Assignment 1 – Evolutionary Algorithm
CS 451 - Computational Intelligence, Spring 2025

Syeda Samah Daniyal 07838, Aina Shakeel 08430

February 9, 2025

Contents

1 Objective **2**

2 Problem Formulation **2**
2.1 Travelling Salesman Problem (TSP) 2
2.1.1 Methodology + Chromosome Representation 2
2.1.2 Initialization 2
2.1.3 Fitness Function 2
2.1.4 Parent Selection Mechanisms 3
2.1.5 Crossover 3
2.1.6 Mutation 3
2.1.7 Survivor Selection 3
2.1.8 Algorithm Execution 3
2.1.9 Results and Analysis 3
2.1.10 Conclusion 10
2.2 Job-Shop Scheduling Problem (JSSP) 10
2.2.1 Chromosome Representation 10
2.2.2 Fitness Function 10
2.2.3 Result + Analysis 10
2.2.4 Overall Analysis 16

3 Evolutionary Art **16**
3.1 Chromosome Representation 16
3.2 Fitness Function 17
3.3 Optimal Results Achieved 17
3.4 Results Across Generations 17
3.5 Overall Analysis 18

1 Objective

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

2 Problem Formulation

2.1 Travelling Salesman Problem (TSP)

This subsection describes the TSP, where the goal is to find the shortest possible route that visits each city exactly once and returns to the origin city. The Qatar dataset with 194 cities will be used for testing, and the known optimal length is 9352. The dataset is available at `http://www.math.uwaterloo.ca/tsp/world/countries.html`.

2.1.1 Methodology + Chromosome Representation

In the context of solving the Travelling Salesman Problem (TSP) using an Evolutionary Algorithm (EA), a chromosome represents a possible solution, which is a permutation of city indices. Each chromosome encodes an ordered list of cities that the salesman must visit in sequence. Mathematically, a chromosome can be represented as:

$$C = (c_1, c_2, c_3, \ldots, c_n) \tag{1}$$

where c_i represents the index of a city in the problem instance, and n is the total number of cities. The objective is to find a permutation that minimizes the total travel distance, computed using the Euclidean distance between consecutive cities:

$$d(C) = \sum_{i=1}^{n-1} \sqrt{(x_{c_{i+1}} - x_{c_i})^2 + (y_{c_{i+1}} - y_{c_i})^2} \tag{2}$$

where (x_{c_i}, y_{c_i}) denotes the coordinates of city c_i.
The EA follows a generational approach, where a population of candidate solutions evolves over multiple generations using selection, crossover, and mutation operations.

2.1.2 Initialization

A population of 300 chromosomes is generated, where each chromosome represents a permutation of city indices.

2.1.3 Fitness Function

Fitness is computed as the total Euclidean distance traveled in a given chromosome (tour representation).

2.1.4 Parent Selection Mechanisms

Various selection mechanisms were implemented:

- Fitness Proportional Selection (FPS)
- Rank-Based Selection (RBS)
- Binary Tournament Selection (BTS)
- Random Selection (RS)

2.1.5 Crossover

We used an order-based crossover method where a subset of one parent's genes is inherited, and missing values are filled based on another parent's order.

2.1.6 Mutation

A two-opt mutation is applied to introduce diversity by swapping segments within a chromosome.

2.1.7 Survivor Selection

The following strategies were explored:

- Fitness Proportional Selection
- Rank-Based Selection
- Binary Tournament Selection
- Random Selection
- Truncation Selection (TS)

Elitism was incorporated, preserving the best 20 individuals in each generation.

2.1.8 Algorithm Execution

The EA was run for 15 iterations, with 3000 generations per iteration. Hall of Fame (HOF) tracking was used to maintain the best solutions found across generations.

2.1.9 Results and Analysis

Figure 1 shows the progress of the algorithm over generations. The Average Best Fitness (BSF) and Average Fitness (ASF) indicate improvement as the number of generations increases.

The best overall solution obtained in our implementation achieved a tour length of i.e. the Best Fitness Score of: **10975.929027155857** with the Best Solution Population: **[1, 2, 4, 6, 8, 20, 65, 90, 85, 86, 98, 130, 127, 125, 126, 132, 134, 137, 140, 142, 146, 145, 156, 149, 139, 138, 154, 150, 144, 141, 153, 152, 157, 164, 163, 161, 169, 176, 182, 172, 179, 186, 194, 190, 187, 183, 174, 173, 175, 189, 192, 191, 188, 184, 181, 177, 168, 178, 180, 193, 185, 171, 166, 170, 167, 165, 159, 162,**

