Habib University

CS 451 - Computational Intelligence

Spring' 2025

Assignment 1 – Evolutionary Algorithm

Ahsan Azeemi, Daniyal Rahim

Q1 - Evolutionary Algorithms

A. Travel Salesman Problem

Approach:

Initialization - TSP is initialized by selecting the top `**population_size**` from randomly `**1000**` generated chromosomes after sorting them based on the fitness function.

Fitness Function - Simple Euclidean distance from City 1 to last city (194 cities from the dataset) and then the distance from last city to City 1 again since the route must be circular.

Crossover - We used Partially Mapped Crossover (PMX) for our TSP problem because it helps preserve the relative order of cities while ensuring valid routes.

How it works:
Select a segment from one parent and copy it to the child.
Fill the remaining positions using genes from the second parent, avoiding duplicates by mapping conflicts.
Example:

Parent 1: [3, 5, 8, 2, 6, 1, 4, 7]
Parent 2: [4, 2, 7, 5, 3, 8, 6, 1]
If we copy [8, 2, 6, 1] (indexes 2-5), we map the rest carefully to avoid repetition.
This ensures valid and optimized TSP routes while inheriting good traits from both parents.

Mutation - We used a 2-opt mutation for our TSP problem because it helps refine routes by reducing unnecessary travel distance.

How it works:
Select two points in the route randomly.
Reverse the segment between them to explore a potentially shorter path.
Example:

Before: [1, 3, 5, 4, 2, 6]
Swapping between 3rd and 5th positions:
After: [1, 3, 2, 4, 5, 6]
This mutation helps avoid local optima and improves the overall tour in the TSP.

Parents and Survivor Selection Scheme - For the sake of simplicity, only 5 of the given scenarios have been tested and will be discussed in the analysis section.

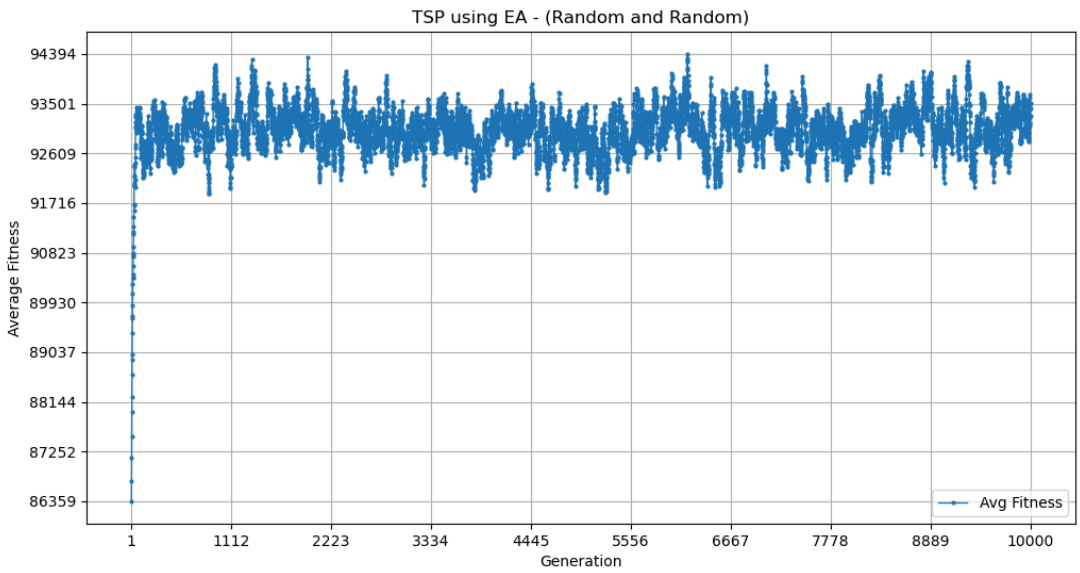
- Fitness Proportional Selection
- Rank based Selection
- Binary Tournament
- Truncation
- Random

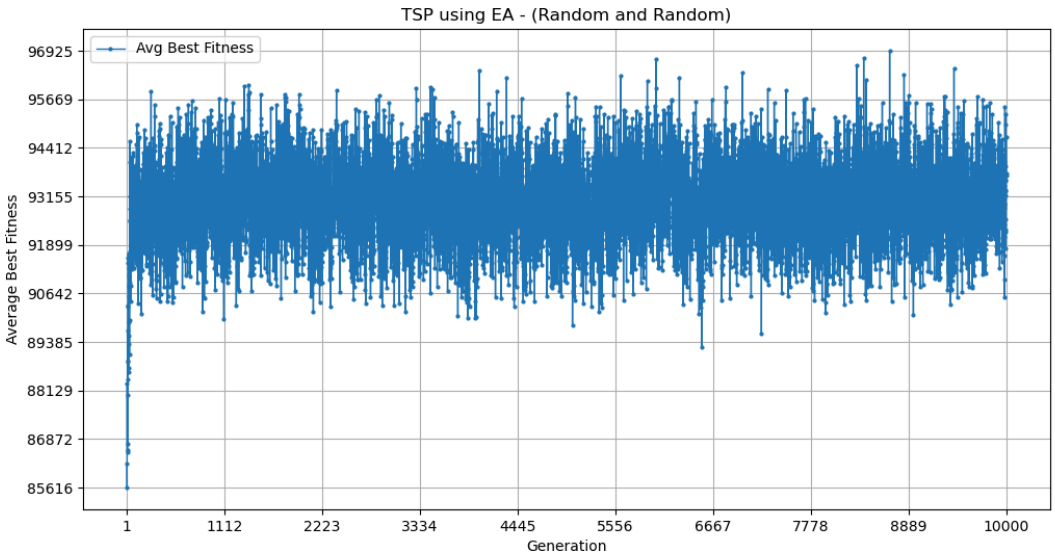
Parameters:

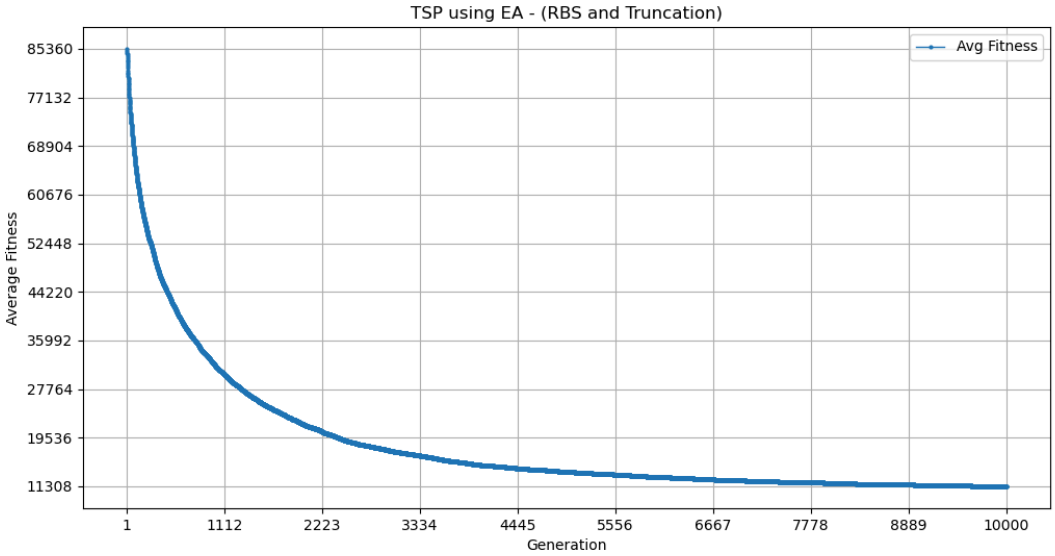
- Population size = 30
- No of offsprings = 10
- No of generations = 10000
- Mutation Rate = 0.5
- Iterations = 10

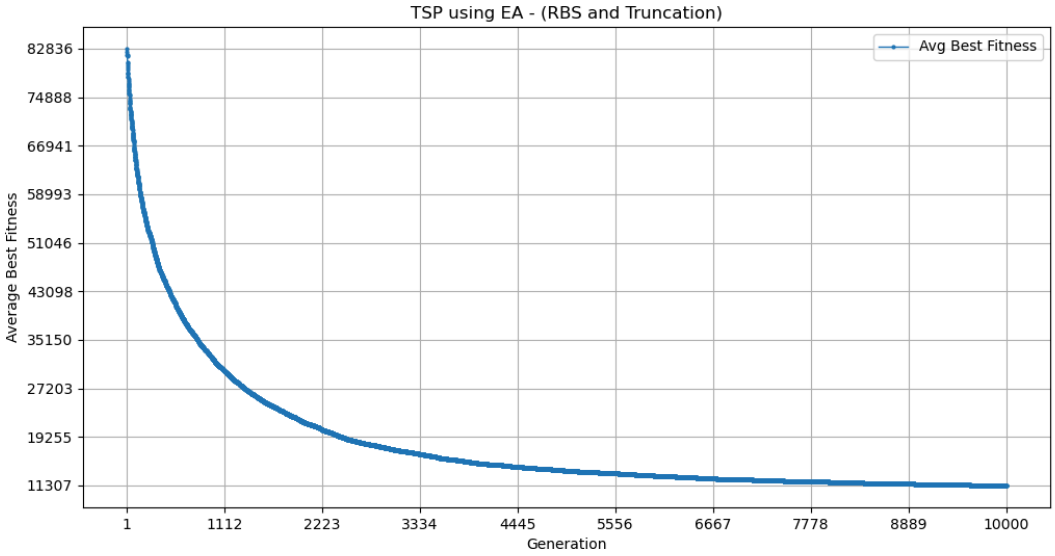
Modifications: We had to increase the number of generations to get under 15,000.

