CS 451 - Computational Intelligence
Assignment 1: Evolutionary Algorithm in Action
Report

Syeda Haya Fatima 07503, Zainab Haider 07104

February 11, 2024

Reported to: Syeda Saleha Raza

1 Basic Implementation of Evolutionary Algorithm

1.1 Traveling Salesman Problem

1.1.1 Chromosome Representation

The chromosome representation for the Traveling Salesman Problem (TSP) is implemented using a permutation-based encoding scheme. Each individual (or chromosome) in the population represents a possible route for the TSP as a permutation of all the cities. The order of cities in the permutation indicates the sequence in which the salesman visits them. This representation ensures that each city is visited exactly once, addressing the constraint of the TSP.

1.1.2 Fitness Function

The fitness function calculates the total distance traveled for a given route in the TSP. It iterates through each pair of consecutive cities in the route, computes the Euclidean distance between them using their coordinates from the provided TSP data, and accumulates these distances to determine the total distance traveled. The lower the fitness value, the better the route, as it indicates a shorter total distance traveled.

1.1.3 Other Features Implemented

We further implemented a Hall of Fame text file where any solution with fitness below 12000 is appended. This was to promote elitism mechanism. Without it, hitting the 10000s or below depended on the random initialization more than our finetuning and selection scheme. However, this quickly led to a loss of diversity and result converged quickly with the elite taking over the entire population.

The second feature that we tried out was generational replacement. It had very unpredictable and random results, so we did not further finetune it. Perhaps a much larger population of offspring and truncation survivor selection scheme would do the trick. In the scope of our problem, we tried 100 parents and 200 offsprings with truncation selection but the results did not look promising.

The graphs for the above are plotted below:

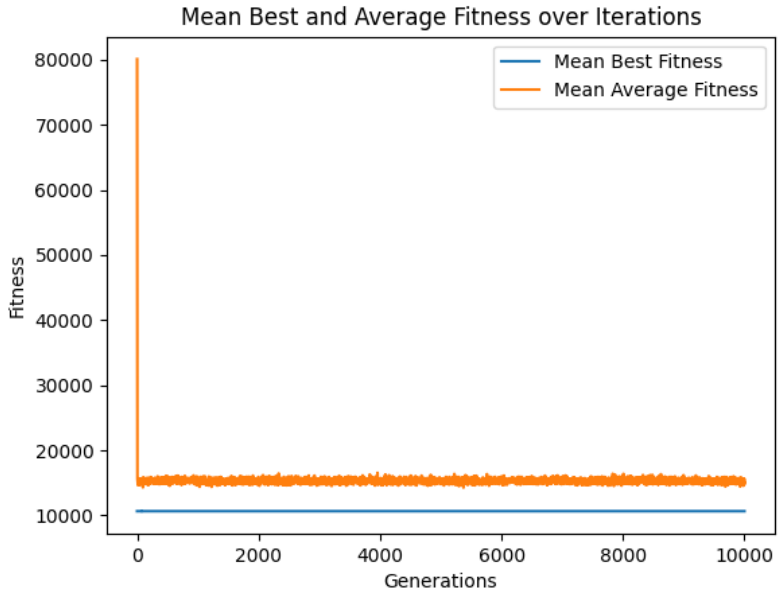


Figure 1: Elitism taking over population

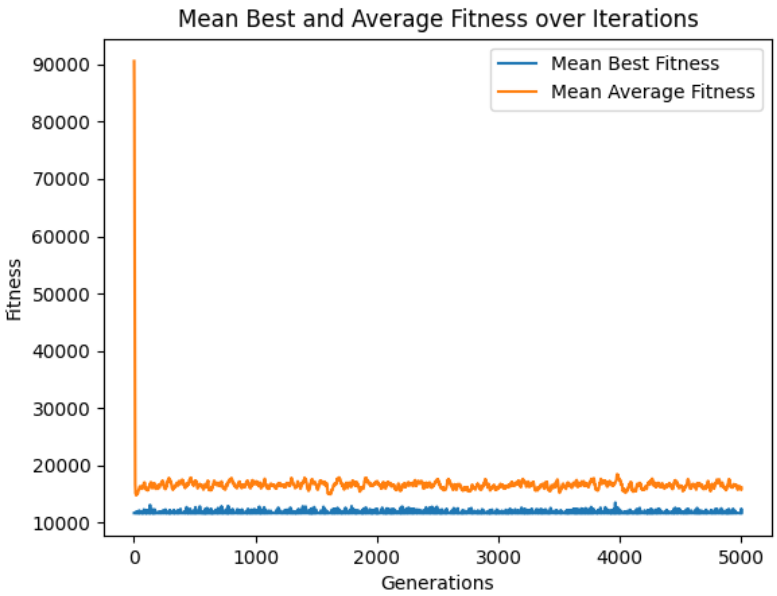


Figure 2: Elitism with more explorative selection schemes

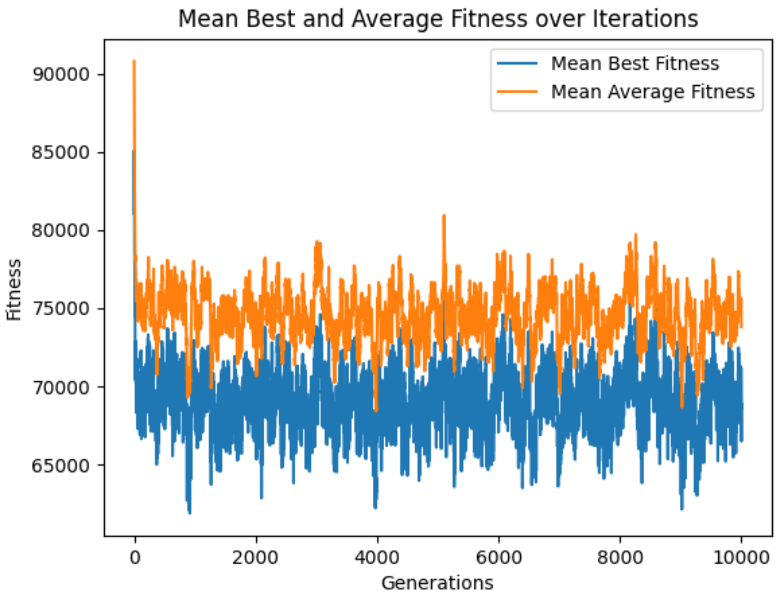


Figure 3: Generational Replacement

1.1.4 Optimal Values Achieved and Optimal Solution

Generally, despite multiple iterations and averaging of values, random initializing did play a significant role. Other techniques like crossover and mutation techniques also had varying results.