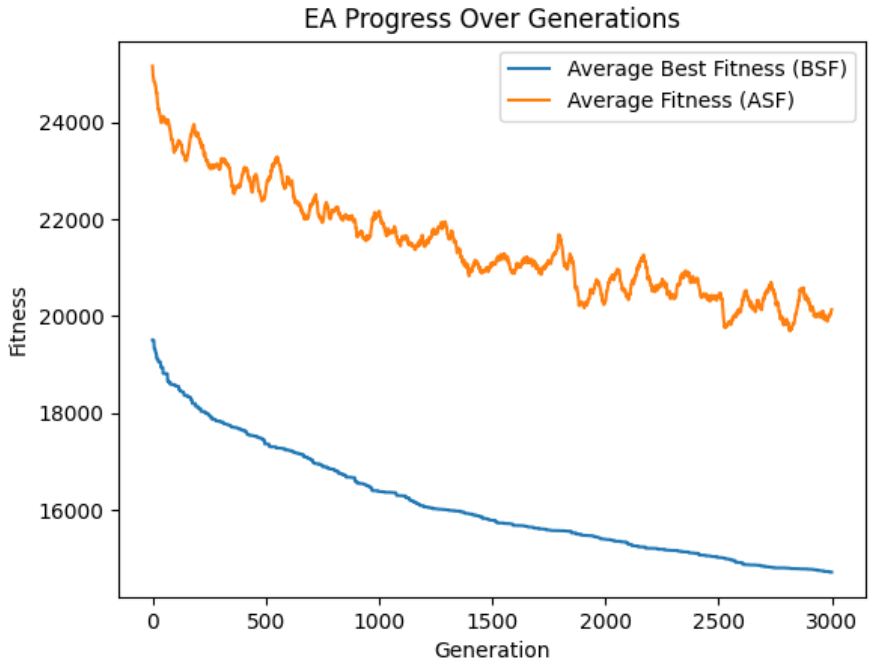


Figure 1: Best Result Achieved

158, 160, 136, 131, 147, 151, 155, 148, 143, 135, 129, 133, 128, 124, 123, 120, 121, 117, 116, 108, 97, 93, 96, 95, 92, 88, 91, 87, 80, 76, 75, 78, 72, 74, 69, 60, 57, 45, 28, 33, 18, 21, 24, 26, 17, 14, 13, 23, 16, 36, 63, 59, 62, 71, 25, 11, 7, 3, 5, 9, 10, 12, 15, 19, 30, 32, 31, 27, 22, 29, 37, 39, 40, 38, 41, 46, 44, 54, 42, 35, 50, 55, 49, 48, 52, 53, 56, 43, 34, 47, 51, 58, 61, 67, 73, 66, 68, 64, 70, 77, 79, 81, 83, 84, 100, 110, 112, 115, 107, 105, 106, 122, 118, 103, 102, 109, 113, 119, 114, 111, 104, 101, 94, 99, 89, 82]

This was achieved with the following parameters:

- Population = 300
- Generations = 3000
- Mutation Rate = 0.3
- Offsprings = 300
- Iterations = 15
- Mutation Technique = Two-Opt Mutation
- Parent Selection Scheme = Rank Based Selection
- Survivor Selection Scheme = Truncation Selection

Through this process we learned that every aspect of EA was important in achieving the optimal result. We needed to keep the initialization of the population partially random along with the Hall of Fame to utilize exploration and exploitative techniques. Initially when the population was just randomized the EA evolved more slowly and needed

a larger population set to get to 18000 fitness score. When we only utilized Hall of Fame the Fitness Scores got stagnant sooner, i.e premature convergence at a local optimum. Furthermore, what kind of mutation and what rate was highly significant as well. In our experience higher mutation rate led to premature convergence but when it was lower the EA kept getting better with each generation and iteration.
Furthermore, the selection techniques played a huge role in optimization. For us **truncation selection** worked best as a survival selection technique as it ensured we were progressing each generation and iteration, and coupled with HoF it improved significantly. We selected the 20 best individuals of each generation for the HoF. For parent selection, we observed that the worst result was with **randomized selection** as there was no steady improvement or convergence, and the best was **rank-based selection** as it ensured we exploited the population but still maintained diversity by not directly being fitness biased.
The mutation technique also played a role in optimization. Initially we utilized the Swapping Mutation technique but when we edited that to me Two-Opt Mutation we saw that the fitness scores kept improving for longer. Two-Opt Mutation selects two non-adjacent edges in a solution path and swaps them with each other, creating a new path with a potentially improved total cost.
Hence, as shown in the plots below, everything from the number of generations to the selection schemes for both parent and survival selection impacted optimization significantly.

- FPS and Random 2: Even though FPS is known to be good parent selection scheme since it preserves fitter individuals, when coupled with Random selection for survivor selection we see that on average there is no improvement in the fitness.

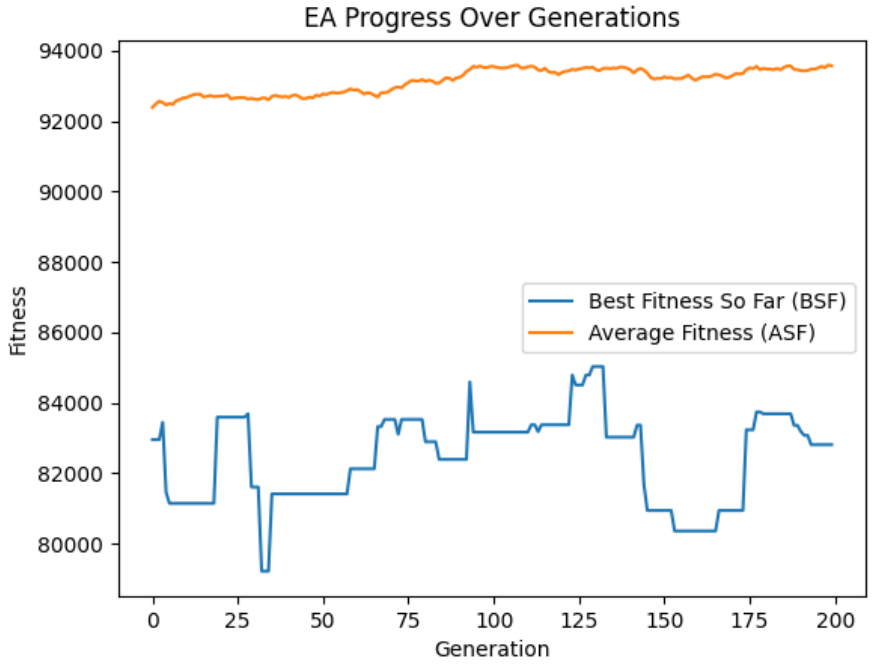


Figure 2: FPS + Random

- FPS and FPS 3: Here both parent and survivor selection favor individuals propor-

tionally to their fitness, leading to a high selection pressure but risk of premature convergence, hence when initially the population overall had very high fitness, fitness remained high throughout the generations.