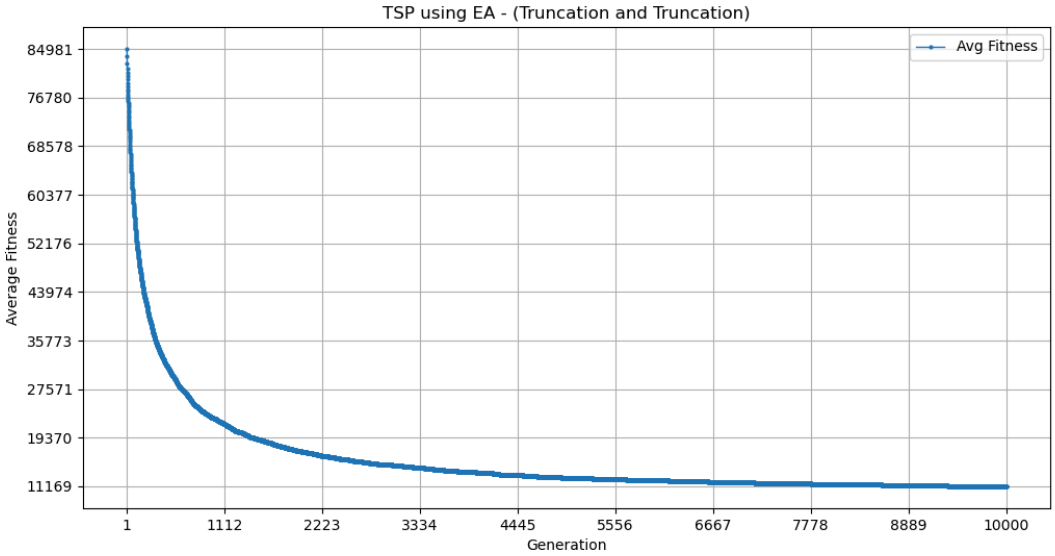
Results:

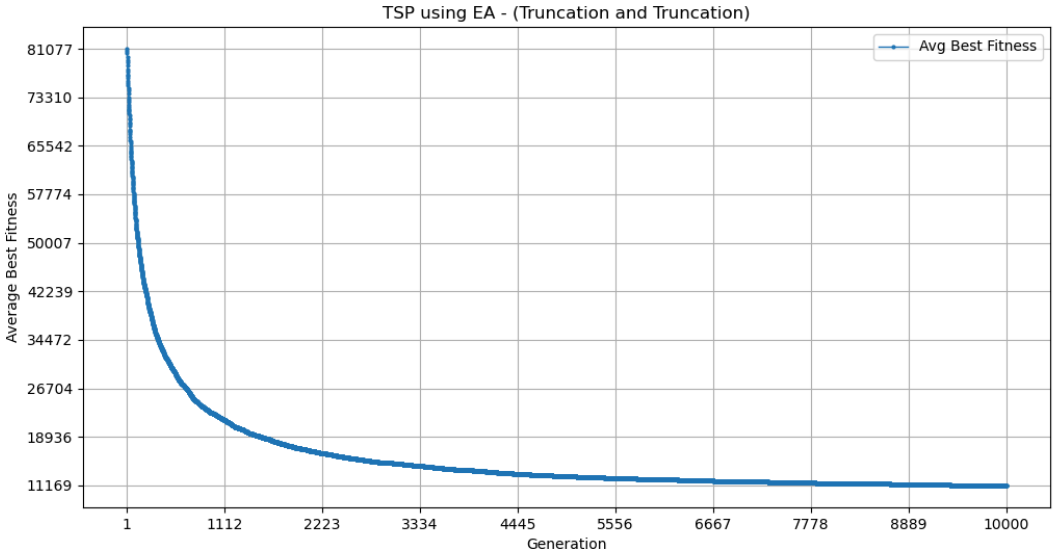


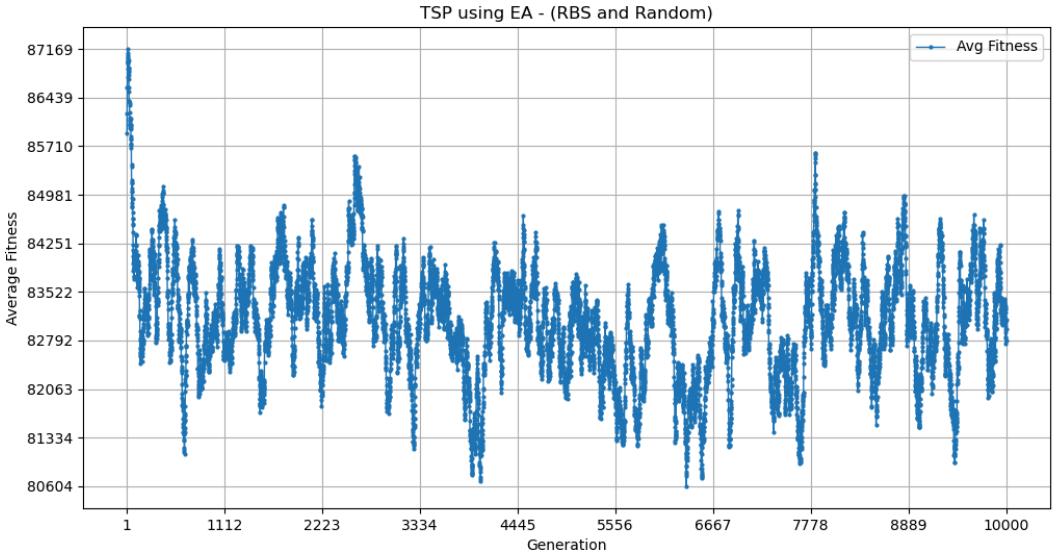

TSP using EA - (Random and Random)
Avg Best Fitness
Average Best Fitness
96925
95669
94412
93155
91899
90642
89385
88129
86872
85616
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation

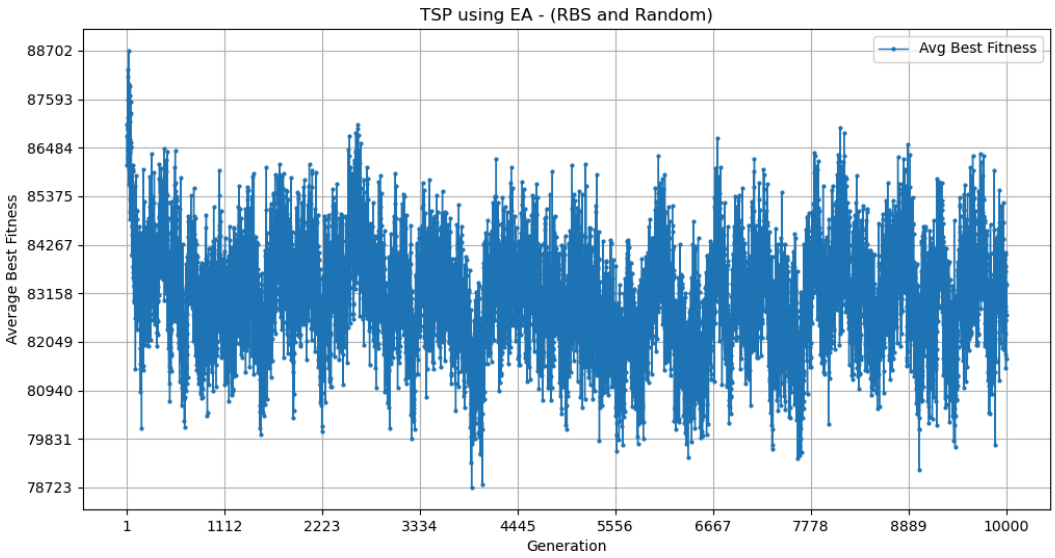

TSP using EA - (RBS and Truncation)
Avg Fitness
Average Fitness
85360
77132
68904
60676
52448
44220
35992
27764
19536
11308
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation

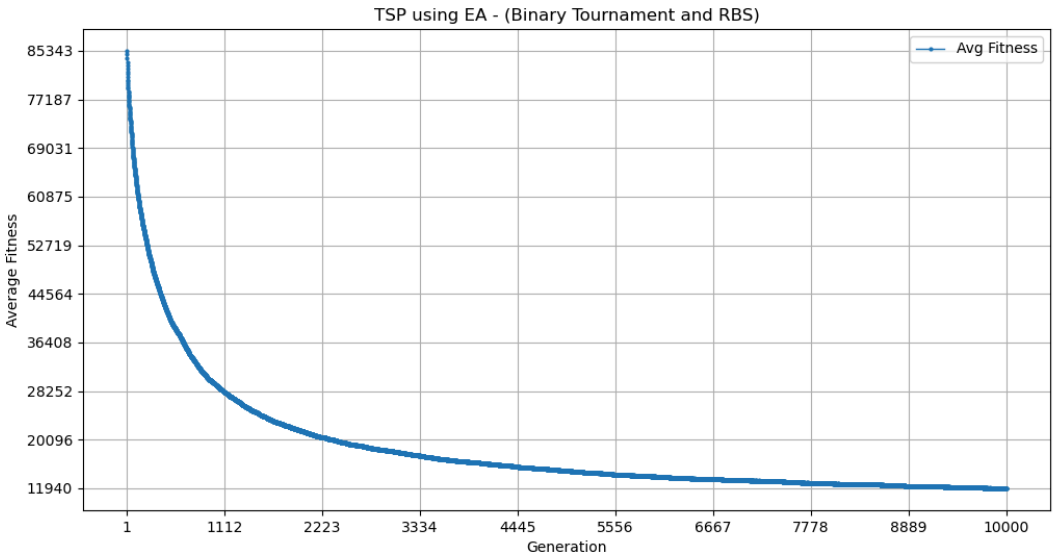

TSP using EA - (RBS and Truncation)
Avg Best Fitness
82836
74888
66941
58993
51046
43098
35150
27203
19255
11307
Average Best Fitness
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation

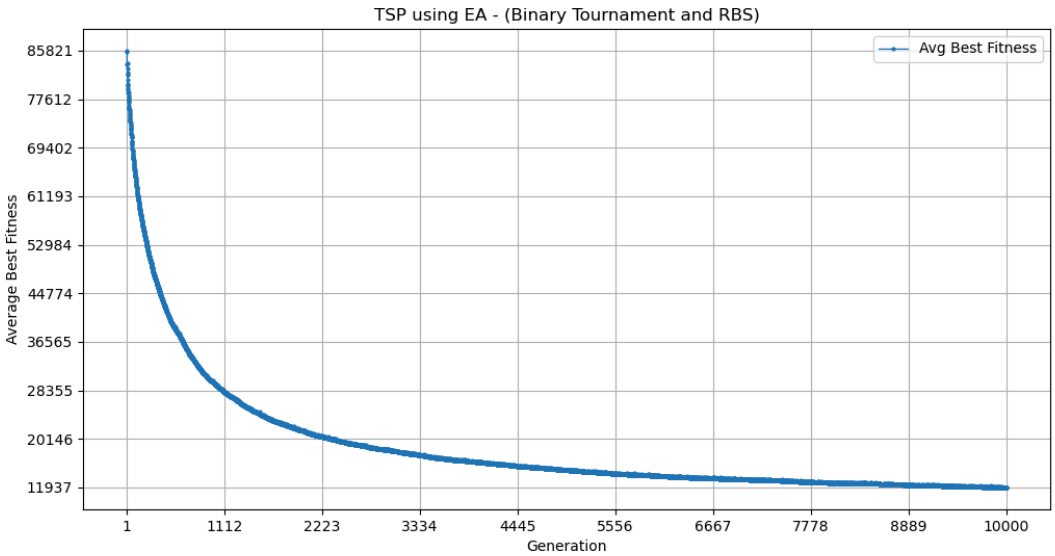

TSP using EA - (Truncation and Truncation)
Avg Fitness
84981
76780
68578
60377
52176
43974
35773
27571
19370
11169
Average Fitness
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation

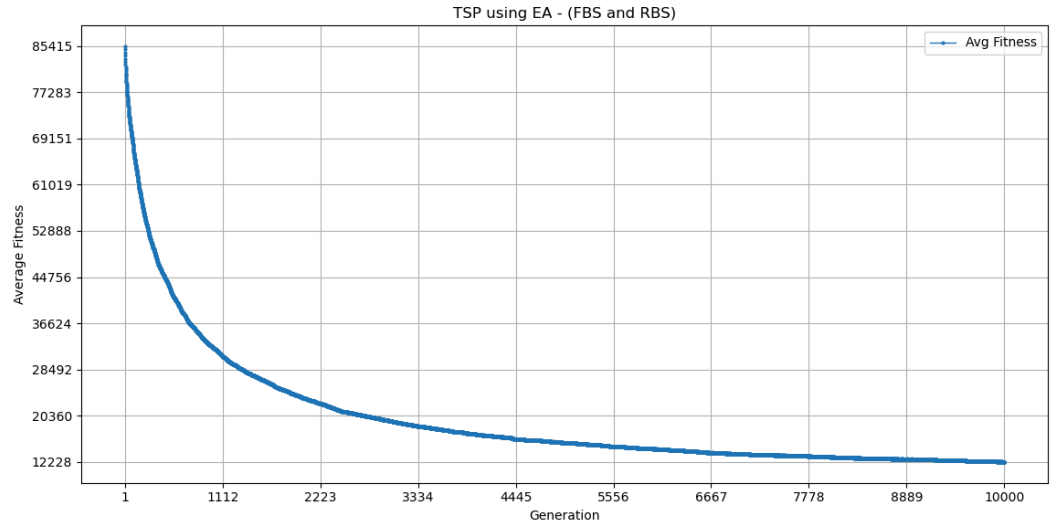

TSP using EA - (Truncation and Truncation)
Avg Best Fitness
Average Best Fitness
81077
73310
65542
57774
50007
42239
34472
26704
18936
11169
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation

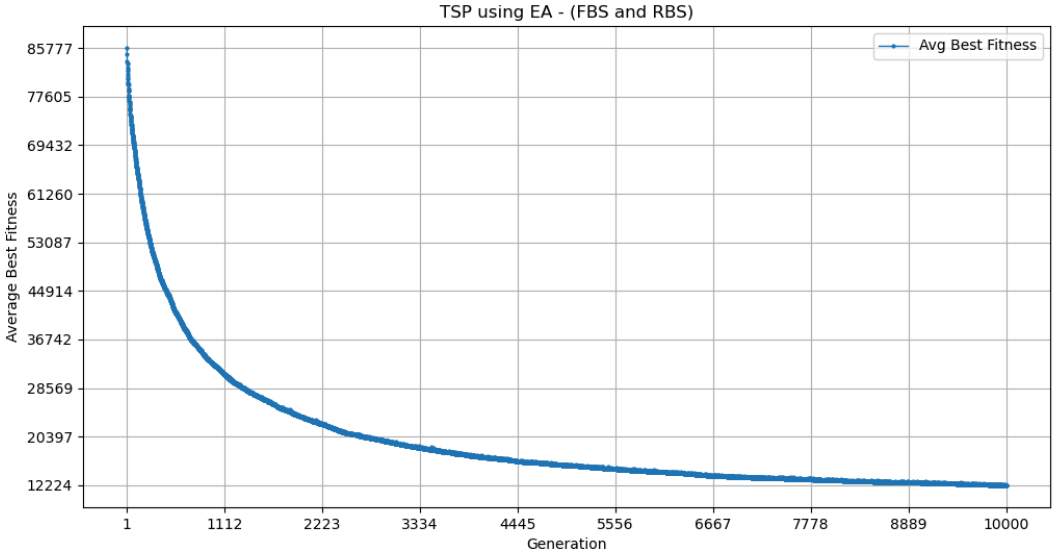

TSP using EA - (RBS and Random)
Avg Fitness
Average Fitness
87169
86439
85710
84981
84251
83522
82792
82063
81334
80604
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation


TSP using EA - (RBS and Random)
Avg Best Fitness
Average Best Fitness
88702
87593
86484
85375
84267
83158
82049
80940
79831
78723
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation


TSP using EA - (Binary Tournament and RBS)
Avg Fitness
Average Fitness
85343
77187
69031
60875
52719
44564
36408
28252
20096
11940
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation


TSP using EA - (Binary Tournament and RBS)
Avg Best Fitness
Average Best Fitness
85821
77612
69402
61193
52984
44774
36565
28355
20146
11937
1
1112
2223
3334
4445
5556
6667
7778
8889
10000
Generation





Analysis:

When solving the Traveling Salesman Problem (TSP), the combination of Truncation and Truncation selection performed best due to its strong balance of exploitation and exploration. Truncation ensures that only the top-performing individuals are selected for reproduction, which accelerates convergence toward optimal solutions by preserving the best genetic material. The crossover operation played a crucial role in combining high-quality solutions to form even better offspring, while mutation maintained diversity and prevented premature convergence by

introducing slight variations. This approach leveraged the strengths of both exploitation—by consistently refining elite solutions—and exploration—by introducing controlled randomness to avoid local optima. When plotting the results, we observed that this scheme consistently found shorter tour lengths with faster convergence compared to other selection strategies. The best average distance for this scheme was **11160**. A number of **9980** was also achieved but it was run for 50000 generations for 3 hours and after trying it running again with graph plotting causing memory errors for python program.

On the other hand, the combination of Random and Random selection performed the worst due to its lack of direction in guiding the evolution of solutions. Since individuals were selected purely at random, there was no guarantee that strong candidates would propagate, leading to inefficient searches and slower convergence. While crossover still helped create new solutions, and mutation added some diversity, the absence of a structured selection process meant that good solutions were frequently lost, and poor ones were often retained. This led to an imbalance where exploration was high but exploitation was nearly absent, preventing the algorithm from consistently refining better solutions. When visualizing the results, the plots revealed erratic performance, with solutions fluctuating unpredictably and struggling to improve over successive generations.

B. Job-shop Scheduling

Approach:

Initialization - The JSSP population is initialized by generating pop_size chromosomes, where each chromosome is a random permutation of job operations. The fitness of each chromosome is computed using the makespan (total time taken to complete all jobs).

Each chromosome consists of a sequence of job-operation pairs, ensuring all job operations are scheduled exactly once.

Fitness Function -

The chromosome is decoded into a schedule based on job precedence and machine availability.

Each operation is assigned a start time based on when both the job and the machine are available.

The objective is to minimize the makespan, meaning shorter schedules are favored.

Crossover (Partially Mapped Crossover) -

Selected two crossover points in the chromosome.

Copied the segment from Parent 1 into Child 1 and from Parent 2 into Child 2.

Mapped the remaining operations to avoid duplicates while preserving order.

This ensured valid schedules while inheriting traits from both parents.

Mutation (Swap Mutation) -

Two random operations within the chromosome are swapped with a small probability (mutation_rate).

This prevented premature convergence by ensuring exploration of different schedules.