Strictly speaking about the optimal value we achieved, it was with the following settings:

Population Size: 200
Offspring Size: 200
Mutation Technique: Displacement Mutation
Mutation Rate: 0.75
Number of Generations: 10000
Parent Selection Scheme: Binary Tournament Selection
Survivor Selection Scheme: Truncation Selection

The graph for the same is plotted below (without elitism):

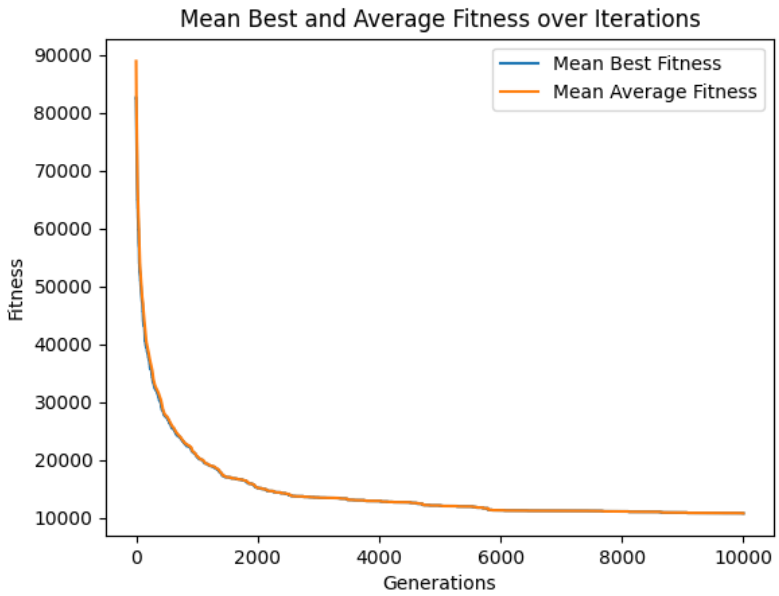


Figure 4: Best Results Achieved on the above Settings (Without elitism)

Best Fitness: 10580.219767432689

Solution: [82, 80, 87, 102, 83, 79, 77, 70, 64, 45, 60, 57, 69, 74, 72, 78, 75, 76, 71, 23, 25, 13, 14, 17, 11, 7, 4, 2, 3, 5, 10, 9, 18, 21, 26, 24, 33, 28, 29, 22, 27, 37, 68, 66, 61, 51, 39, 47, 43, 48, 46, 41, 31, 32, 30, 19, 15, 12, 34, 40, 38, 35, 44, 42, 49, 50, 55, 54, 52, 53, 58, 56, 67, 73, 84, 81, 88, 92, 97, 96, 93, 91, 103, 95, 106, 105, 107, 108, 100, 110, 112, 115, 116, 117, 118, 131, 135, 133, 128, 124, 123, 120, 121, 129, 136, 143, 148, 160, 155, 151, 147, 159, 158, 162, 166, 171, 170, 178, 181, 177, 184, 175, 173, 174, 179, 176, 182, 161, 163, 164, 169, 172, 183, 186, 194, 187, 190, 191, 192, 189, 188, 193, 185, 180, 167, 168, 165, 157, 154, 146, 149, 156, 145, 132, 134, 130, 127, 126, 125, 137, 142, 140, 138, 139, 152, 153, 150, 144, 141, 122, 119, 113, 109, 114, 111, 90, 89, 94, 101, 99, 104, 98, 85, 86, 65, 20, 63, 62, 59, 36, 16, 8, 6, 1]

After we attained some high fitness scores, we leveraged the elitism strategy. We identified and preserved the best-performing individuals in each generation. As we observed improvements in fitness over iterations, we selectively retained these superior individuals, allowing them to influence subsequent generations. This iterative process of retaining and building upon the most successful solutions ultimately led us to discover the optimal individual.

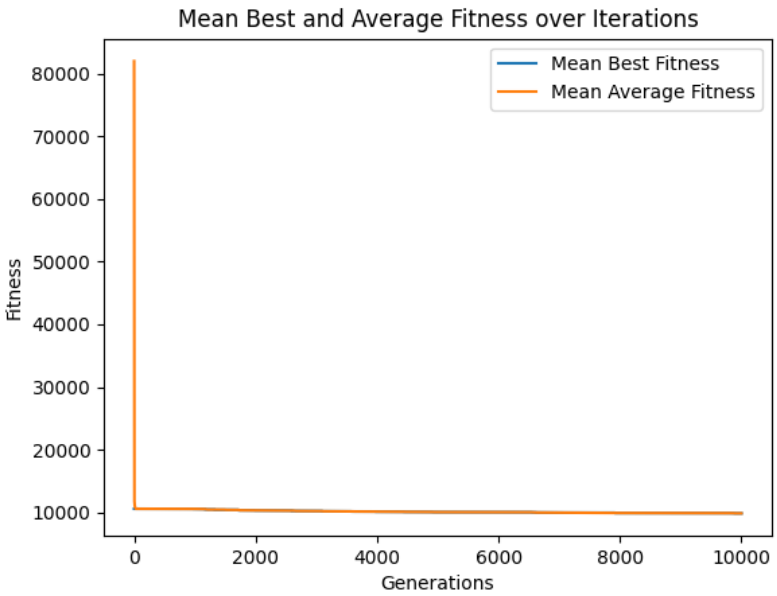


Figure 5: Best Results Achieved on the above Settings (With elitism)