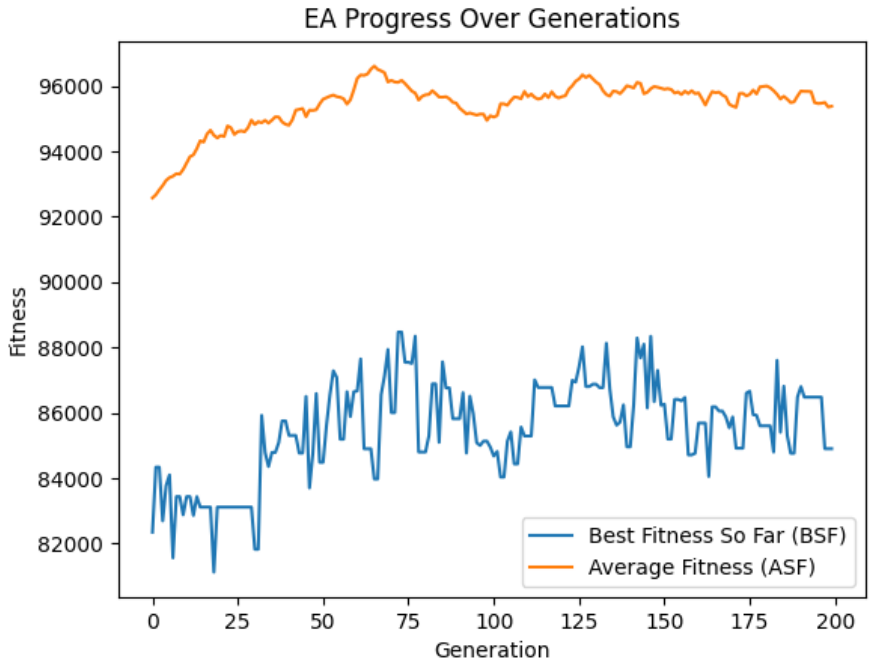


Figure 3: FPS + FPS

- FPS and Binary Tournament 4: Parents are selected based on fitness proportionate selection, while survivors are chosen via pairwise competition, balancing selection pressure and diversity. So we can see a steady decrease in the fitness value.

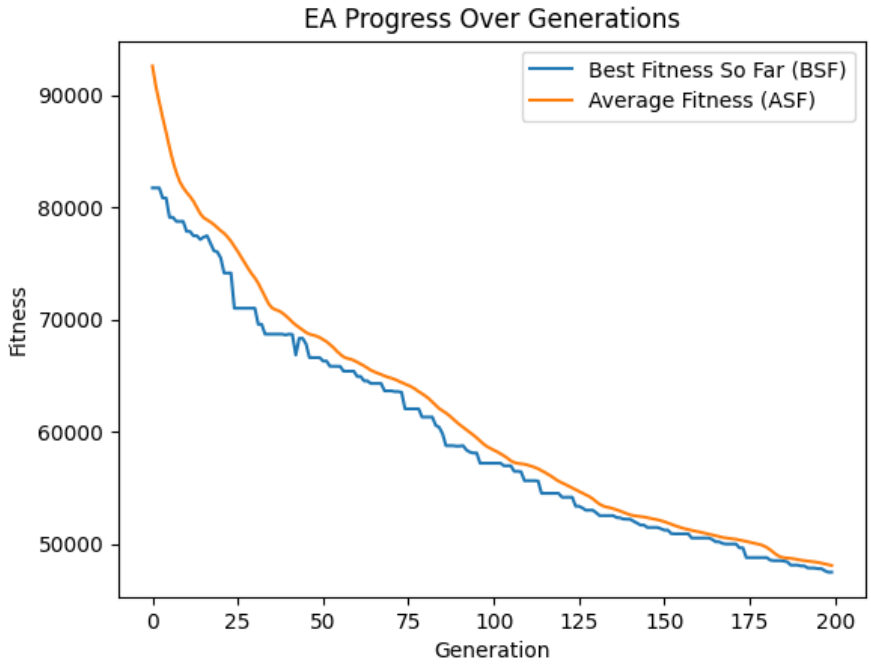


Figure 4: FPS + BTS

- FPS and Rank-Based 5: Parents are selected based on fitness proportion, but survivors are chosen based on rank, reducing the dominance of high-fitness individuals and promoting diversity. So the fitness values can be seen worsening.

