions of good quality at a reasonable time - and this is why they are given a lot of interest in a scientific community. In this script**, a basic, simple generational evolutionary algorithm is considered**. It consists of the following steps:

A. Construct an initial population
B. Evaluate solutions in the population
C. Sort the population
D. Construct a mating pool and select parent solutions
E. Generate the offspring
F. Evaluate the offspring
G. Merge the population with the offspring
H. Sort the merged population
I. Remove the worst-performing solutions.
J. Go to 4 unless stopping criteria are satisfied.

In what follows, the A-J steps are briefly delineated. The discussion is based on the TSP problem.

A. CONSTRUCT AN INITIAL POPULATION OF SIZE N

Firstly, we need to construct an initial population of size N, i.e., generate N solutions. This step can involve some problem-specific procedures, but also solutions can be generated randomly. In the TSP optimization problem, a solution can be a random permutation of 1, 2,..., M indices.

Example population for M = 5 (number of cities) and N = 5 (population size):

$$\boldsymbol{P} = \{\boldsymbol{S_{11}} = [\mathbf{3,5,4,1,2}],\ \boldsymbol{S_{12}} = [\mathbf{5,2,3,1,4}], \boldsymbol{S_{13}} = [\mathbf{3,1,4,2,5}],\ \boldsymbol{S_{14}} = [\mathbf{4,1,2,5,3}], \boldsymbol{S_{15}} = [\mathbf{1,2,3,4,5}]\}$$

B. EVALUATE SOLUTIONS IN THE POPULATION – ASSIGN FITNESS

Solutions in the population P have to be evaluated. We can calculate total distances for paths imposed by solutions with a distance matrix D.

Example evaluations for the above data and some example distance matrix D:

Distance matrix D:

D	1	2	3	4	5
1	0	3	1	5	3
2	-	0	2	1	2
3	-	-	0	3	1
4	-	-	-	0	5
5	-	-	-	-	0

Solution evaluations:

ID	Solution	Evaluation
S_{11}	$[3,5,4,1,2]$	1 + 5 + 5 + 3 + 2 = 16
S_{12}	$[5,2,3,1,4]$	2 + 2 + 1 + 5 + 5 = 15
S_{13}	$[3,1,4,2,5]$	1 + 5 + 1 + 2 + 1 = 10
S_{14}	$[4,1,2,5,3]$	5 + 3 + 2 + 1 + 3 = 14
S_{15}	$[1,2,3,4,5]$	3 + 2 + 3 + 5 + 3 = 16

C. SORT THE POPULATION ACCORDING TO FITNESS:

We have to sort solutions in the population according to their fitness values. Solutions having better fitness, i.e., lower total distances, have to be ranked better. Having a sorted population will allow performing other evolutionary steps more efficiently.

Sorted population:

Rank	ID	Solution	Evaluation
1	S_{13}	$[3,1,4,2,5]$	1 + 5 + 1 + 2 + 1 = 10
2	S_{14}	$[4,1,2,5,3]$	5 + 3 + 2 + 1 + 3 = 14
3	S_{12}	$[5,2,3,1,4]$	2 + 2 + 1 + 5 + 5 = 15
4	S_{15}	$[1,2,3,4,5]$	3 + 2 + 3 + 5 + 3 = 16
5	S_{11}	$[3,5,4,1,2]$	1 + 5 + 5 + 3 + 2 = 16

D. CREATE A MATING POOL AND SELECT PARENT SOLUTIONS

Now, we can create a mating pool and select parent solutions. A mating pool is a subset of solutions from the population that potentially can be selected as parent solutions for the reproduction step. Naively, we can assume that the whole population is a mating pool. However, it is usually more beneficial to restrict the mating pool. Then, having a mating pool, we can select solutions for reproduction. There are different schemes for this process. In this simple example, let's assume that:

1. We generate as many offspring solutions as many solutions there are in the population.
2. We select two parent solutions from the population P to generate one offspring.

Hence, we have to perform N selections, each time selecting two solutions. Solutions having better fitness should have higher chances of being chosen because of the assumption that it is more probable that good offspring will be generated from goo parents than vice-versa. The two most population selection procedures are:

1. Proportional selection: the selection probability is proportional to the solution's fitness.

2. Tournament selection of size K: We randomly pick K indices and select the one having a better rank. If K = 1, the solutions are chosen randomly with a uniform distribution. If K = N and the selection is without replacement, then always the population's best solution will be selected.