Best Fitness: 9573.117971832517

Solution: [7, 4, 2, 3, 5, 9, 10, 12, 15, 19, 30, 32, 35, 31, 27, 37, 29, 22, 28, 18, 21, 24, 26, 33, 45, 68, 66, 51, 47, 39, 34, 40, 43, 48, 46, 41, 38, 44, 42, 50, 49, 55, 54, 52, 53, 56, 58, 61, 67, 73, 84, 92, 97, 95, 96, 106, 105, 107, 108, 100, 110, 112, 115, 116, 117, 118, 131, 136, 135, 133, 128, 124, 123, 120, 121, 129, 143, 148, 160, 155, 151, 158, 162, 166, 171, 180, 178, 181, 177, 184, 175, 173, 174, 183, 194, 182, 176, 169, 161, 163, 164, 172, 179, 186, 187, 190, 192, 191, 189, 188, 193, 185, 170, 167, 168, 165, 159, 147, 152, 153, 150, 157, 154, 146, 149, 142, 140, 145, 156, 130, 127, 125, 126, 132, 134, 137, 138, 139, 144, 141, 122, 119, 113, 109, 114, 111, 104, 101, 99, 94, 89, 90, 98, 86, 85, 65, 20, 63, 36, 62, 59, 82, 80, 87, 102, 103, 91, 93, 88, 83, 81, 79, 77, 70, 64, 57, 60, 69, 74, 72, 78, 75, 76, 71, 25, 23, 17, 14, 11, 13, 16, 8, 6, 1]

You may also refer to the HallofFame.txt file and refer to some of the best individuals we found. We halted our optimization process despite ongoing improvements due to time constraints.

1.1.5 Plots

Using the preceding settings, we generated the following plots on varying selection schemes.