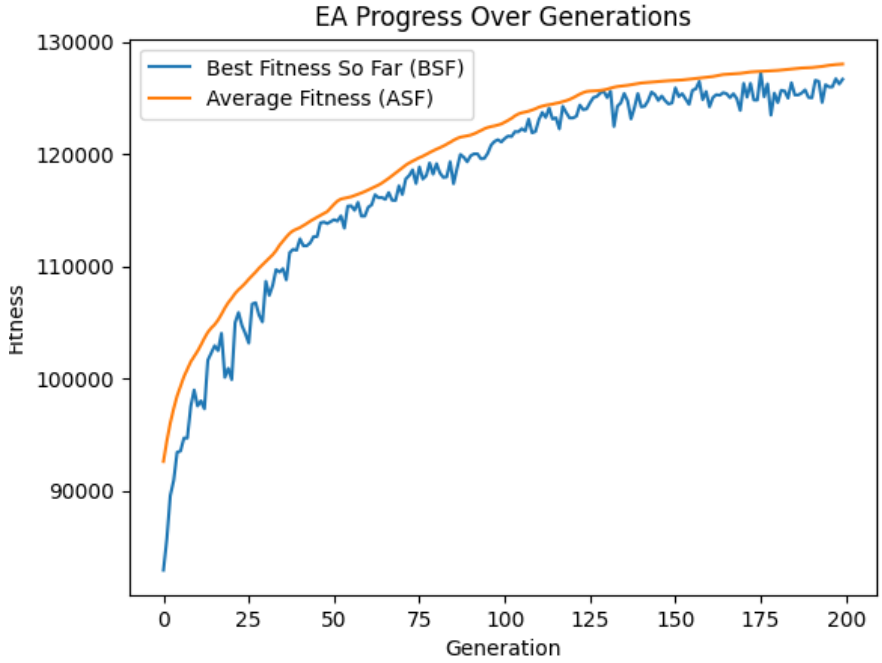


Figure 5: FPS + RBS

- FPS and Truncation 6: Parents are selected proportionally to fitness, but only

the top-ranked individuals survive, leading to strong selection pressure and fast convergence, improving the fitness values.

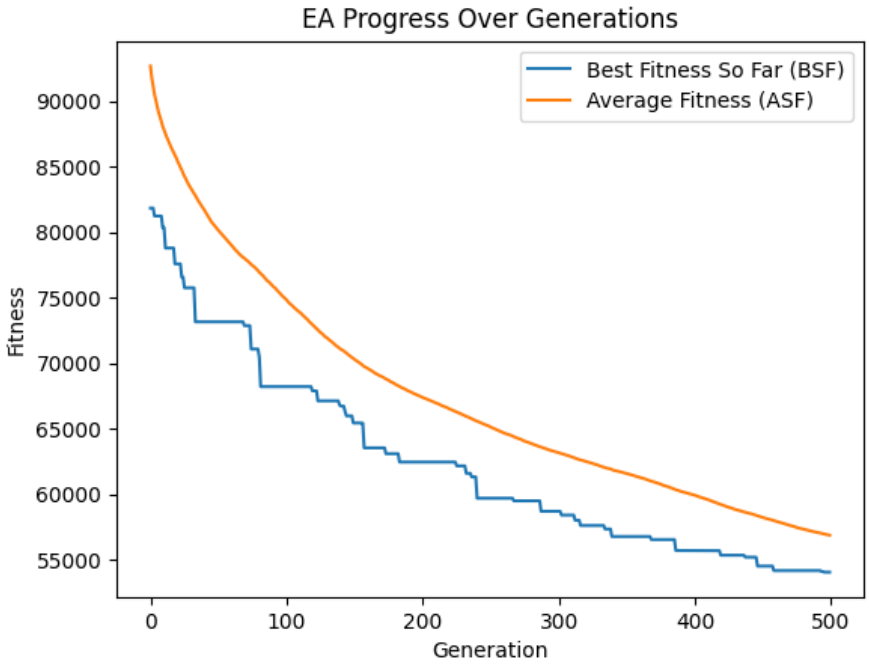


Figure 6: FPS + TS

From the above combinations of selection schemes, we realized that truncation as the survival gives the most improvement steadily in the fitness score so we decided to change the parent selection and generation number to optimize further.

- Binary Tournament and Truncation 7: Parents are selected through pairwise competition, ensuring diversity, but only the best individuals survive. We tried this combination with several different number of generations but the best it reached was within 19000.

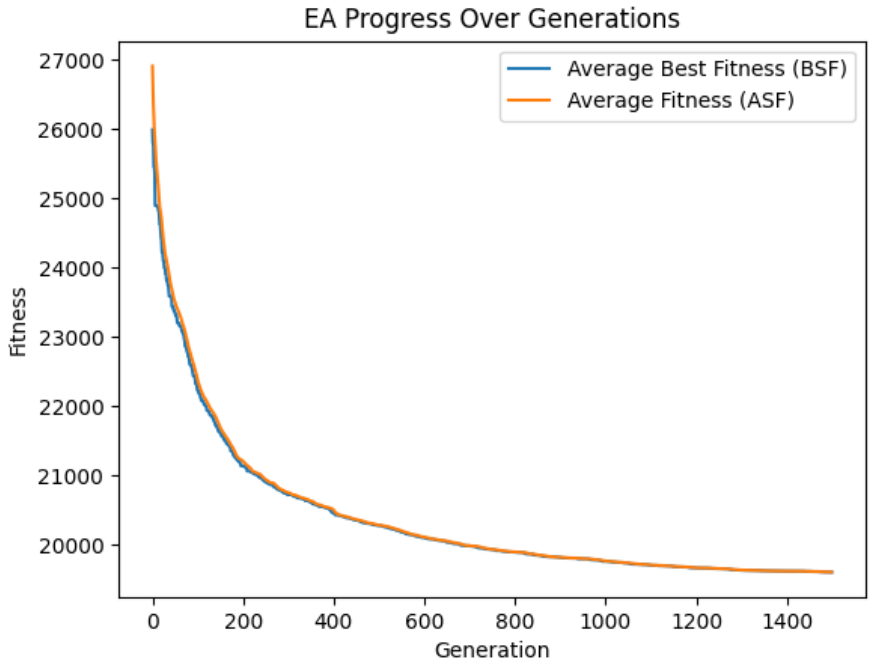


Figure 7: BTS + TS

- Rank-Based and Truncation 8: Parents are selected based on rank-based selection, ensuring diversity, but only the top-ranked individuals survive, maintaining strong selection pressure. This process utilized exploration and exploitation perfectly giving the best results for us.

