CS 451 : Computational Intelligence

Assignment 1

Evolutionary Algorithms

Mustafa Sohail | ms06860@st.habib.edu.pk
Muhammad Azeem Haider | mh06858@st.habib.edu.pk

February 11, 2024

Dhanani School of Science and Engineering
Habib University
Fall 2023

Contents

1 Introduction

The first assignment of the course Computational Intelligence (CS-451) required us to implement evolutionary algorithms for three problems which are as followed:

1. **Travelling Salesman Problem:** The Travelling Salesman Problem (TSP) is a classic problem in combinatorial optimization. It is the problem of finding a tour of a set of n cities that is of minimum cost. A tour is a permutation of the cities, and the cost of a tour is the sum of the distances between adjacent cities in the tour.

2. **Job-Shop Scheduling Problem:** The job-shop scheduling problem is a classic problem in combinatorial optimization. It is the problem of scheduling a set of n jobs on a set of m machines. Each job consists of a sequence of operations, each of which must be processed on a specific machine for a specific amount of time.

3. **Evolutionary Art (Mona Lisa):** The evolutionary art problem is a classic problem in evolutionary computation. It is the problem of evolving a population of images to match a target image. Each image in the population is represented as a string of genes, and the fitness of an image is the similarity between the image and the target image.

2 Travelling Salesman Problem

2.1 Problem Formulation

The travelling salesman problem requires finding of the minimum cost (distance) to cover all the cities. The **chromosome generation** is being carried out by randomly shuffling the cities of the Qatar dataset.

Furthermore, the **mutate function** assigns each chromosome a random number from 0 to 1, and if the number is less than the mutation rate, mutation occurs.

Finally, the **crossover function** once again, carries out crossover between two parents by assigning the crossover point, and all the repeated cities from the second parent, that are already added in the first part, are added linearly in the end.

2.2 Analysis

While carrying out initial combination for selection schemes for parent selection and survival selection, it was clear that a combination of an explorative scheme as a parent selection and an exploitative scheme as a survival selection would be the best. As a result, two main combinations that were tested were the following:

1. **Parent Selection:** Random **Survival Selection:** truncation

2. **Parent Selection:** tournament selection **Survival Selection:** truncation

The result reported in *Figure 1*, was obtained by the following parameters:

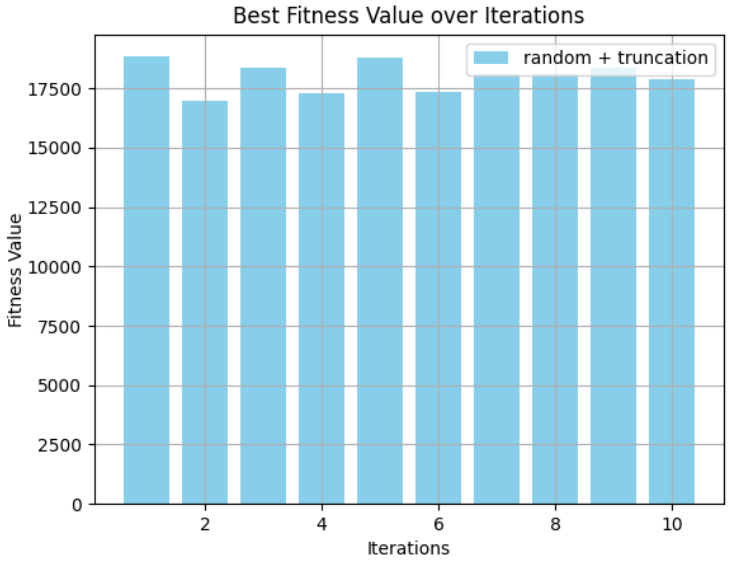

Figure 1: Travelling Salesman Problem: Combination 1

1. **Population Size:** 1000
2. **Offspring Size:** 200
3. **Number of Generations:** 5000
4. **Mutation Rate:** 0.45
5. **Iterations:** 10

3 Job-Shop Scheduling Problem

3.1 Problem Formulation

The job-shop scheduling problem requires finding the minimum time to complete all the jobs on all the machines.