FPS + Random

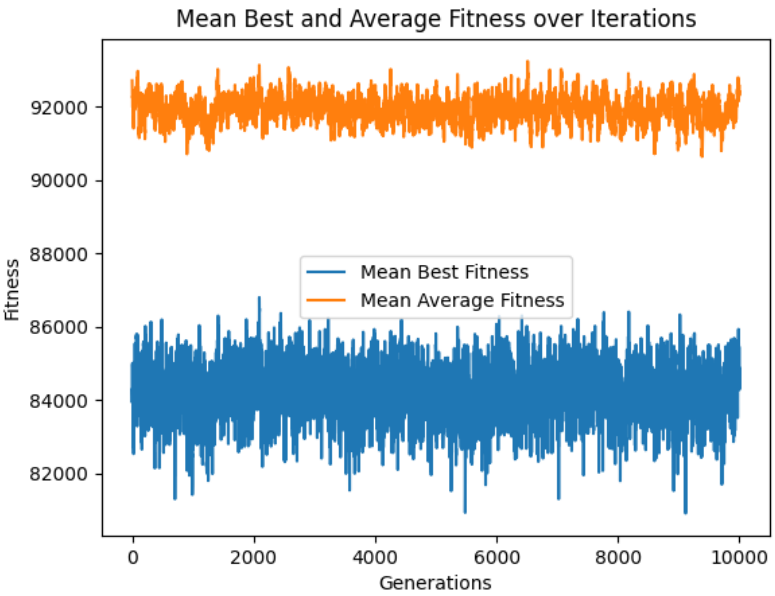


Figure 6: FPS + Random

Binary Tournament + Truncation

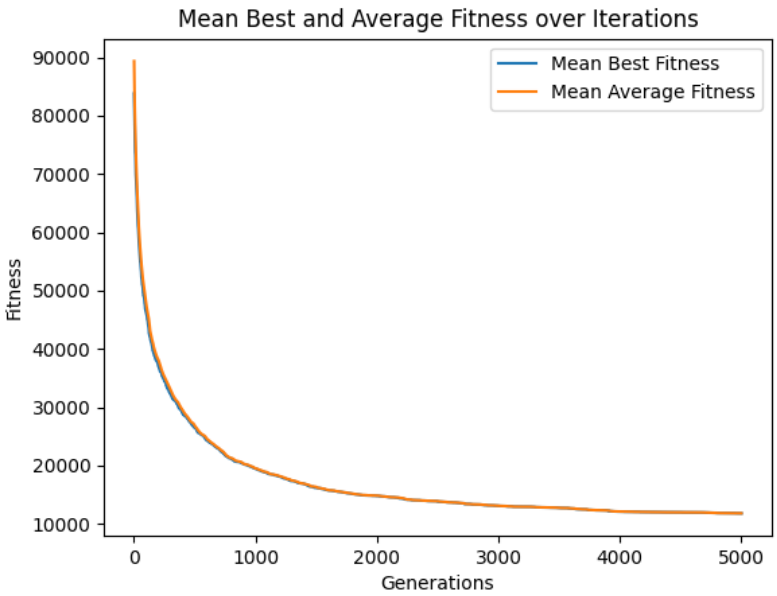


Figure 7: BTS + Truncation

Truncation + Truncation

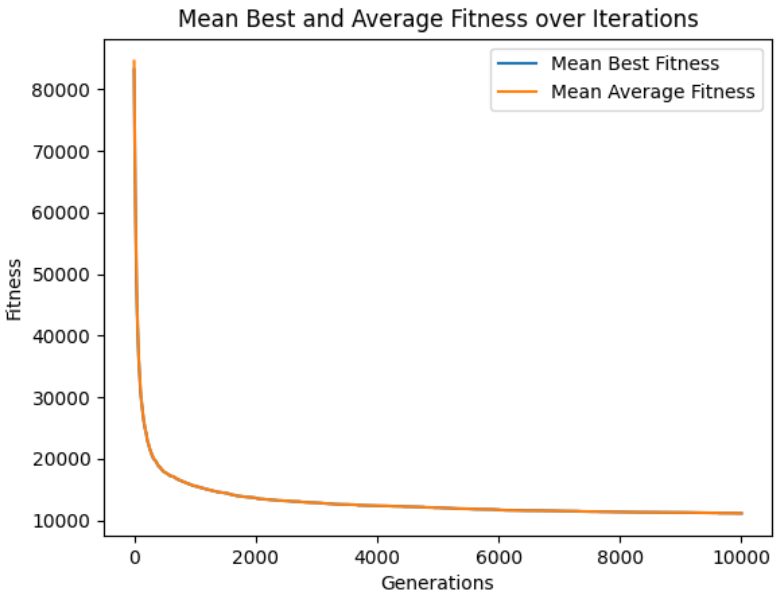


Figure 8: Truncation + Truncation

Random + Random

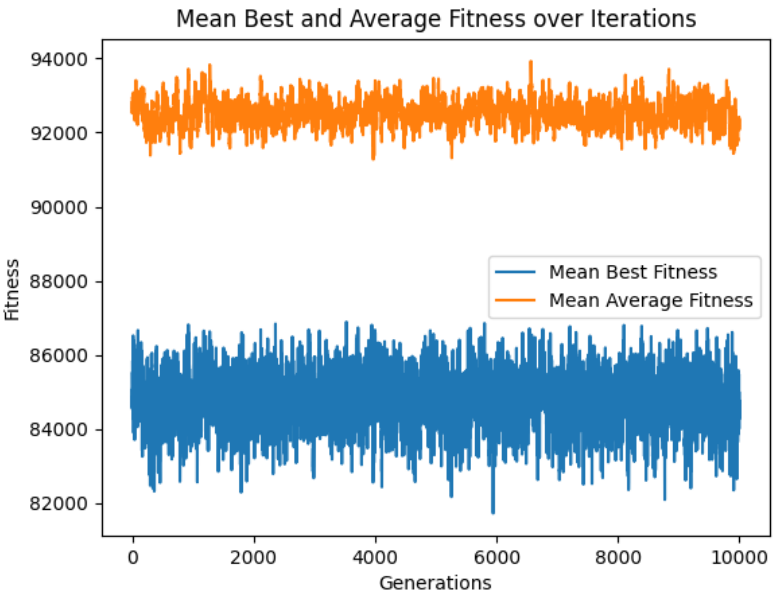


Figure 9: Random + Random

FPS + RBS

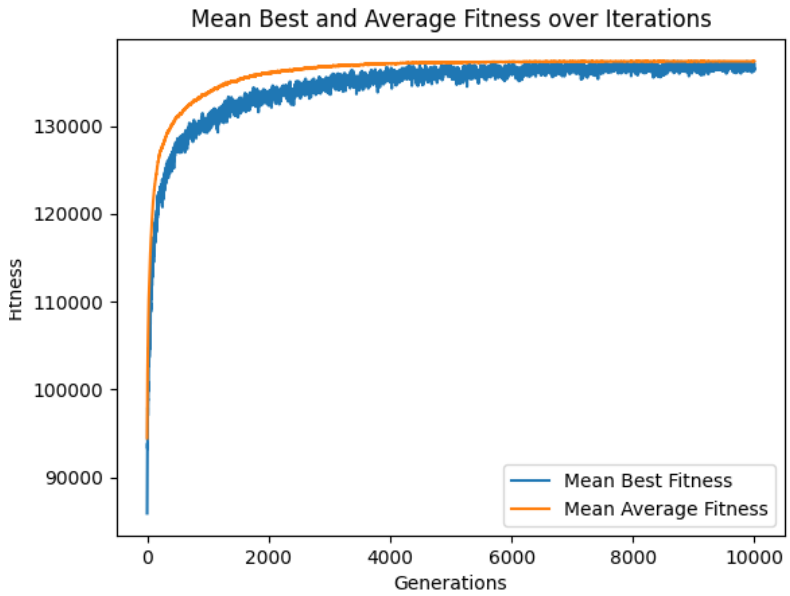


Figure 10: FPS + RBS

1.1.6 Overall Analysis

Random only resulted in, as expected, random values not converging to anything. Whereas truncation only did actually get pretty close, but because it converged so quickly, it lost diversity and subsequently, the fittest took over the entire population. And there was not much variation between average values and fit values. FPS + Random made no sense as we were not converging to fitter individuals but rather random ones, and fitness-proportional was also more random since we had a huge number to pick from. Lastly, in FPS + RBS, the fitness proportional selection may lead to the premature convergence of the population towards sub-optimal solutions due to the proportional emphasis on the fittest individuals. Meanwhile, rank-based selection might not adequately preserve the diversity of the population, potentially hindering the exploration of the search space, both of which could result in a decline in overall fitness. Adjusting selection pressures or exploring alternative selection methods could help mitigate this issue. Binary Tournament and Truncation were perhaps the best and most consistent results we were getting, thats why we stuck with it when trying to get the best solution. Binary tournament selection and truncation selection complement each other effectively because binary tournament selection introduces diversity by selecting individuals based on random pairwise comparisons, while truncation selection focuses on preserving the fittest individuals in the population. This combination helps balance exploration and exploitation, facilitating a more comprehensive search of the solution space while prioritizing

the retention of the best solutions.

1.2 Job-Shop Scheduling Problem

1.2.1 Chromosome Representation

In our scheduling strategy, chromosomes represent potential schedules by breaking down jobs into individual operations and sequencing them across machines. Each chromosome consists of a sequence of job numbers indicating the order of operations across machines. For instance, "1524325413" denotes Operation 1 of Job 1 followed by Operations 1 of Job 5, Job 2, Job 4, and Job 3 successively.

This instance is repeated the number of operations times. The first instance schedules first operation for each job as in the example above. The second instance schedules second operation for each job and so on.