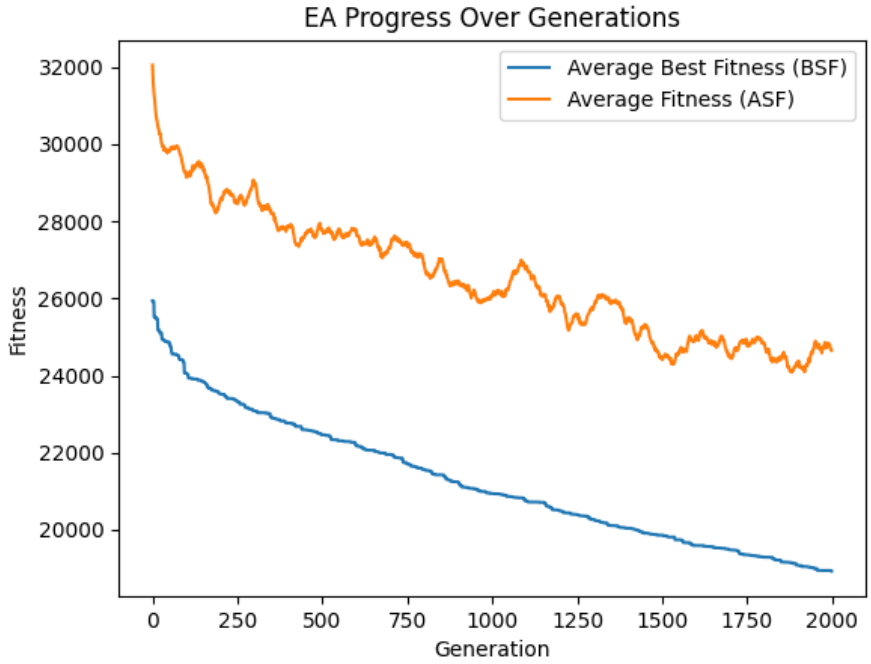


Figure 8: RBS + TS

We then continued to work on this combination of selection scheme and obtained the best result 1.

2.1.10 Conclusion

This study demonstrated the application of EA to solve TSP. While the obtained solution was competitive, further tuning of mutation rates, population size, and selection mechanisms could lead to better results. In our EA, the fitness score was still being optimized when the generations and iterations ended. However, increasing those would have taken a lot of time. Future work could incorporate hybrid metaheuristics, such as combining Genetic Algorithms with local search techniques.

2.2 Job-Shop Scheduling Problem (JSSP)

This subsection outlines the Job-Shop Scheduling Problem, an NP-hard optimization problem where the goal is to find the optimal schedule for allocating shared resources to reduce the overall completion time. The instances tested will include ft10, abz7, and la19. JSSP

2.2.1 Chromosome Representation

In the scheduling strategy used, 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 example, "1524325413" denotes Operation 1 of Job 1 followed by Operations 1 of Job 5, Job 2, Job 4, and Job 3 successively. This pattern is repeated for each operation in a job. The first instance schedules the first operation for each job as described, and the subsequent instances schedule the second operation for each job, and so on. This representation ensures the explicit capture of operation order within jobs and guarantees adherence to the machine order for each job, although it restricts the flexibility of sequencing within a job.

2.2.2 Fitness Function

The fitness function converts each chromosome into a schedule by considering the dataset and resolving overlaps between operations, ensuring efficient machine utilization by calculating idle times. The schedule is generated by determining the machine and job completion times for each operation in the chromosome. The fitness is determined by calculating the makespan, which is the maximum completion time across all machines. The final fitness value $C_(max)$ represents the total time for all jobs to be completed.

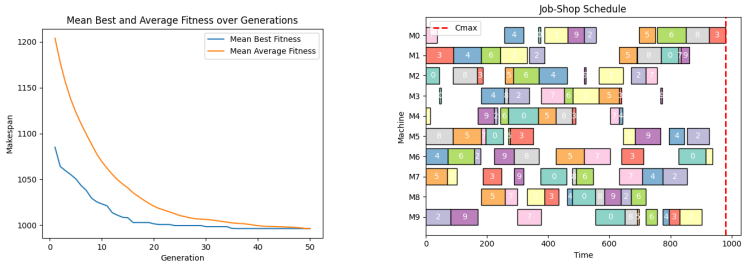
2.2.3 Result + Analysis

As we implemented TSP first, we carried forward what we learned from it to our EA implementation of JSSP. From EA we learned that our population size should be moderate and the mutation rate should be as low as possible, moreover, truncation selection is the best selection scheme for survivor selection. After several combinations of survivor selections we chose Truncation Selection again. The overall optimal solutions over all three datasets are calculated using the following settings:

- Population Size: 70
- Offspring Size: 50
- Mutation Rate: 0.3
- Number of Generations: 50
- Parent Selection Scheme: Fitness Proportional Selection
- Survivor Selection Scheme: Truncation Selection

The best results across different datasets are displayed below:

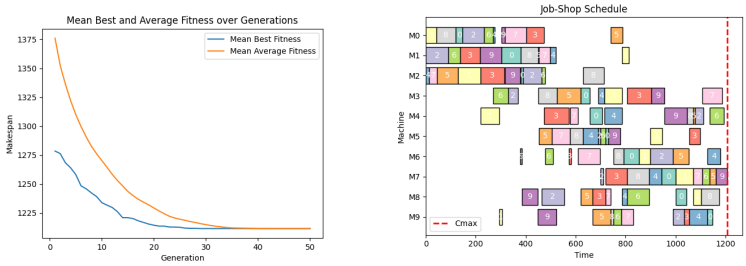
- Dataset 1 (la19):
  Optimal Value: Achieved makespan of **981 mins**.
  The solution associated with this value is plotted in the corresponding figure.