The chromosome is generated by providing each job with an index, and repeating that number of index for the number of operations on that job. For example for abz5, if there are 10 jobs with 10 operations each, each job will be given an index from 0-9 and each number from 0-9 will repeat exactly 10 times. In this manner, we have flattened our chromosome.

The mutate and crossover function work in similar fashion to the travelling salesman problem. The mutate function checks probability assigned and if it is less than mutation rate, mutation occurs, and crossover occurs between two parents, and repeated jobs from parent two are added in the end of the new offsprings.

3.2 Analysis

Unlike, the travelling salesman problem, the job-shop scheduling problem had three input files which were abz(5-7). The analysis for each file will be divided into subsections.

3.2.1 First input file "abz5"

The strategy to solve the job-shop scheduling problem was to use the same combination of parent selection and survival selection as the travelling salesman problem. Since keeping the parent selection scheme entirely explorative through *random* function or on the higher exploration through rank-based selection or tournament selection. The survival selection was to be kept entirely exploitative through truncation or on the higher exploitation through fitness proportional selection. The best score achieved **1242** was through **random** and **truncation**, with the following parameters.

1. **Population Size:** 200
2. **Offspring Size:** 60
3. **Number of Generations:** 2000
4. **Mutation Rate:** 0.5
5. **Iterations:** 10

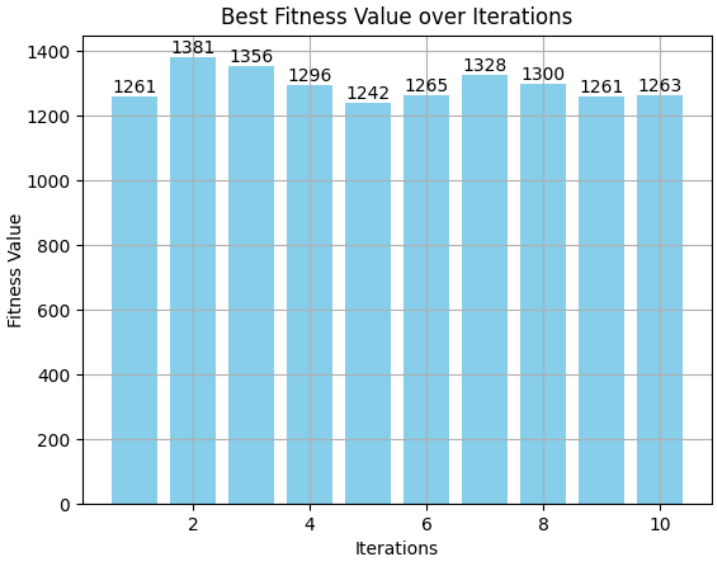

Figure 2: Job-shop Scheduling Problem

The following graph Figure 3, has the same selection schemes and parameters, except for the fact that it is ran on 25 generations, to attach statistics for the Best Fitness and Average Fitness.

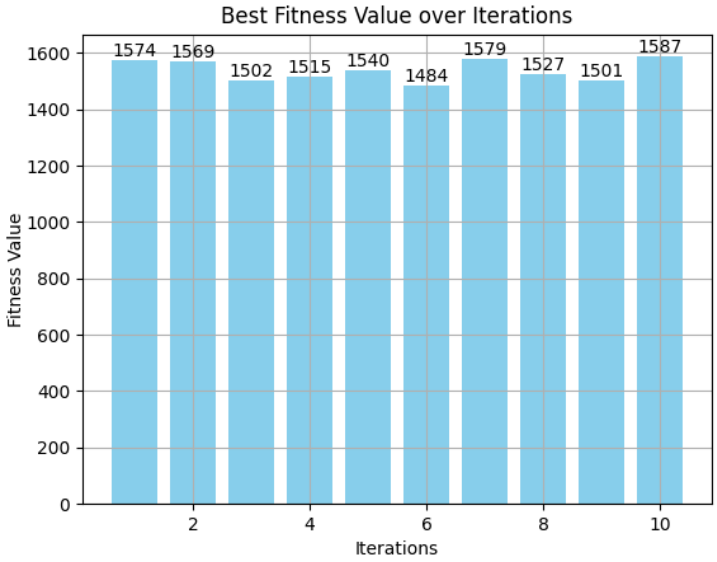

Figure 3: Job-shop Scheduling Problem

The following tables highlight the scores between the generations for 10 iterations, and the best and average scores amongst the generations.