The frequency of each job number in the chromosome signifies the order of operations within each job. Although it is somewhat restricting, this approach ensures explicit capture of operation order within jobs and adherence to the machine order for each job.

1.2.2 Fitness Function

The fitness function first converts each chromosome into a schedule by considering the dataset and resolving overlaps between operations. It calculates idle times between jobs to optimize efficiency. Once the schedule is established, the function evaluates it by determining the maximum time taken on each machine. This is achieved by identifying the last job scheduled on each machine and calculating its end time. By recording these end times and identifying the maximum value, the fitness function generates a fitness value (C_max) denoting the total time for all jobs to be completed across all machines.

1.2.3 Optimal Value Achieved and Optimal Solution

The optimal solutions were calculated using the following settings:
Population Size: 50
Offspring Size: 40
Mutation Technique: Insert Mutation
Mutation Rate: 0.2
Number of Generations: 150
Parent Selection Scheme: Rank based Selection
Survivor Selection Scheme: Truncation Selection

The best results of each dataset are mentioned below:

Dataset 1 (abz5): The optimal value we achieved for this dataset was 1336 mins. The solution associated with this value was plotted and is displayed in figure 10.

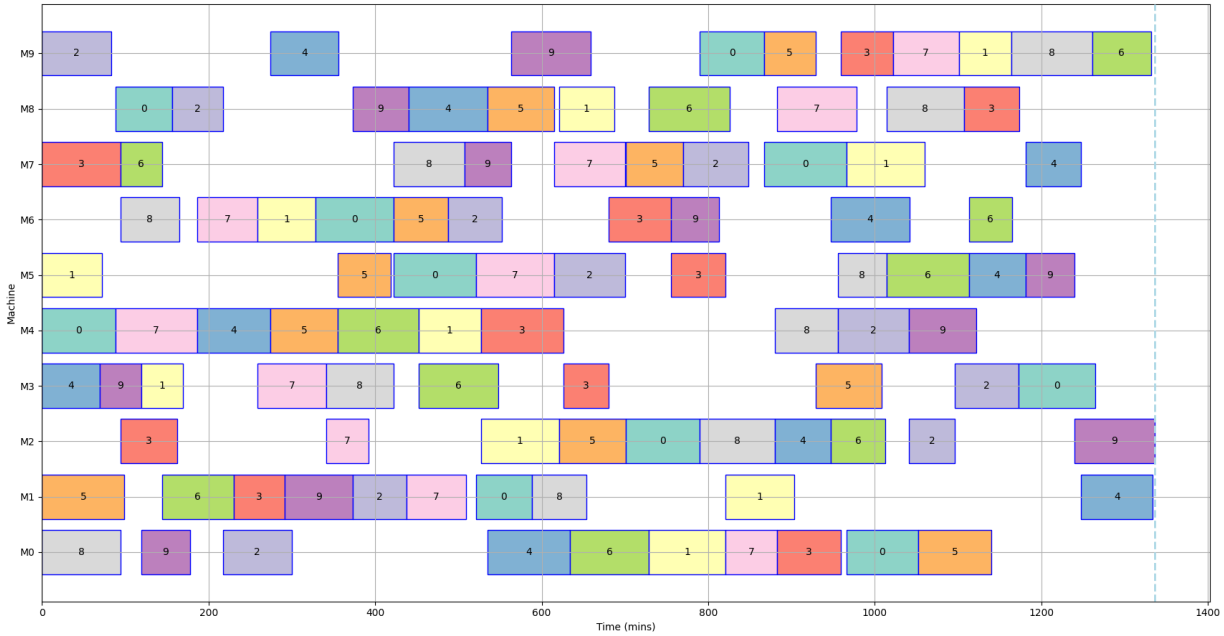


Figure 11: Optimal solution for dataset abz5

Dataset 2 (abz6): The optimal value we achieved for this dataset was 1032 mins. The solution associated with this value was plotted and is displayed in figure 11.

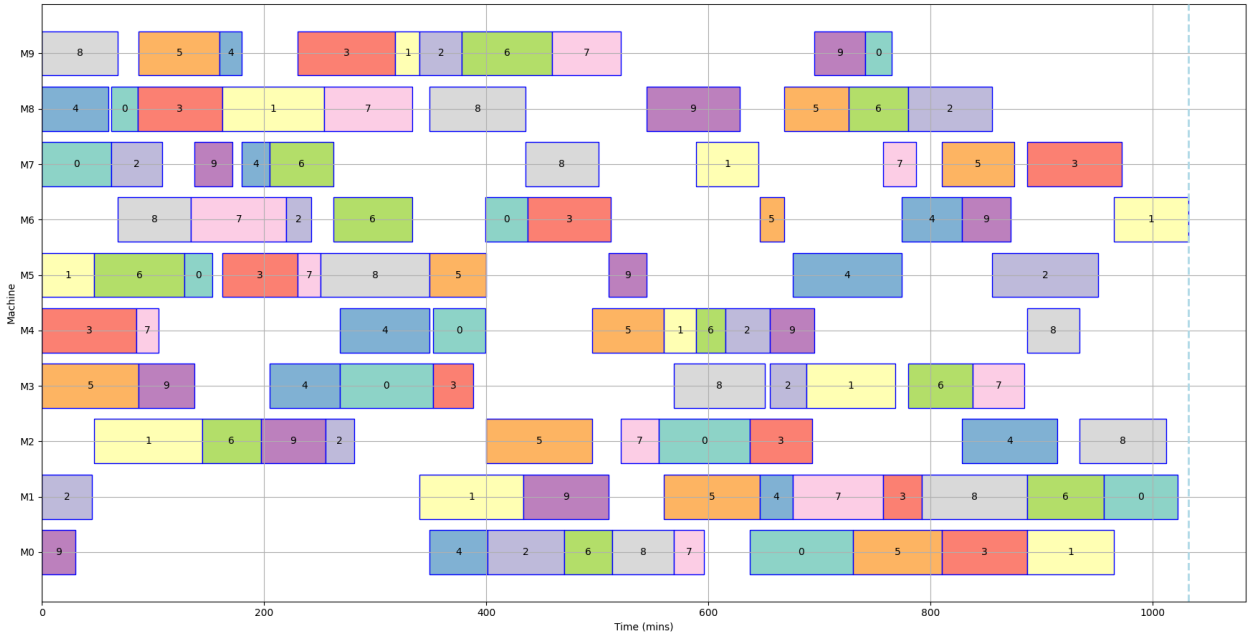


Figure 12: Optimal solution for dataset abz6

Dataset 3 (abz7): The optimal value we achieved for this dataset was 783 mins. The solution associated with this value was plotted and is displayed in figure 12.