(a) Best Solution's Fitness (b) The Best Solution

Figure 9: la19 with FPS + Truncation

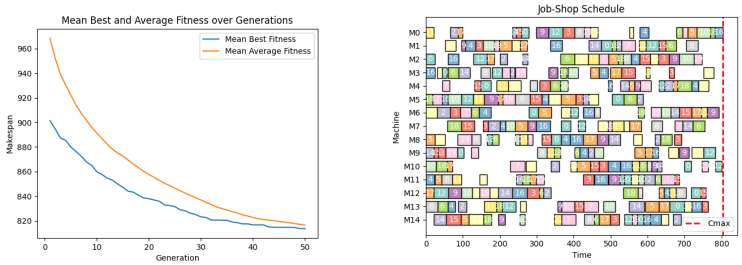
- Dataset 2 (ft10):
  Optimal Value: Achieved makespan of **1208 mins**.
  The solution associated with this value is displayed in the corresponding figure.



(a) Best Solution's Fitness (b) The Best Solution

Figure 10: ft10 with FPS + Truncation

- Dataset 3 (abz7):
  Optimal Value: Achieved makespan of **803 mins**.
  The solution associated with this value is displayed in the corresponding figure.

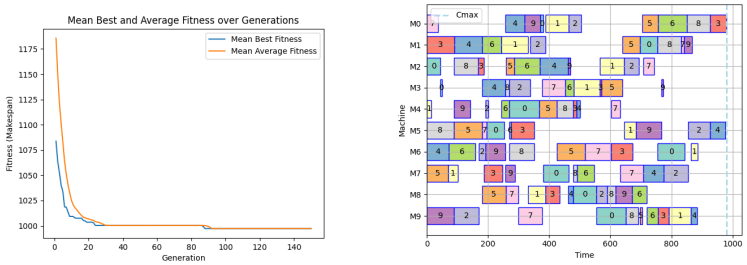


(a) Best Solution's Fitness (b) The Best Solution

Figure 11: abz7 with FPS + Truncation

The following plots show the solutions for each dataset with different parent selection schemes:

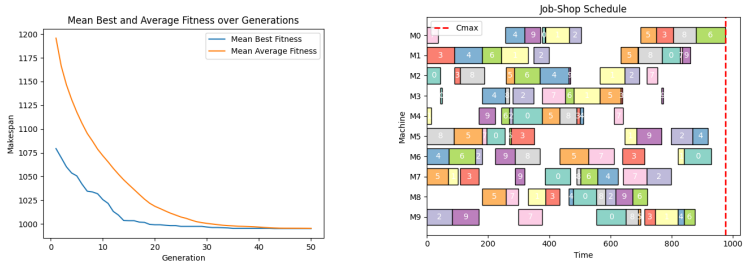
- Dataset 1 (la19):
  - RBS + Truncation:
    Optimal Value: Achieved makespan of **981 mins**.
    The solution associated with this value is plotted in the corresponding figure.



(a) (b)

Figure 12: la19 with RBS + Truncation

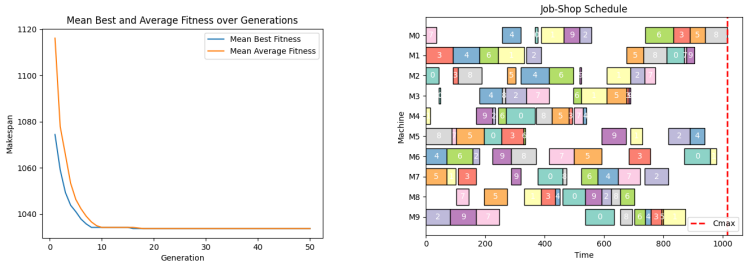
  - Random + Truncation:
    Optimal Value: Achieved makespan of **975 mins**.
    The solution associated with this value is plotted in the corresponding figure.



(a) (b)

Figure 13: la19 with Random + Truncation

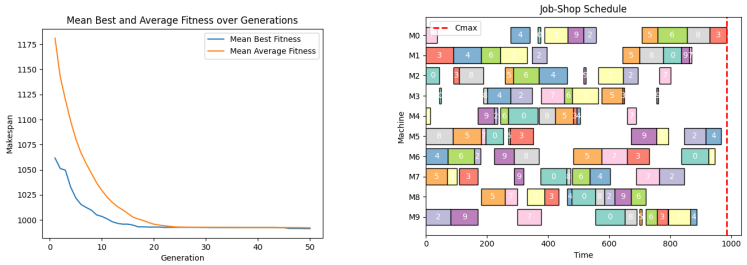
– Truncation + Truncation:
Optimal Value: Achieved makespan of **1015 mins**.
The solution associated with this value is plotted in the corresponding figure.



(a) (b)

Figure 14: la19 with Truncation + Truncation

– BTS + Truncation:
Optimal Value: Achieved makespan of **984 mins**.
The solution associated with this value is plotted in the corresponding figure.

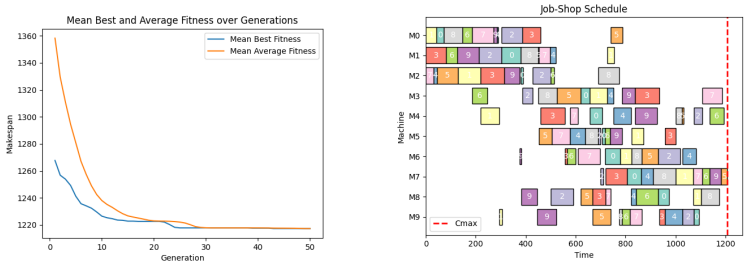


(a) (b)

Figure 15: la19 with BTS + Truncation

- Dataset 2 (ft10):