Parent Selection Strategies

FPS, Binary Tournament, Truncation Selection, Random Selection

Survivor Selection Strategies

Random Selection, Truncation Selection, RBS

Parameters and Experimentation

Population size: 100

Generations: 50

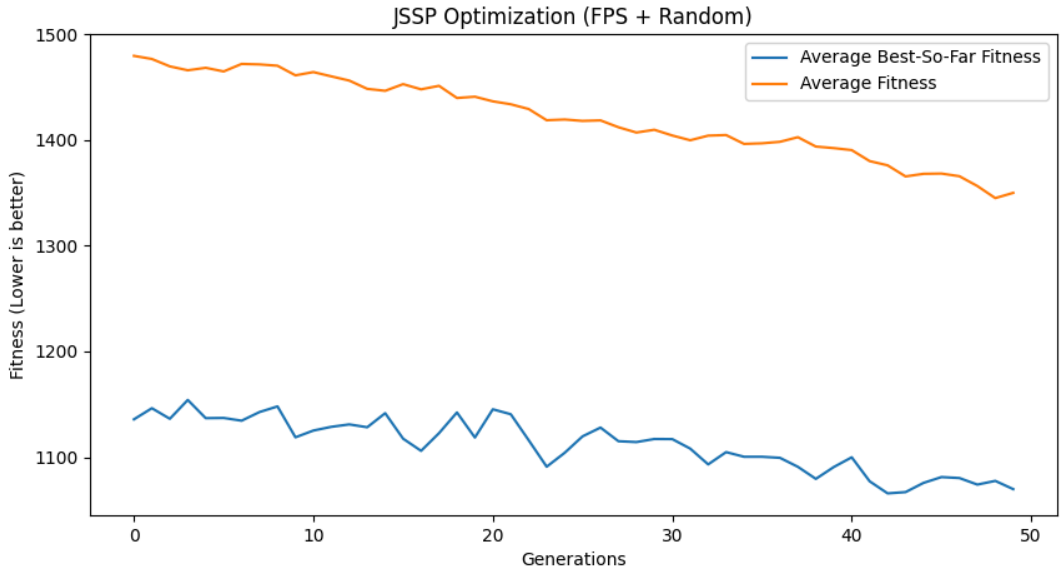
Iterations: 10

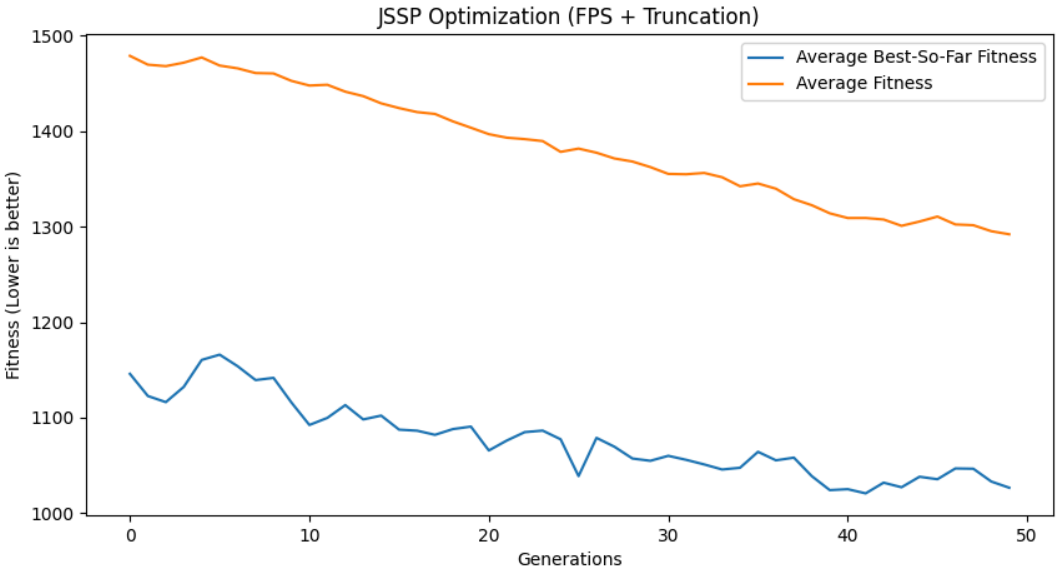
Mutation Rate: 0.1

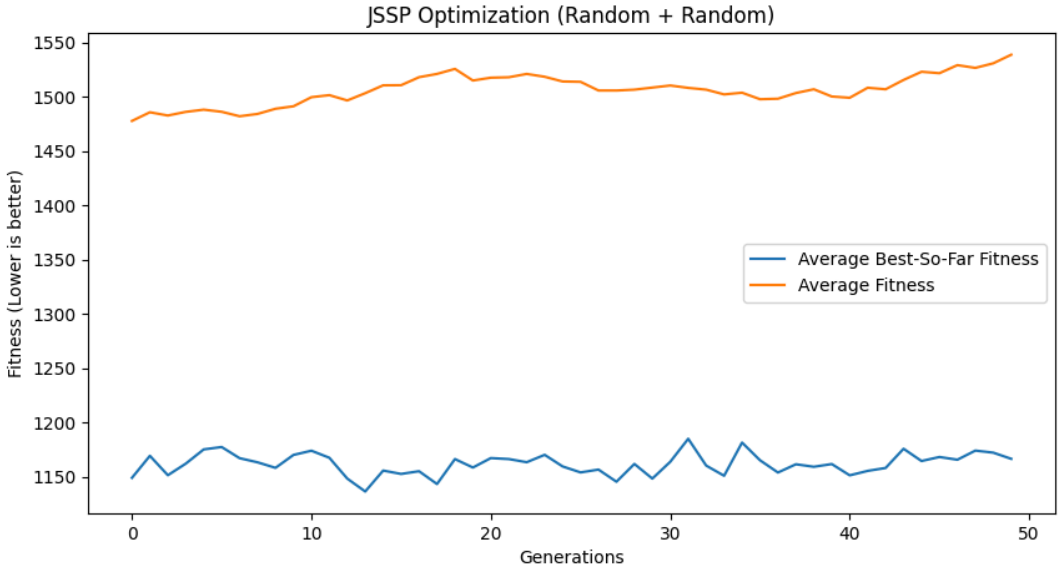
Various different Selection + Survival Combinations were tested for each problem and are shown and analyzed in the plots below

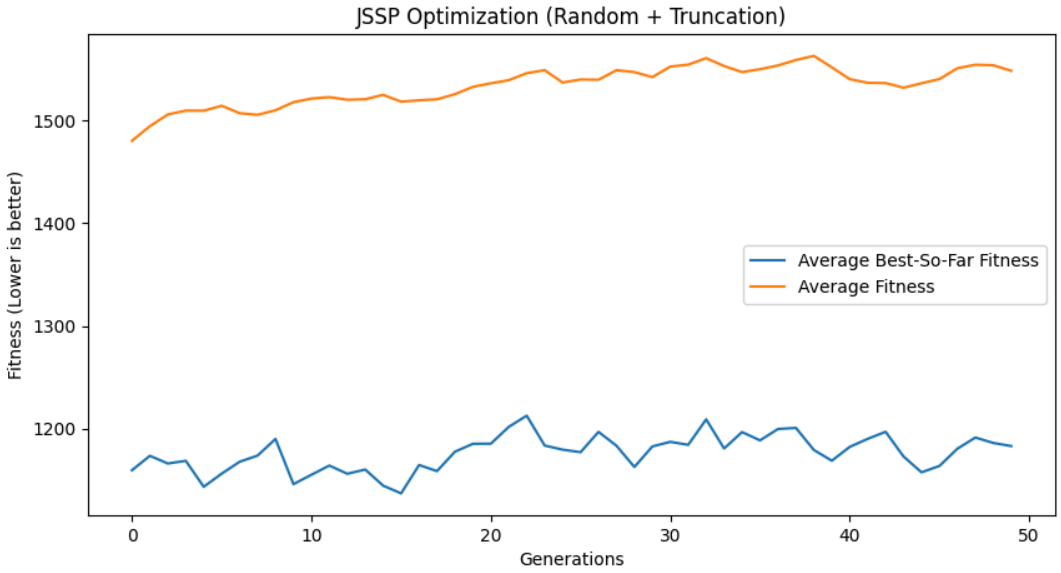
Results:

f110 Dataset





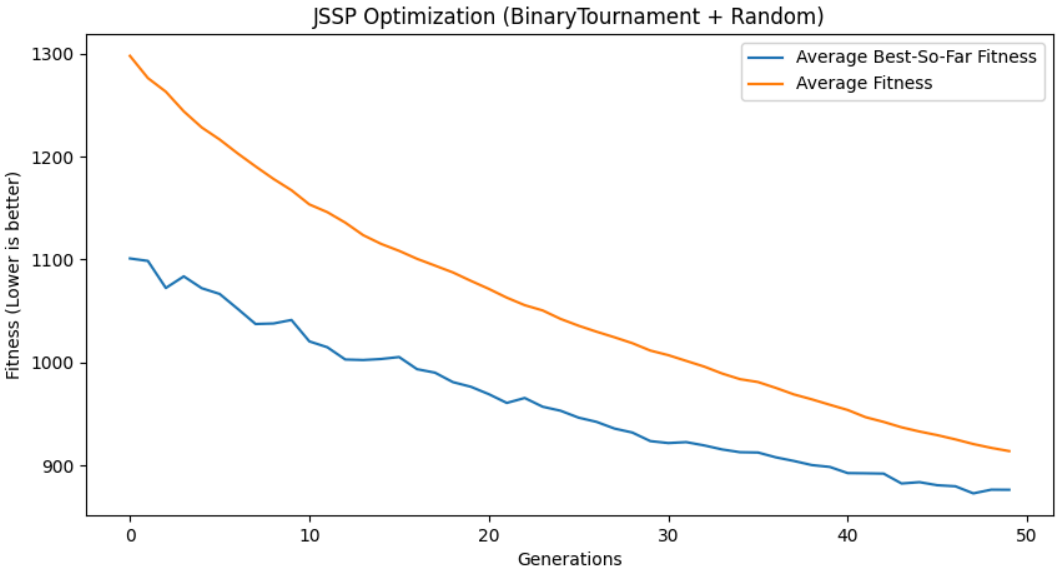


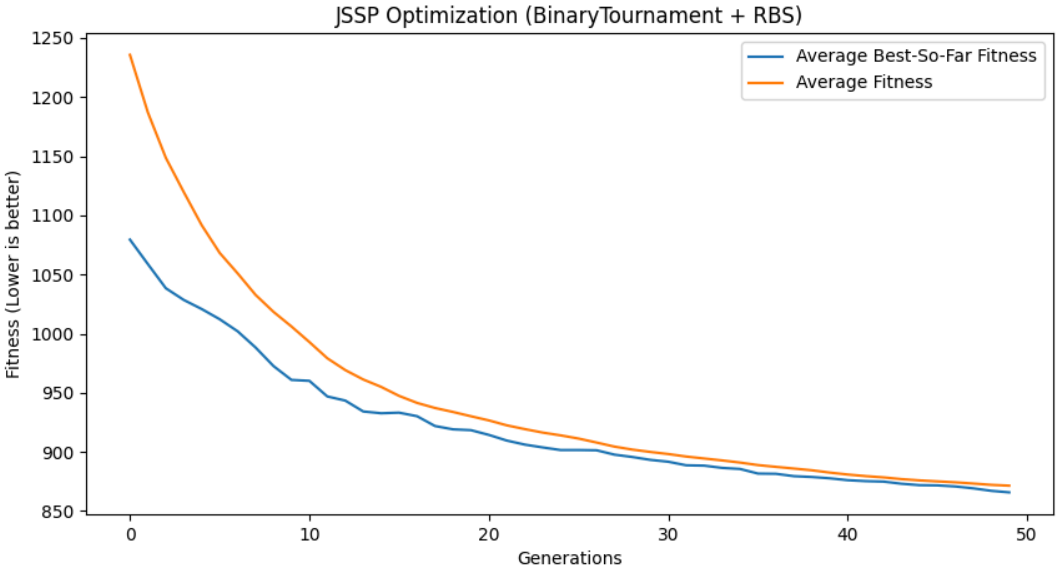


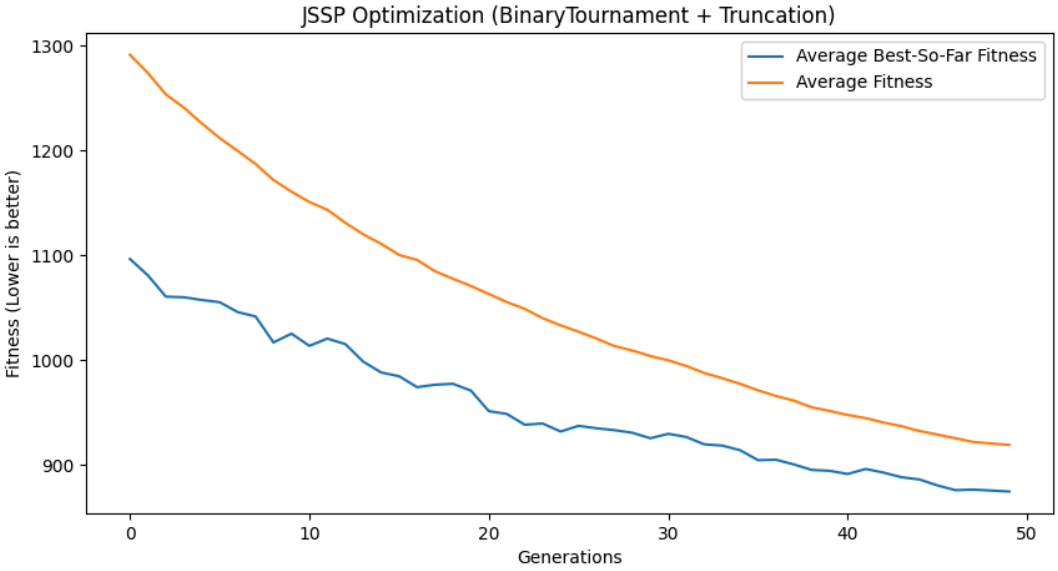
Analysis:

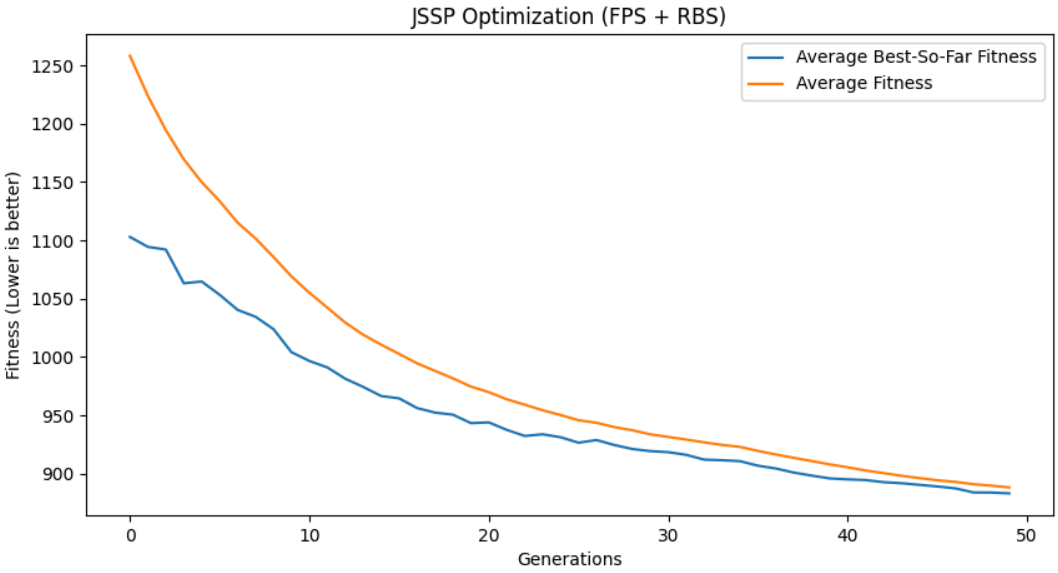
Optimal Value achieved was 1031. The FPS + Truncation scheme performed best because fitness-proportionate selection favors high-quality parents, while truncation selection ensures only the top solutions survive, leading to steady improvements. On the other hand, the Random + Random scheme performed the worst because both parents and survivors were chosen randomly, preventing meaningful selection pressure and leading to poor optimization.