Population P

Rank	ID	Solution	Evaluation
1	S_{13}	$[3,1,4,2,5]$	10
2	S_{14}	$[4,1,2,5,3]$	14
3	S_{12}	$[5,2,3,1,4]$	15
4	S_{15}	$[1,2,3,4,5]$	16
5	S_{11}	$[3,5,4,1,2]$	16

An example selections (N pairs of Parent solutions)

ID	Parents
1	$[S_{13}, S_{12}]$
2	$[S_{14}, S_{12}]$
3	$[S_{13}, S_{15}]$
4	$[S_{13}, S_{14}]$
5	$[S_{13}, S_{11}]$

The number of times of being selected:

Rank	ID	Selected
1	S_{13}	4
2	S_{14}	2
3	S_{12}	2
4	S_{15}	1
5	S_{11}	1

E. CREATE AN OFFSPRING POPULATION O

The algorithm can construct offspring having the selected pairs of parents. For this reason, algorithms often apply two types of reproduction operators:

1. **Crossover operator (exploitation)**: Firstly, the method "merges" parent solutions to generate an offspring. The representative operator here is a single-point crossover. Firstly, it randomly picks an index, i.e., point. Then, the offspring takes the left-side data as imposed by the selected point of the parent genotype. The remaining values are taken from the right-side data of the second parent genotype. Note that in TSP standard solution representation, it is assumed that city indices are unique. Hence, such crossover may generate invalid solutions - as some duplicates may appear - and, thus, some repairing procedure has to be applied.

An example application of the single-point crossover

|| - indicates the point of cut

duplicates are marked with red color.

1st parent	2nd parent	Offspring	ID	After repairing
$[3,1,4,2,5]$	$[5,2,3,1,4]$	$[3,1,\|\|3,1,4]$		$[2,1,\|\|3,5,4]$
$[4,1,2,5,3]$	$[5,2,3,1,4]$	$[4,1,2,\|\|1,4]$		$[3,1,2,\|\|5,4]$
$[3,1,4,2,5]$	$[1,2,3,4,5]$	$[3,\|\|2,3,4,5]$		$[3,\|\|2,1,4,5]$
$[3,1,4,2,5]$	$[4,1,2,5,3]$	$[3,1,4,\|\|5,3]$		$[2,1,4,\|\|5,3]$
$[3,1,4,2,5]$	$[3,5,4,1,2]$	$[3,1,\|\|4,1,2]$		$[3,5,\|\|4,1,2]$

2. Mutation operator (exploration): Secondly, the method usually applies some mutation operator. This operator is used to alter the solutions slightly, which helps to escape the problem's local optima. In the context of solutions represented as indices' permutations, a representative mutation operator is a random swap. It replaces two indices in the vector.

An example application of the random swap mutation

swapped solutions are marked with red color.

ID	Crossed-over solutions	Mutated solutions
S_{21}	$[2,1,3,5,4]$	$[2,5,3,1,4]$
S_{22}	$[3,1,2,5,4]$	$[1,3,2,5,4]$
S_{23}	$[3,2,1,4,5]$	$[3,5,1,4,2]$
S_{24}	$[2,1,4,5,3]$	$[2,1,5,4,3]$
S_{25}	$[3,5,4,1,2]$	$[2,5,4,1,3]$

F. EVALUATE THE OFFSPRING

Now, we have to evaluate the offspring:

Offspring evaluations:

ID	Solution	Evaluation
S_{21}	$[2,5,3,1,4]$	2 + 1 + 1 + 5 + 1 = 10
S_{22}	$[1,3,2,5,4]$	1 + 2 + 2 + 5 + 5 = 15
S_{23}	$[3,5,1,4,2]$	1 + 3 + 5 + 1 + 2 = 12
S_{24}	$[2,1,5,4,3]$	3 + 3 + 5 + 3 + 2 = 16
S_{25}	$[2,5,4,1,3]$	2 + 5 + 5 + 1 + 2 = 15

G. MERGE WITH OFFSPRING & SORT

Next, we have to merge the original population P with the offspring P. Following, we can sort thus combined population.

Merged and sorted population:

Rank	ID	Solution	Evaluation
1	S_{13}	$[3,1,4,2,5]$	10
2	S_{21}	$[2,5,3,1,4]$	10
3	S_{23}	$[3,5,1,4,2]$	12
4	S_{14}	$[4,1,2,5,3]$	14
5	S_{12}	$[5,2,3,1,4]$	15
6	S_{22}	$[1,3,2,5,4]$	15
7	S_{25}	$[2,5,4,1,3]$	15
8	S_{15}	$[1,2,3,4,5]$	16
9	S_{11}	$[3,5,4,1,2]$	16
10	S_{24}	$[2,1,5,4,3]$	16

H. APPLY THE "SURVIVAL OF THE FITTEST RULE"

To conclude, we can remove the worst-performing solutions while preserving the best ones. To keep the population size constant throughout the evolutionary run, we can remove the population's worst-half. The remaining, better performing best-half will be passed to the next iteration, called a generation.

Example application of the "survival of the fittest rule"

Solutions marked with green are to be passed to the next generation. Notably, two solutions in this set came from the offspring. Hence, the algorithm reported an improvement in this generation.

Rank	ID	Solution	Evaluation
1	S_{13}	$[3,1,4,2,5]$	10
2	S_{21}	$[2,5,3,1,4]$	10
3	S_{23}	$[3,5,1,4,2]$	12
4	S_{14}	$[4,1,2,5,3]$	14
5	S_{12}	$[5,2,3,1,4]$	15
6	S_{22}	$[1,3,2,5,4]$	15
7	S_{25}	$[2,5,4,1,3]$	15
8	S_{15}	$[1,2,3,4,5]$	16
9	S_{11}	$[3,5,4,1,2]$	16
10	S_{24}	$[2,1,5,4,3]$	16