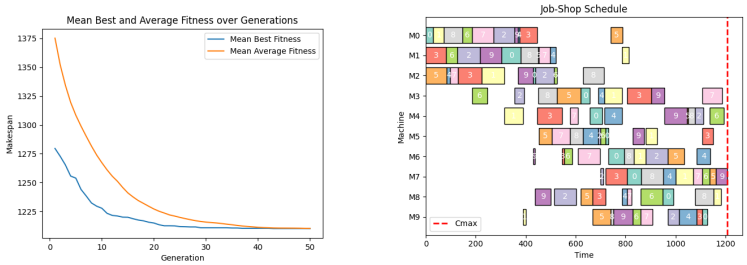
– RBS + Truncation:
Optimal Value: Achieved makespan of **1208 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 16: ft10 with RBS + Truncation

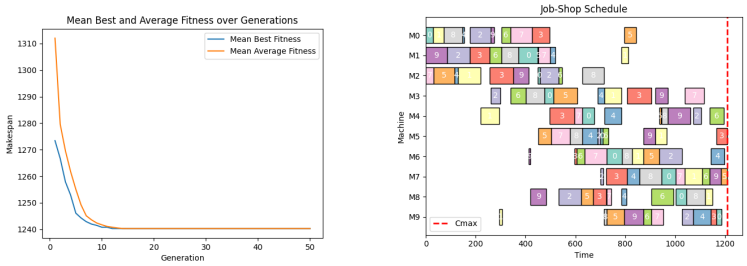
– Random + Truncation:
Optimal Value: Achieved makespan of **1210 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 17: ft10 with Random + Truncation

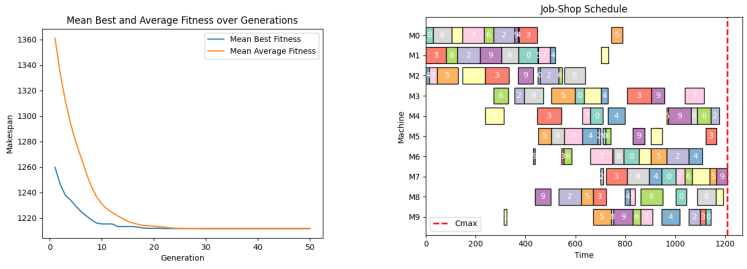
– Truncation + Truncation:
Optimal Value: Achieved makespan of **1210 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 18: ft10 with Truncation + Truncation

– BTS + Truncation:
Optimal Value: Achieved makespan of **1210 mins**.
The solution associated with this value is displayed in the corresponding figure.

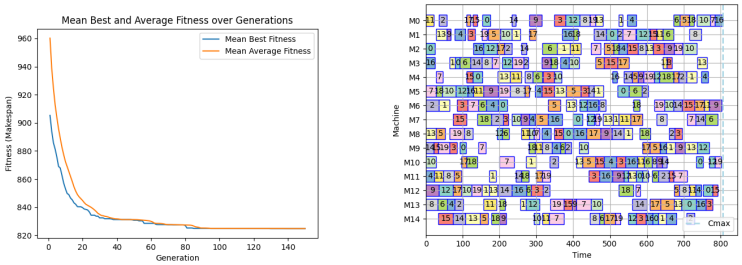


(a) (b)

Figure 19: ft10 with BTS + Truncation

- Dataset 3 (abz7):

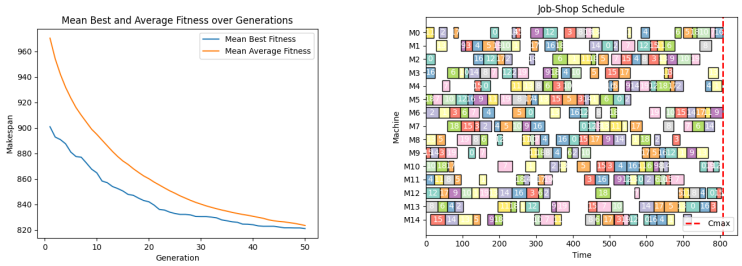
– RBS + Truncation:
Optimal Value: Achieved makespan of **806 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 20: abz7 with RBS + Truncation

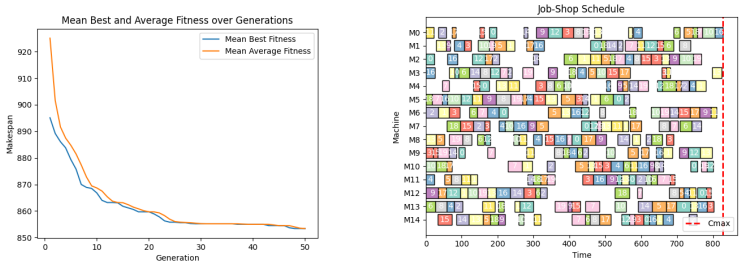
– Random + Truncation:
Optimal Value: Achieved makespan of **807 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 21: abz7 with Random + Truncation

– Truncation + Truncation:
Optimal Value: Achieved makespan of **827 mins**.
The solution associated with this value is displayed in the corresponding figure.

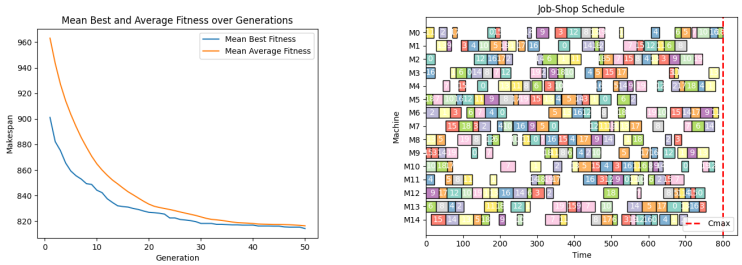


(a) (b)

Figure 22: abz7 with Truncation + Truncation

– BTS + Truncation:

Optimal Value: Achieved makespan of **800 mins**.
The solution associated with this value is displayed in the corresponding figure.