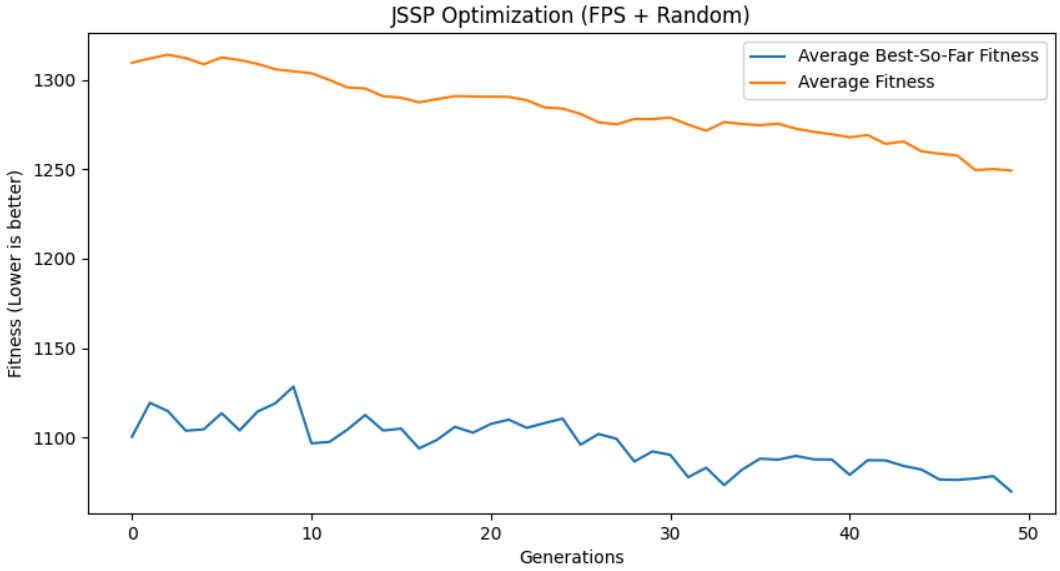
abz7 Dataset

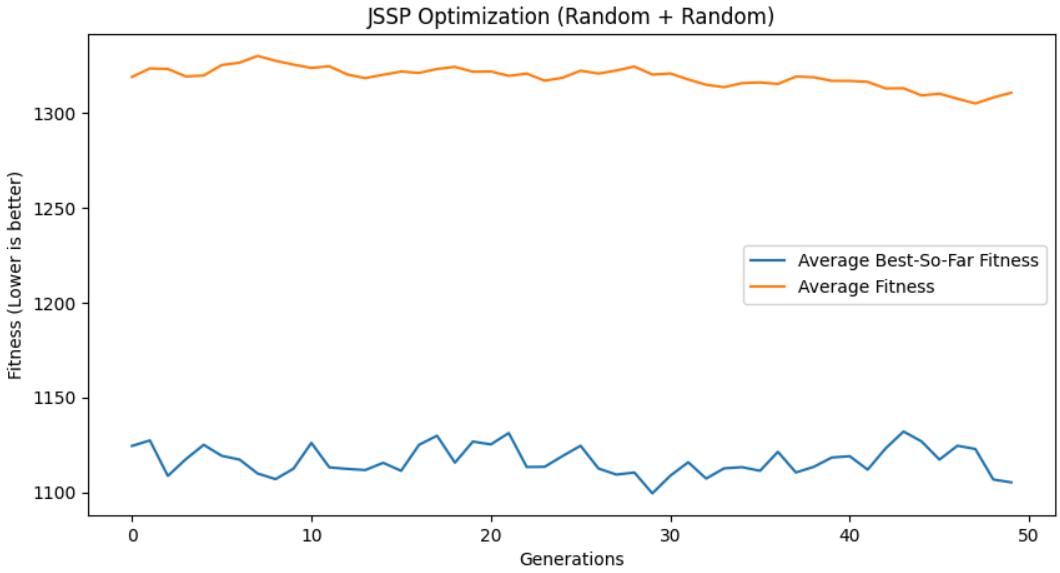


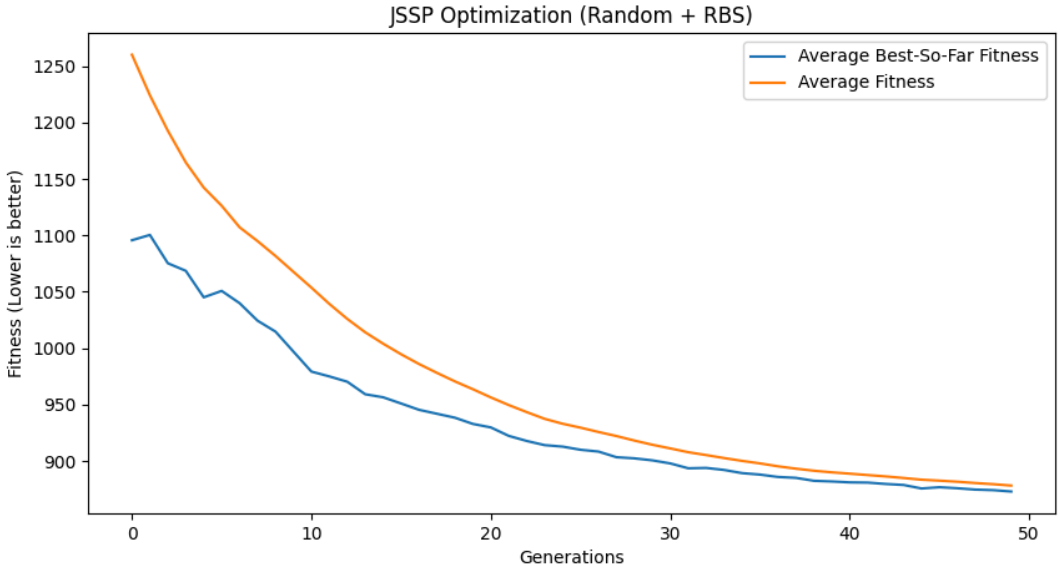


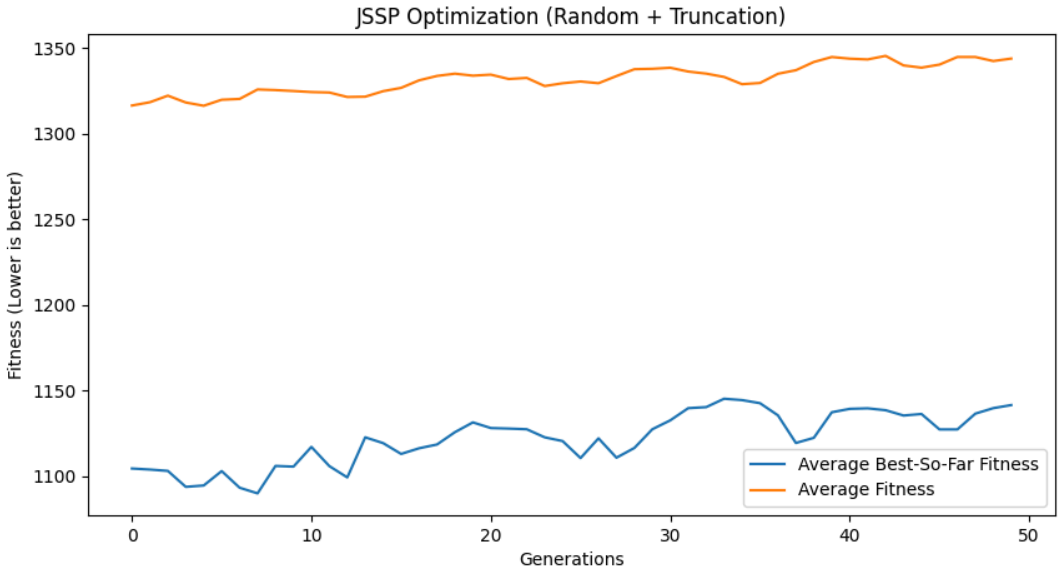

JSSP Optimization (BinaryTournament + Truncation)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
Generations
1300
1200
1100
1000
900
0
10
20
30
40
50

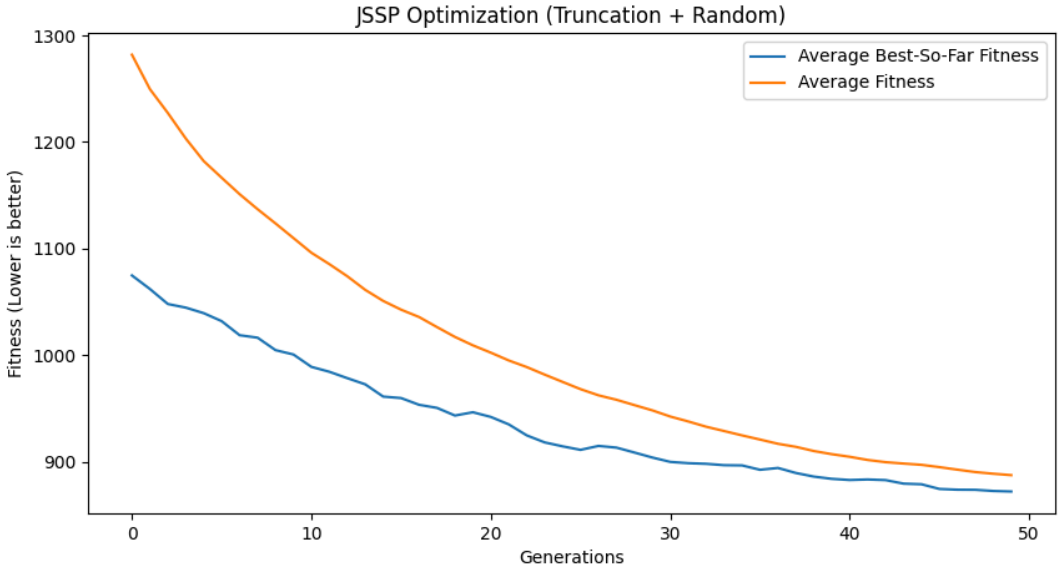

JSSP Optimization (FPS + RBS)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
Generations
1250
1200
1150
1100
1050
1000
950
900
0
10
20
30
40
50

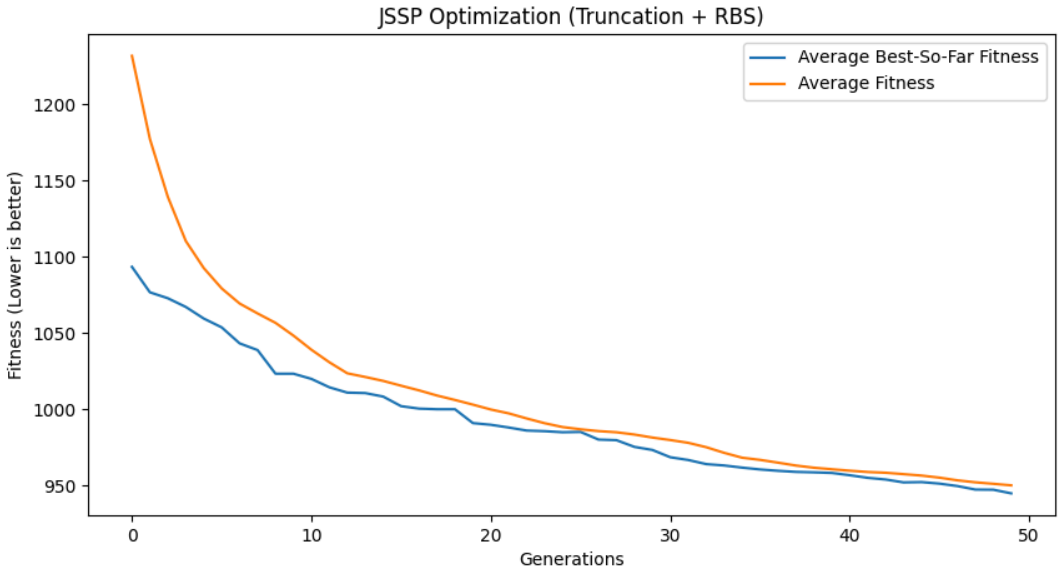

JSSP Optimization (FPS + Random)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
1300
1250
1200
1150
1100
0
10
20
30
40
50
Generations


JSSP Optimization (Random + Random)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
1300
1250
1200
1150
1100
0
10
20
30
40
50
Generations


JSSP Optimization (Random + RBS)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
Generations
1250
1200
1150
1100
1050
1000
950
900
0
10
20
30
40
50


JSSP Optimization (Random + Truncation)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
Generations
1350
1300
1250
1200
1150
1100
0
10
20
30
40
50