Table 1: Generation-wise Fitness Scores

Generations	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10	Best Fitness Score
Gen 1	1718	1658	1741	1583	1640	1559	1656	1560	1533	1587	1533
Gen 2	1622	1658	1699	1583	1640	1559	1656	1560	1533	1587	1533
Gen 3	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1533
Gen 4	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1533
Gen 5	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1533
Gen 6	1622	1632	1654	1583	1640	1559	1656	1560	1533	1587	1533
Gen 7	1622	1607	1631	1583	1640	1559	1656	1560	1533	1587	1533
Gen 8	1588	1593	1631	1551	1640	1559	1601	1560	1533	1587	1533
Gen 9	1588	1593	1631	1551	1582	1559	1601	1560	1533	1587	1533
Gen 10	1588	1573	1631	1551	1582	1559	1601	1527	1533	1587	1527
Gen 11	1588	1573	1631	1551	1582	1559	1601	1527	1533	1587	1527
Gen 12	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 13	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 14	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 15	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 16	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 17	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1502
Gen 18	1588	1573	1502	1551	1582	1559	1601	1527	1529	1587	1502
Gen 19	1574	1573	1502	1551	1582	1559	1601	1527	1501	1587	1501
Gen 20	1574	1573	1502	1515	1540	1559	1601	1527	1501	1587	1501
Gen 21	1574	1573	1502	1515	1540	1559	1579	1527	1501	1587	1501
Gen 22	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1484
Gen 23	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1484
Gen 24	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1484
Gen 25	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1484

Table 2: Generation-wise Average Fitness Scores

Generations	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10	Average Fitness Score
Gen 1	1718	1658	1741	1583	1640	1559	1656	1560	1533	1587	1612.8
Gen 2	1622	1658	1699	1583	1640	1559	1656	1560	1533	1587	1613.8
Gen 3	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1612.2
Gen 4	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1612.4
Gen 5	1622	1646	1699	1583	1640	1559	1656	1560	1533	1587	1612.4
Gen 6	1622	1632	1654	1583	1640	1559	1656	1560	1533	1587	1608.0
Gen 7	1622	1607	1631	1583	1640	1559	1656	1560	1533	1587	1603.0
Gen 8	1588	1593	1631	1551	1640	1559	1601	1560	1533	1587	1595.3
Gen 9	1588	1593	1631	1551	1582	1559	1601	1560	1533	1587	1592.7
Gen 10	1588	1573	1631	1551	1582	1559	1601	1527	1533	1587	1588.5
Gen 11	1588	1573	1631	1551	1582	1559	1601	1527	1533	1587	1588.5
Gen 12	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 13	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 14	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 15	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 16	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 17	1588	1573	1502	1551	1582	1559	1601	1527	1533	1587	1576.5
Gen 18	1588	1573	1502	1551	1582	1559	1601	1527	1529	1587	1575.5
Gen 19	1574	1573	1502	1551	1582	1559	1601	1527	1501	1587	1567.0
Gen 20	1574	1573	1502	1515	1540	1559	1601	1527	1501	1587	1564.9
Gen 21	1574	1573	1502	1515	1540	1559	1579	1527	1501	1587	1562.9
Gen 22	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1556.6
Gen 23	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1556.6
Gen 24	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1556.6
Gen 25	1574	1573	1502	1515	1540	1484	1579	1527	1501	1587	1556.6

The ratio between population size and offsprings was tinkered with as well as the value of mutation rate. Other combinations of schemes such as rank-based selection as parent selection and truncation as survival selection yielded the following result, **1253** which came very close to our best score.