(a) (b)

Figure 23: abz7 with BTS + Truncation

2.2.4 Overall Analysis

In our analysis of the Job-Shop Scheduling Problem (JSSP) implementation, we found that the model consistently produced competitive schedules across various benchmark instances, albeit with limited evolution over successive generations. Our most favorable outcomes were achieved by employing either Fitness Proportional Selection (FPS) or Binary Tournament Selection (BTS) for parent selection, in combination with truncation survivor selection. In our experiments, using a population size of 100, generating 80 offspring per generation, and setting the mutation rate to 0.3 consistently yielded near-optimal performance.

However, we observed that integrating FPS with random selection schemes led to unexpected increases in fitness values rather than the desired decrease, proving to be the worst performing approach in our trials. Similarly, while random selection methods introduced some degree of diversity, they also resulted in significant fluctuations in fitness levels without any substantial overall improvement. Based on these observations, our analysis recommends adopting either FPS or BTS for parent selection, complemented by truncation-based survivor selection, as the most effective strategy for solving the JSSP. This approach not only ensures steady convergence toward optimal schedules but also maintains sufficient diversity to avoid premature stagnation.

3 Evolutionary Art

3.1 Chromosome Representation

In our approach, a chromosome represents an image with specific dimensions (height and width). Each chromosome is stored as a dictionary containing its height, width, image, array form, and fitness score.

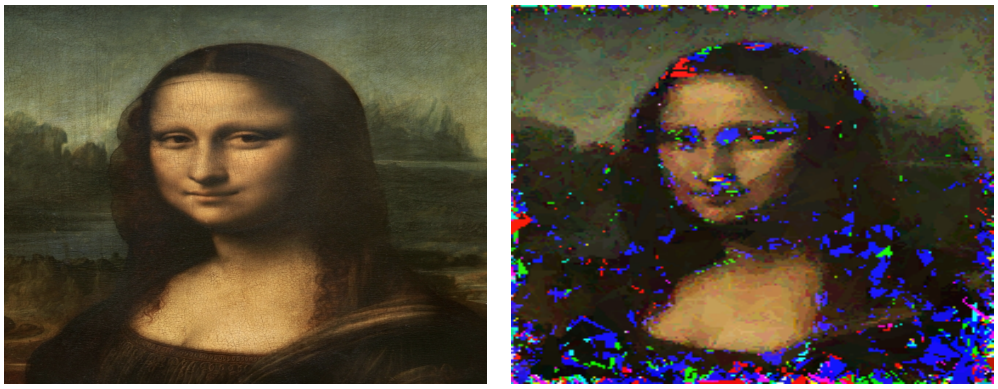
When we create a chromosome, we generate a random image, transform it into an array, and then evaluate its fitness. The fitness score helps determine how well the image aligns with our target image, guiding the evolutionary process. This structure allows us to apply genetic operations like selection, mutation, and crossover effectively, gradually evolving the images toward optimal results.

3.2 Fitness Function

Our fitness function is designed to measure how closely a generated image resembles the target image. To achieve this, we compare the pixel values of both images. Specifically, we calculate the sum of absolute differences between corresponding pixels in the target image and the generated image. The lower this sum, the closer our generated image is to the desired outcome. This approach ensures that our evolutionary algorithm selects and evolves images that progressively become more similar to the target over generations.

3.3 Optimal Results Achieved

Through experimentation, we found that lowering the mutation rate to 0.05 and increasing the population size to 200 significantly improved the results. Increasing the number of generations to 100000 also helped us improve our resulting image. These adjustments allowed the algorithm to explore better solutions while maintaining stability in the evolution process.



(a) Target Image (b) Recreated Image

Figure 24: Comparison between the target image and the best-generated image.

3.4 Results Across Generations

To analyze the progression of our evolutionary algorithm, we captured images at different stages of evolution. The model was trained for 100,000 generations, but for clarity, we showcase a subset of images taken at intervals. This helps visualize how the population gradually improves and converges towards the target image.

(a) Gen 0 (b) Gen 500 (c) Gen 2000 (d) Gen 5000 (e) Gen 10000

(f) Gen 20000 (g) Gen 30000 (h) Gen 50000 (i) Gen 70000 (j) Gen 999990

Figure 25: Evolution of the image over multiple generations.

3.5 Overall Analysis

Our approach to image recreation using a genetic algorithm involved several iterative refinements. Instead of starting with a completely random population, we initialized individuals by assigning average colors to different regions of the image. This provided a better starting point compared to full randomness, allowing the algorithm to converge more effectively. We used elitism to preserve the best candidates across generations, ensuring that high-quality solutions were not lost. Tournament selection with tournament size of 8 was employed to balance exploration and exploitation, selecting individuals based on competitive performance rather than pure fitness ranking.

Initially, we started with a population size of 100, which yielded decent results. However, after experimentation, we found that increasing the population to 200 improved the recreated image's quality by enhancing diversity. Reducing the population size too much led to poor diversity and suboptimal results.

We experimented by running the model for 100,000 generations, but the process was extremely slow. After analyzing the results, we found that around 30000 generations already produced visually impressive results, and running it for 100,000 generations did not significantly improve the output. This suggests that 30,000 generations is a good balance between accuracy and computational efficiency.

A key addition to our approach was the Hall of Fame mechanism, which stored the top five best-performing individuals across generations. This prevented the loss of high-quality solutions and periodically reintroduced them to the population, reinforcing strong genetic traits and improving overall convergence.

Through our experiments, we found that the combination of a population size of 200, elitism, tournament selection, Hall of Fame integration, and 30,000 generations produced the best results while maintaining computational feasibility.