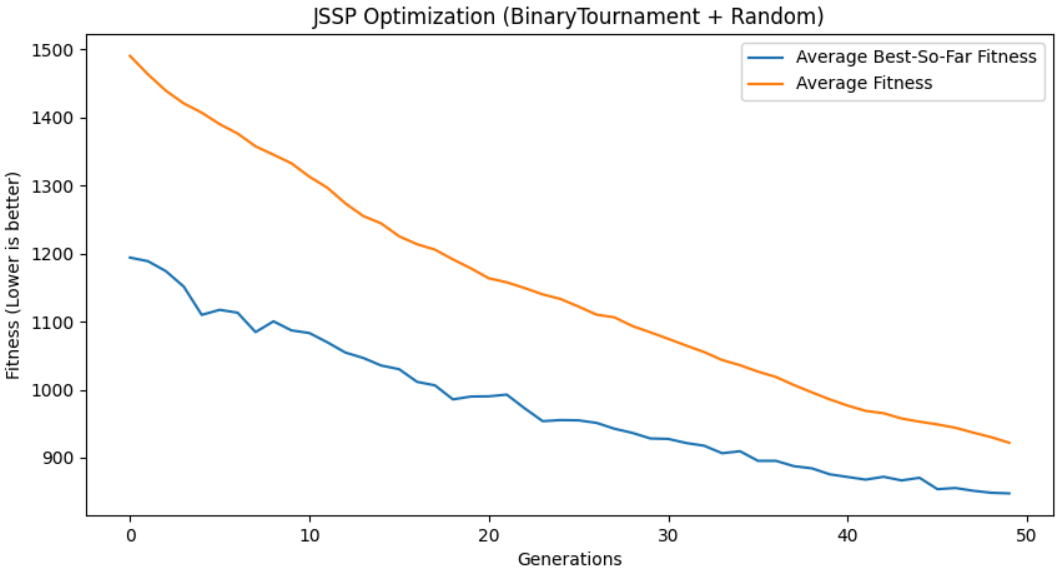


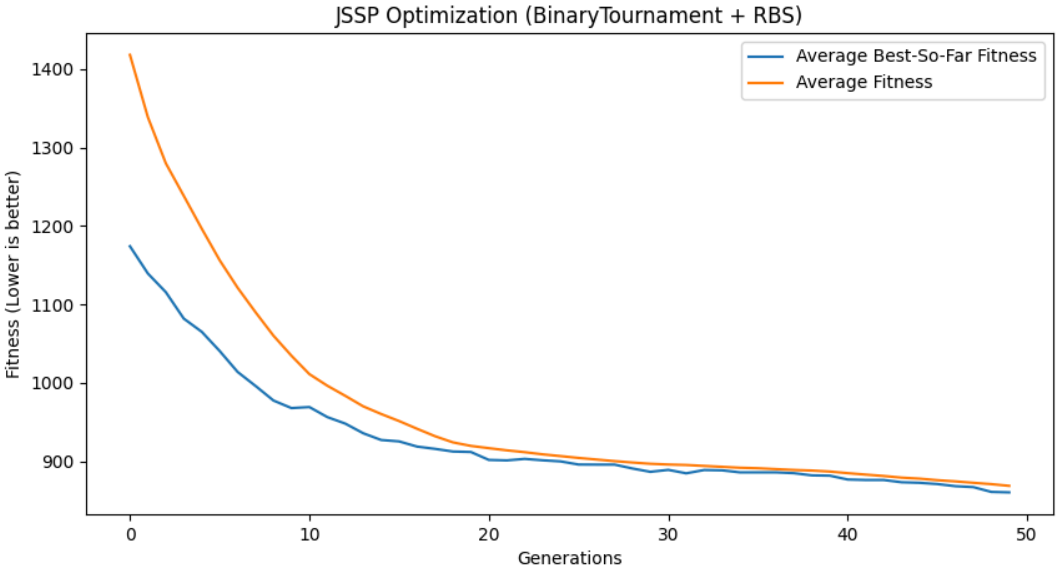


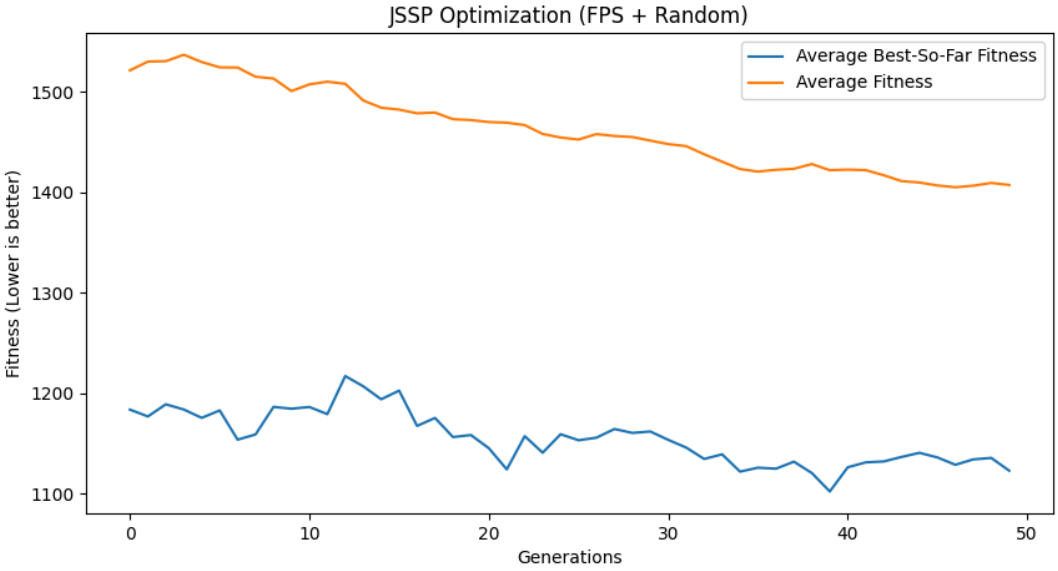
Analysis:

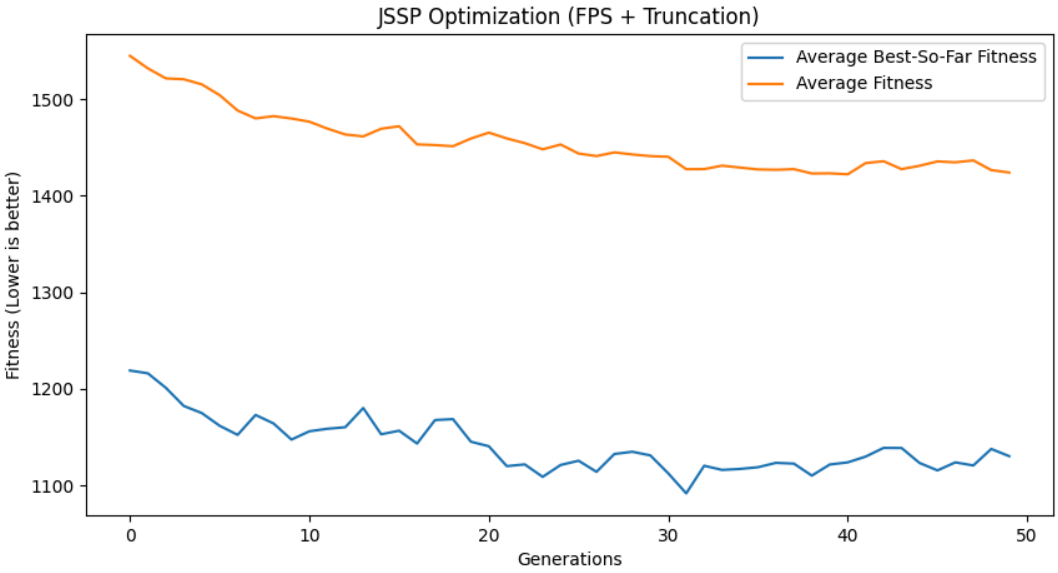
Optimal Value achieved was 865. The Binary Tournament + RBS scheme performed best because binary tournament selection ensures competitive parent selection, and rank-based selection maintains diversity while favoring better solutions. In contrast, the Random + Truncation scheme performed the worst because random parent selection introduced weak candidates, and truncation aggressively removed diversity, leading to premature convergence and suboptimal solutions.