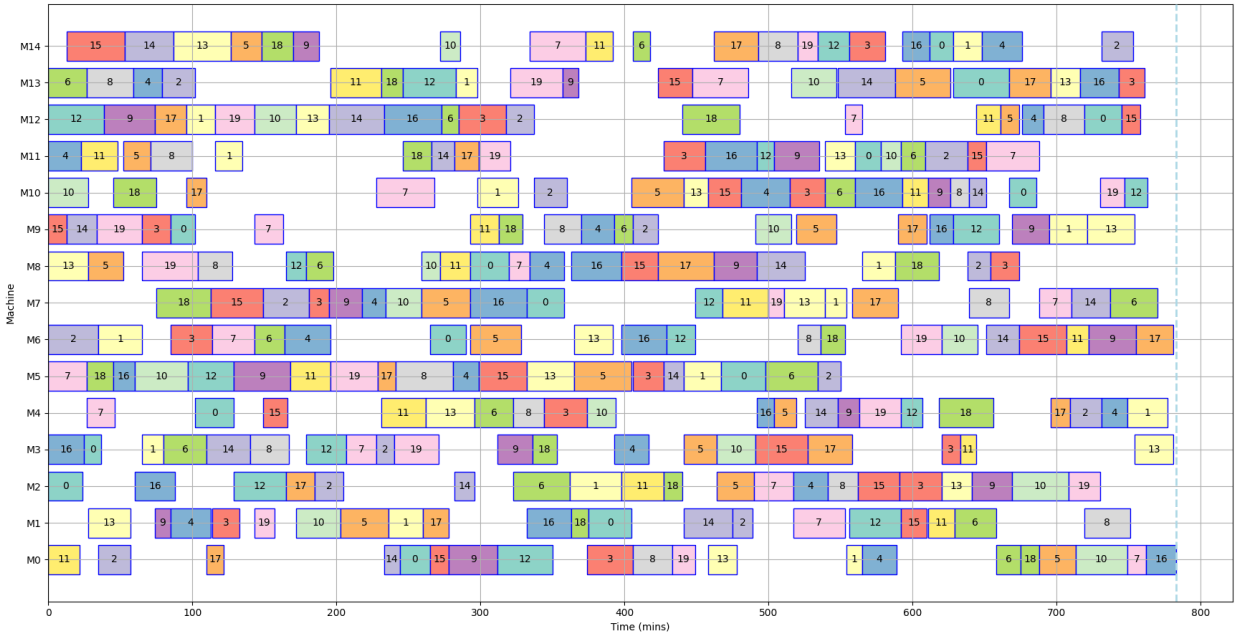
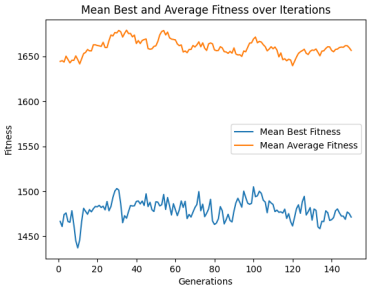


Figure 13: Optimal solution for dataset abz7

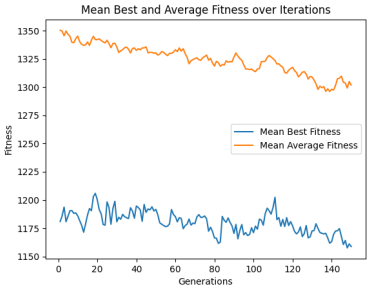
1.2.4 Plots:

Avg. BSF and Avg. ASF for various combinations of schemes using the settings above are shown, for all 3 datasets.