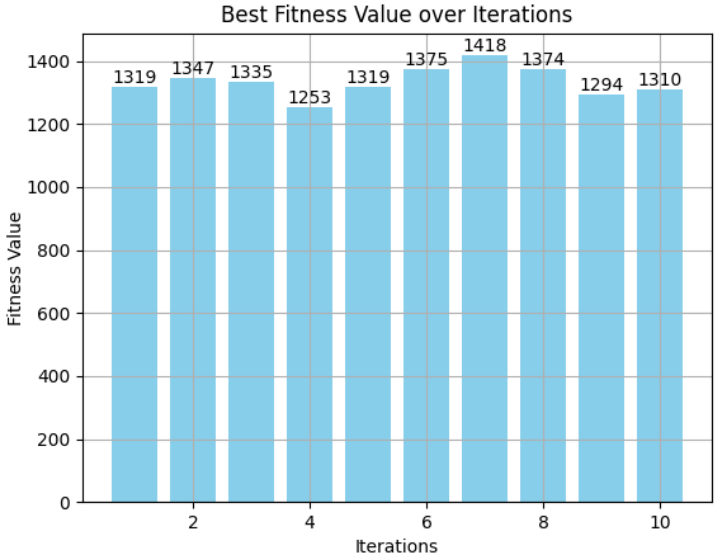

Figure 4: Job-shop Scheduling Problem

The following parameters were used to achieve the result in *Figure 3*:

1. **Population Size:** 150
2. **Offspring Size:** 40

3. **Number of Generations:** 2000
4. **Mutation Rate:** 0.5
5. **Iterations:** 10

3.2.2 Input files "abz6" and "abz7"

The combination of schemes for both the input files are random and truncation with the following parameters:

1. **Population Size:** 1000
2. **Offspring Size:** 200
3. **Number of Generations:** 500
4. **Mutation Rate:** 0.45
5. **Iterations:** 10

The best score for the input file "abz6" was **972**. Whereas, the best score for the input file "abz7" was **776**. The reason for the scores of input file "abz6" and "abd7" being lower than the input "abz5" is due to the fact that the time needed by each process on each machine is significantly low.

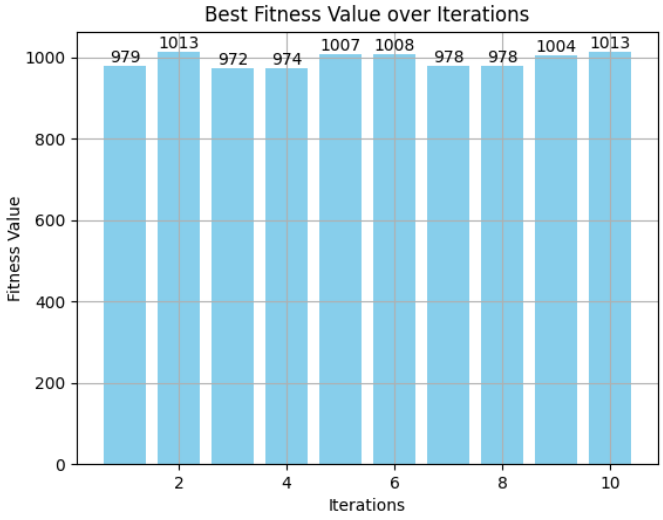

Figure 5: Job-shop Scheduling Problem: Input file abz6

The following table and graph are generated on the same selection schemes and parameters, except for the fact that the generation suze is 25.