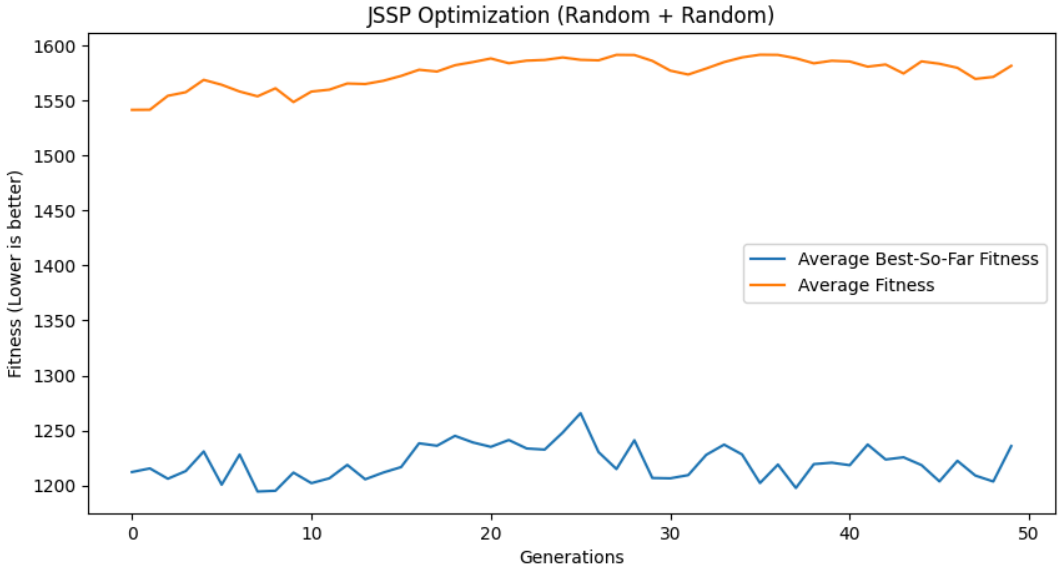
la19 Dataset

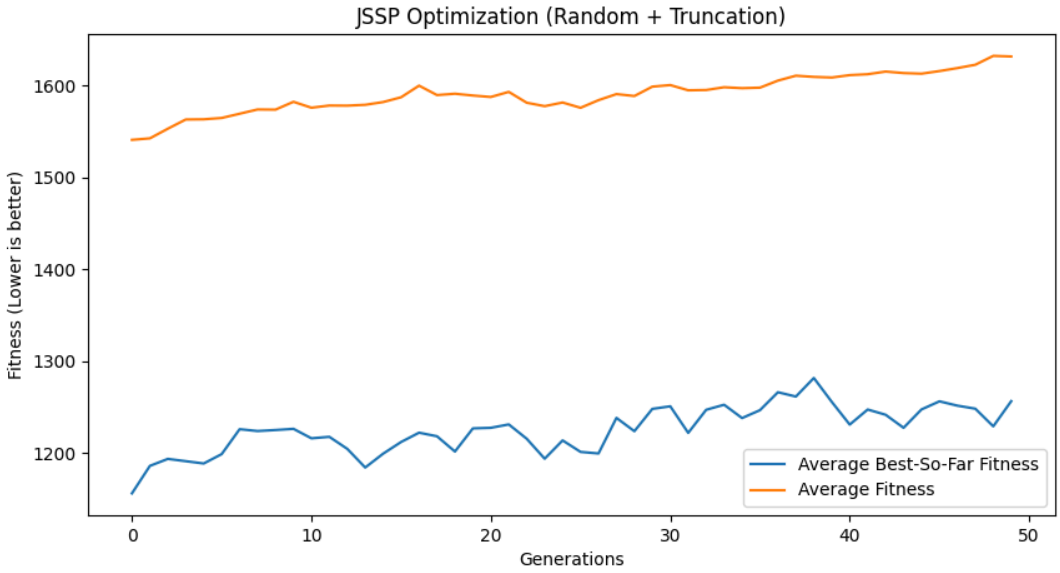


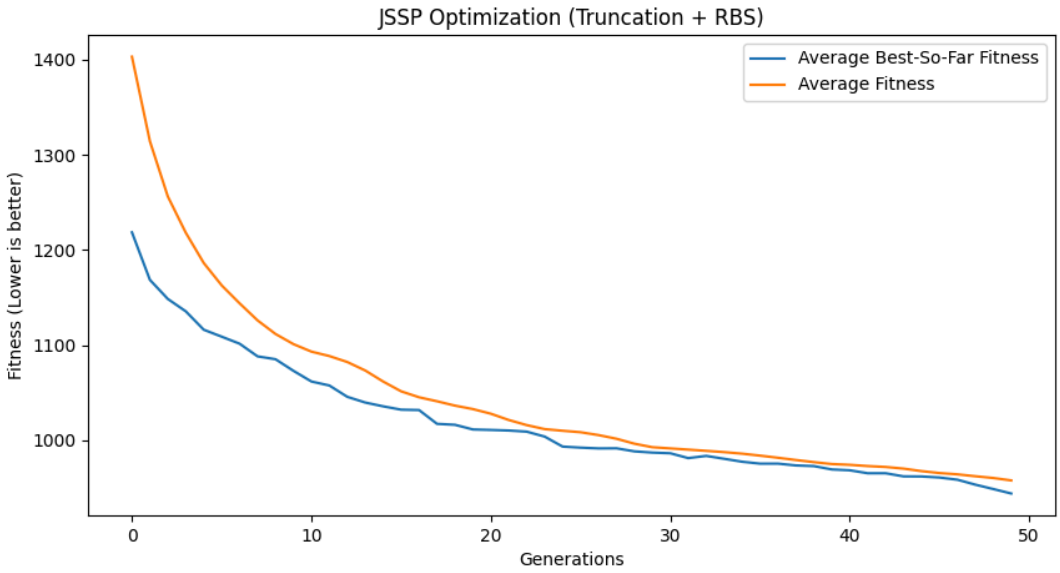


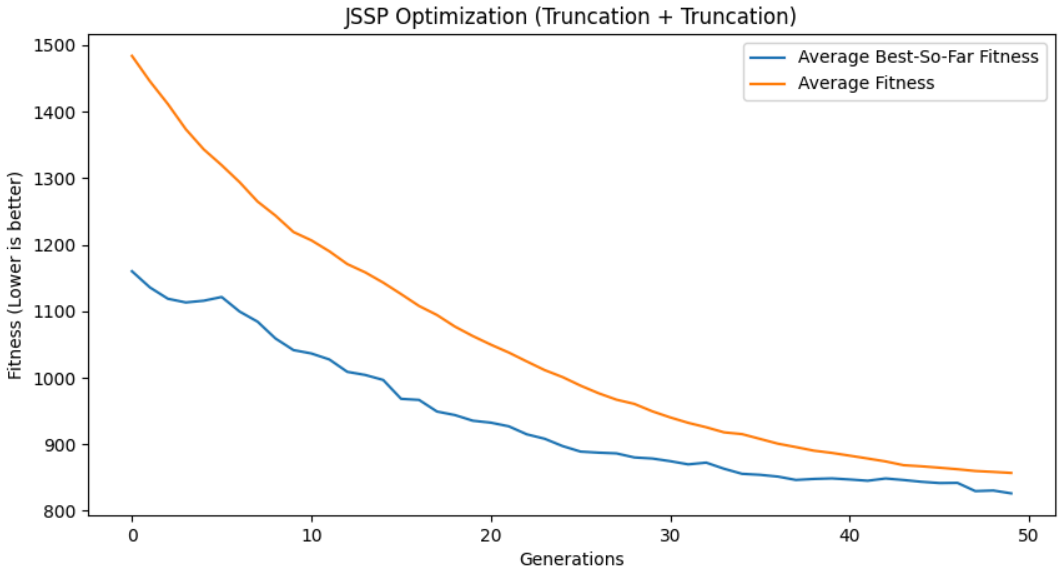

JSSP Optimization (FPS + Random)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
1500
1400
1300
1200
1100
0
10
20
30
40
50
Generations


JSSP Optimization (FPS + Truncation)
Average Best-So-Far Fitness
Average Fitness
Fitness (Lower is better)
1500
1400
1300
1200
1100
0
10
20
30
40
50
Generations


JSSP Optimization (Random + Random)
Fitness (Lower is better)
Generations
Average Best-So-Far Fitness
Average Fitness
1600
1550
1500
1450
1400
1350
1300
1250
1200
0
10
20
30
40
50


JSSP Optimization (Random + Truncation)
Fitness (Lower is better)
Generations
Average Best-So-Far Fitness
Average Fitness
1600
1500
1400
1300
1200
0
10
20
30
40
50





Analysis:

Optimal Value achieved was 855. The Truncation + Truncation scheme performed best because selecting both parents and survivors based on top fitness values ensures strong genetic material is consistently passed down, leading to faster convergence. On the other hand, the Random + Random scheme performed the worst since both selection processes were entirely random, preventing meaningful improvements and leading to poor overall optimization.

Q2 - Evolutionary Art (Mona Lisa)

Approach:

Initialization - The Mona Lisa Evolutionary Algorithm begins by creating an initial population of **population_size** individuals. Each individual represents a potential solution, which in this case is a set of polygons designed to resemble the Mona Lisa. These initial sets of polygons are generated randomly, with each polygon having a random number of vertices, random positions, and random colors (including alpha).

Fitness Function - The fitness function evaluates how well each individual's set of polygons matches the target Mona Lisa image. It calculates the pixel-wise difference between the image generated by the polygons and the reference Mona Lisa image. A lower difference indicates a better fit, so the fitness score is essentially the mean pixel difference. The goal is to minimize this difference.

Crossover - The algorithm uses a specialized crossover method, random_length_crossover, adapted for the polygon representation. This crossover combines the polygon sets of two parent individuals to create offspring. It selects a random segment from one parent's polygon list and inserts it into the offspring. Then, it fills the remaining polygon slots in the offspring with polygons from the second parent, ensuring that the total number of polygons doesn't exceed a maximum limit. This method allows for the mixing of "good" polygon combinations from different parents.

Mutation - The mutation process introduces variations into the offspring's polygon sets. Several types of mutations are applied, including: adding a new polygon, removing a polygon, moving (swapping the positions of) polygons, adding a vertex to a polygon, removing a vertex from a polygon, and slightly altering the color (red, green, blue, and alpha channels) and vertex positions of existing polygons.

Parents and Survivor Selection Scheme - The algorithm employs a combination of parent and survivor selection methods. For both, several options are available, like truncation, random, binary tournament, rank-based, and fitness-proportional selection. The choice of these methods significantly influences the algorithm's performance.

Parameters:

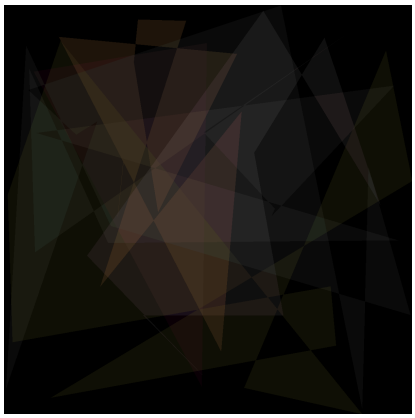
Population size = 100
No of offsprings = 50
No of generations = 10000
Mutation Rate = 0.5

Iterations = 10

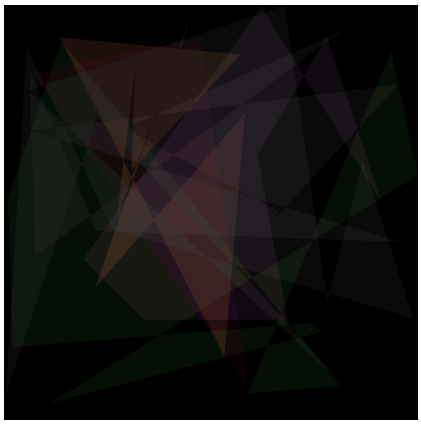
Modifications: In practice, the number of generations might need to be adjusted based on the complexity of the target image and the desired level of resemblance. For very complex images, or to achieve near-perfect results, many more generations might be necessary.

Results:

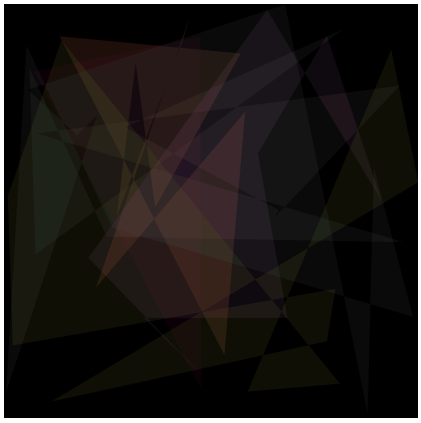
Generation 1



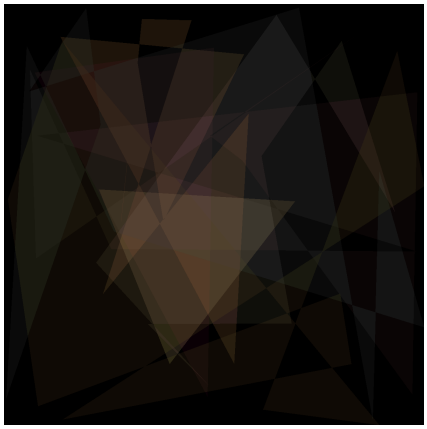
Generation 2500



Generation 5000



Generation 10000 (Face cut is appearing)



Analysis:

When evolving images to resemble the Mona Lisa, the choice of parent and survivor selection methods plays a crucial role. For example, truncation selection for both parents and survivors often performs well. Truncation selection ensures that the best individuals (those with the lowest fitness scores, representing the smallest image difference) are more likely to reproduce and survive, driving the population towards better solutions.