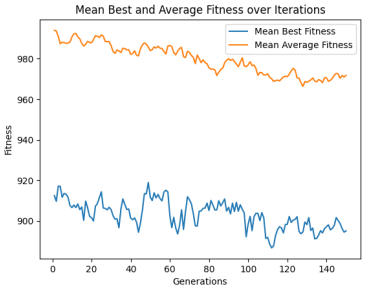
- **FPS + Random:**



(a) abz5



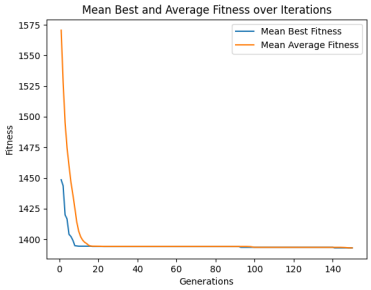
(b) abz6



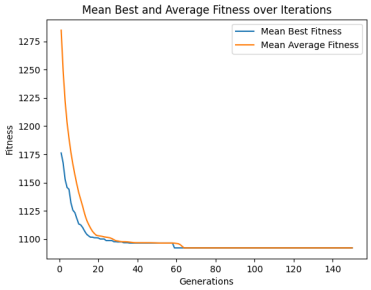
(c) abz7

Figure 14: FPS + Random

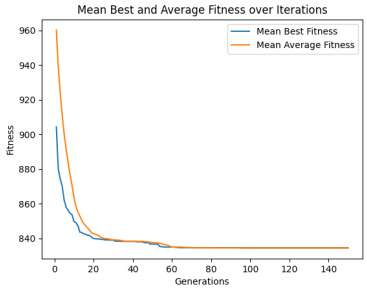
- **Binary Tournament + Truncation:**



(a) abz5



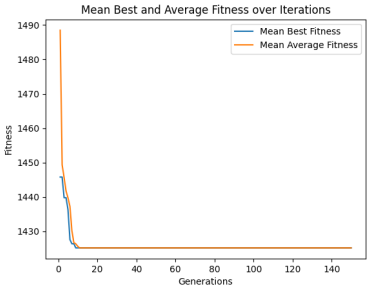
(b) abz6



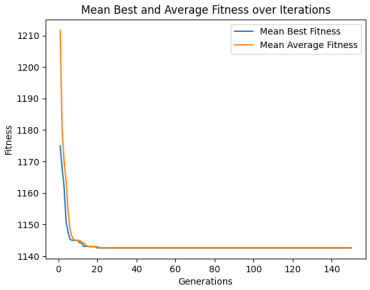
(c) abz7

Figure 15: Binary Tournament + Truncation

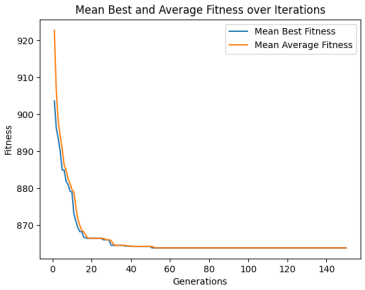
- **Truncation + Truncation:**



(a) abz5



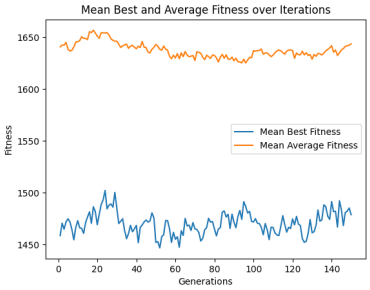
(b) abz6



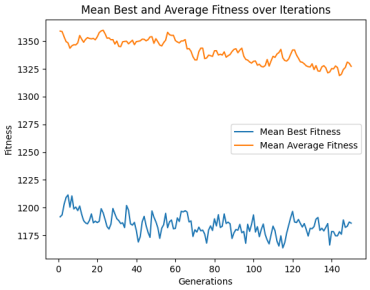
(c) abz7

Figure 16: Truncation + Truncation

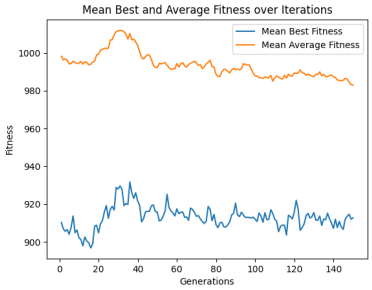
- **Random + Random:**



(a) abz5



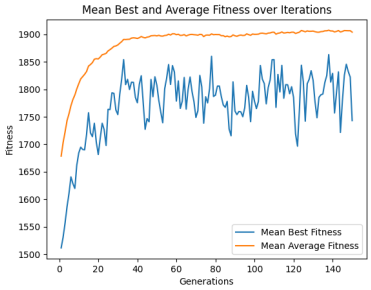
(b) abz6



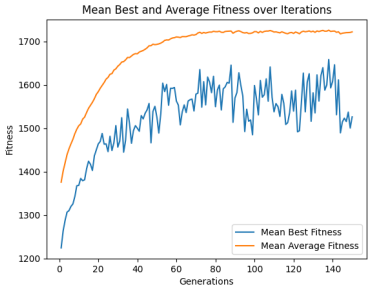
(c) abz7

Figure 17: Random + Random

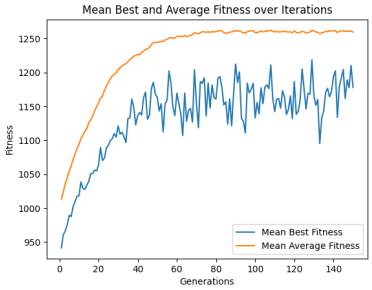
- **FPS + RBS:**



(a) abz5



(b) abz6



(c) abz7

Figure 18: FPS + RBS

Overall Analysis

In our analysis of the Job Shop Scheduling Problem (JSSP) model, we found that while the model produced accurate results, it exhibited limited evolution over successive generations. Our most successful outcomes across various datasets were consistently achieved by employing Fitness Proportionate Selection (FPS) or Rank Based Selection (RBS) in conjunction with truncation survivor selection. Notably, using a population size of 50, generating 40 offspring per iteration, and setting the mutation rate to 0.2 yielded optimal performance. However, we observed that combining FPS with RBS resulted in an unexpected increase in fitness values rather than the anticipated decrease proving to be the worst performing. Additionally, random selection schemes introduced fluctuations in fitness levels without significant overall improvement. Consequently, our analysis recommends adopting FPS and RBS for parent selection, complemented by truncation survivor selection, to achieve the most favorable results in solving the JSSP.

2 Evolutionary Art

2.1 Chromosome Representation:

The chromosome representation comprises two main classes: Individual and Polygon. The Individual class represents a solution, with each instance generating an image using semi-transparent polygons. It stores the image dimensions (imageSize) and a list of polygons forming the image. Each polygon, represented by the Polygon class, includes attributes such as color (in RGBA format for transparency) and points (polygon vertices).

2.2 Random Initialization:

A specified number of chromosomes are initialised where each chromosome consists of multiple polygons. For each chromosome, a random set of polygons is generated, with each polygon having a random color and a random number of vertices. The polygons are semi-transparent, with an RGBA color value and a set of vertices.

2.3 Fitness Function:

The fitness function calculates image dissimilarity by converting target and drawn images into arrays. It measures differences between corresponding pixels, sums them for overall dissimilarity, and computes fitness as the square root of the sum of squared differences. To speed up evolution, we also used parallel computing to calculate the fitness across population in parallel.