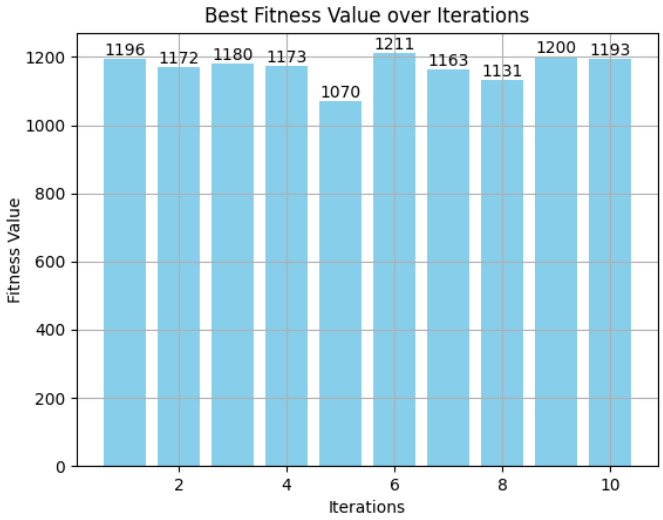

Figure 6: Job-shop Scheduling Problem: Input file abz6

Table 3: Generation-wise Best Fitness Scores for Input File `abz6`

Generations	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10	Best Fitness Score
Gen 1	1204	1240	1279	1224	1193	1288	1279	1214	1264	1231	1193
Gen 2	1204	1240	1279	1224	1193	1288	1279	1214	1264	1231	1193
Gen 3	1204	1240	1279	1224	1193	1288	1279	1214	1264	1231	1193
Gen 4	1204	1172	1279	1224	1193	1288	1279	1214	1264	1231	1172
Gen 5	1204	1172	1279	1224	1193	1273	1250	1214	1264	1231	1172
Gen 6	1204	1172	1279	1224	1193	1273	1250	1214	1264	1231	1172
Gen 7	1204	1172	1279	1224	1193	1273	1250	1214	1252	1231	1172
Gen 8	1204	1172	1274	1190	1193	1273	1250	1176	1252	1231	1172
Gen 9	1204	1172	1251	1190	1193	1273	1250	1176	1252	1231	1172
Gen 10	1204	1172	1251	1190	1182	1273	1250	1176	1252	1231	1172
Gen 11	1204	1172	1207	1190	1182	1273	1250	1176	1252	1231	1172
Gen 12	1204	1172	1207	1190	1182	1273	1250	1176	1252	1231	1172
Gen 13	1204	1172	1180	1190	1182	1273	1232	1176	1252	1231	1172
Gen 14	1204	1172	1180	1190	1182	1273	1232	1176	1252	1214	1172
Gen 15	1204	1172	1180	1190	1094	1232	1232	1176	1252	1214	1094
Gen 16	1196	1172	1180	1190	1094	1232	1232	1176	1252	1214	1094
Gen 17	1196	1172	1180	1190	1094	1232	1232	1176	1252	1214	1094
Gen 18	1196	1172	1180	1190	1094	1232	1232	1176	1243	1214	1094
Gen 19	1196	1172	1180	1190	1094	1232	1232	1176	1243	1214	1094
Gen 20	1196	1172	1180	1173	1094	1211	1204	1176	1243	1214	1094
Gen 21	1196	1172	1180	1173	1094	1211	1204	1176	1243	1214	1094
Gen 22	1196	1172	1180	1173	1094	1211	1204	1176	1219	1214	1094
Gen 23	1196	1172	1180	1173	1094	1211	1192	1176	1210	1214	1094
Gen 24	1196	1172	1180	1173	1070	1211	1192	1131	1210	1214	1070
Gen 25	1196	1172	1180	1173	1070	1211	1163	1131	1200	1193	1070

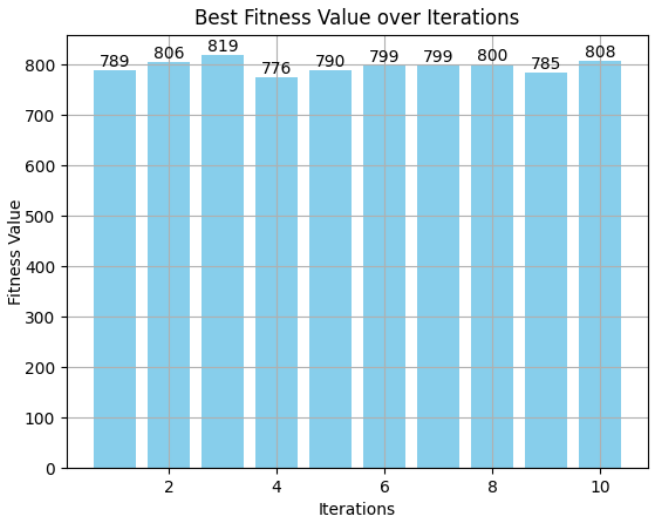

Figure 7: Job-shop Scheduling Problem: Input file abz7