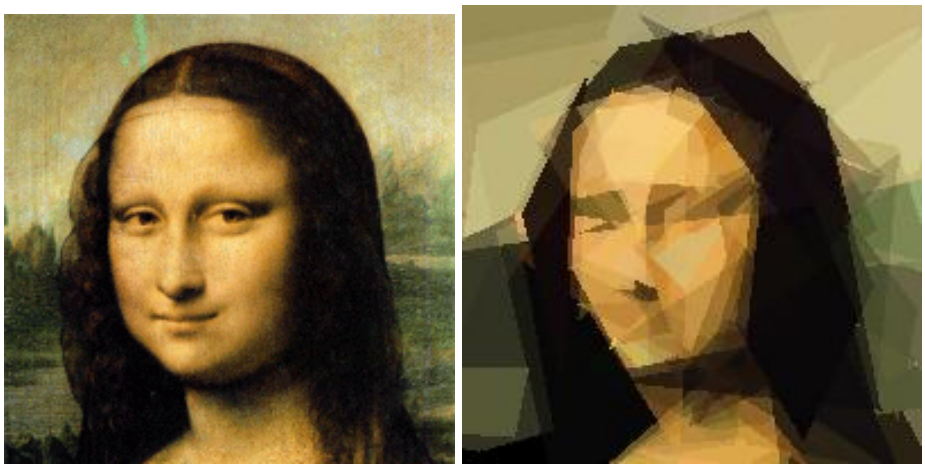
2.4 Our Approach:

In our approach, we experimented with various crossover and mutation methods within the framework of conventional Evolutionary Algorithms (EAs). Despite testing multiple crossover techniques, we encountered challenges in achieving convergence, particularly due to the inherent complexity of the task. Simply combining polygons from different parents through crossover yielded inconsistent results, often leading to suboptimal outcomes. Recognizing the limitations of traditional crossover strategies, we opted to focus solely on mutation. This decision was motivated by the observation that randomly selecting polygons from parents tended to produce inferior results rather than improvements. By exclusively employing mutation and implementing a truncation scheme for survivor selection, we achieved the most promising outcomes. Consequently, we adopted this mutation-centric approach as it consistently delivered the best results and ensured that our target image was reconstructed accurately.

2.5 Optimal Value Achieved and Optimal Solution:

We tested our program on multiple images and the following outputs in figure 18, 19 and 20 offer a clear understanding of the program's capabilities across

different image types:



(a) Original Image (b) Reconstructed Image

Figure 19: Result of Mona Lisa Image

(a) Original Image (b) Second image

Figure 20: Result of Afghani girl Image

(a) Original Image (b) Second image

Figure 21: Result of Girl with a Pearl Earing Image

2.6 Plots:

Figure 21 shows a succession of images generated during the evolution of the Mona Lisa image which demonstrates the evolution process clearly:

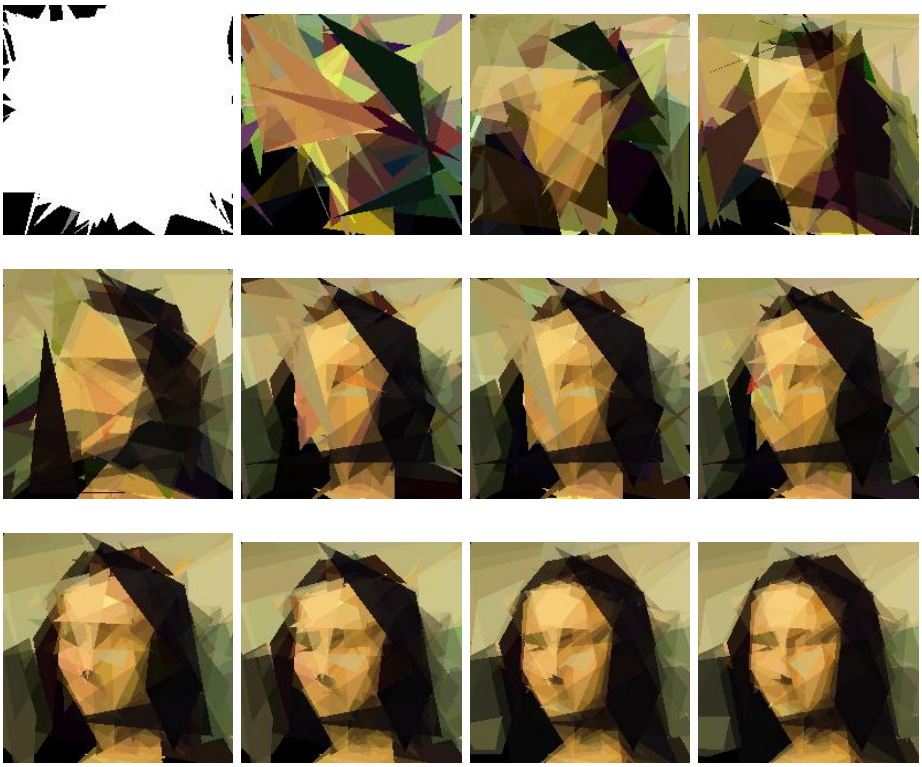


Figure 22: Evolution process of Mona Lisa

Overall Analysis

Our analysis of the evolutionary art program highlights several key points. Firstly, we successfully generated a reproduction of the Mona Lisa image, demonstrating the effectiveness of our approach. Testing on various other images also yielded correct results, indicating the versatility of our program. Increasing the number of generations improved output quality, enhancing the clarity of features in reproduced images. However, output generation was time-consuming, suggesting a need for optimization. We implemented slight modifications in mutation and survivor selection processes, deviating from conventional Evolutionary Algorithm approaches. To address performance issues, we maintained a low population size of 5, as increasing it led to program slowdowns. Despite its slowness, the program accurately reproduces images beyond the Mona Lisa, showcasing its efficacy and potential for further improvement.