Computational Intelligence

Assignment 1: Evolutionary Algorithms

By

Nabila Zahra & Muhammad Youshay

Date: February 11, 2024

Contents

1 Problem 1: TSP

1.1 Problem

The Traveling Salesman Problem (TSP) is an optimization problem where:

- A salesperson is tasked with visiting a defined set of cities, each connected by specific distances or travel costs.
- The objective is to find the most efficient path that allows the salesperson to visit each city exactly once and return to the starting point.

Objective: Determine the shortest possible route that visits a given set of cities and returns to the starting city, minimizing the total distance traveled.

1.2 Chromosome Representation

A chromosome in this code is a list of numbers where each number represents a city, and the order of the numbers represents a specific path through all the cities. The goal of the genetic algorithm is to find the chromosome that represents the shortest possible path that visits every city once. So basically, a chromosome represents a solution to the Traveling Salesman Problem (TSP).

1.3 Fitness Function

The fitness function evaluates the total distance traveled for the path represented by a chromosome. It first initializes the total distance to 0. Then, it iterates through the chromosome from the 1st to the 2nd last city, using the TSP instance's distance matrix to look up the edge weight between each pair of consecutive cities. It adds this weight to the total distance. Once the entire path has been evaluated, the total distance is returned as the fitness score. Lower scores are fitter since they represent shorter total TSP tours. This fitness function directly captures the optimization objective of the TSP problem - minimizing the total tour distance. By minimizing the value returned by the fitness function through the genetic algorithm, optimal or near-optimal solutions to the TSP can be evolved.

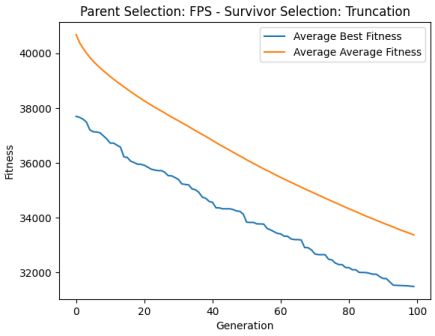
1.4 Results

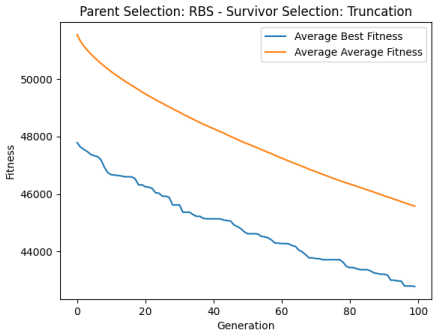
1.4.1 Parameters

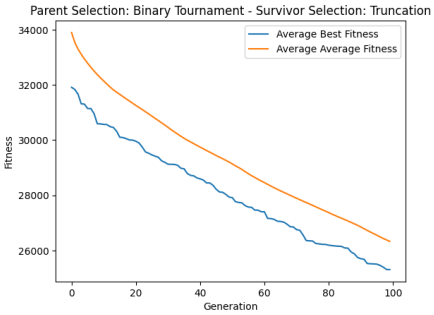
- Offspring Count = 1000
- No Of Generations = 100
- Mutation Rate = 0.5
- No Of Iterations = 10
- Population Size = 1000

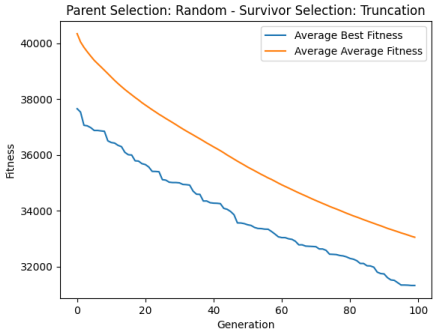
1.4.2 Plots

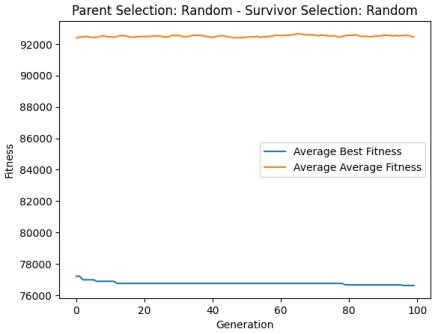
Plots of Avg. BSF and Avg. ASF for various combinations of schemes

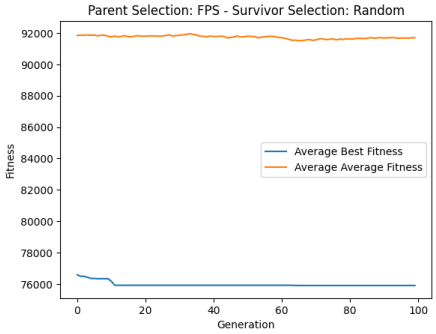


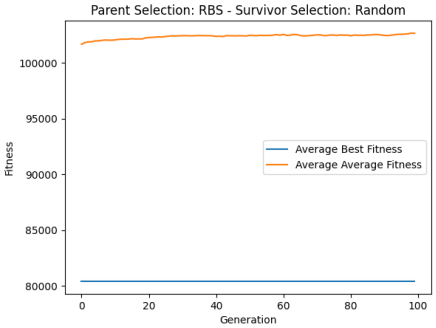


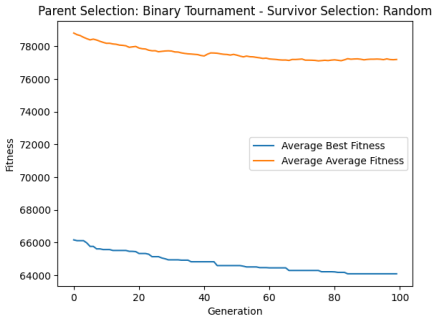


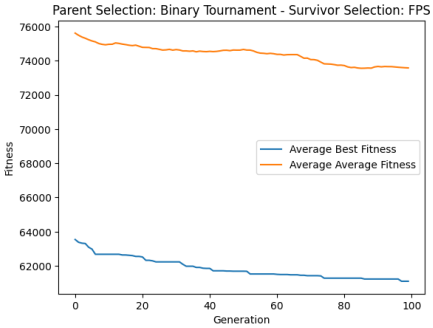


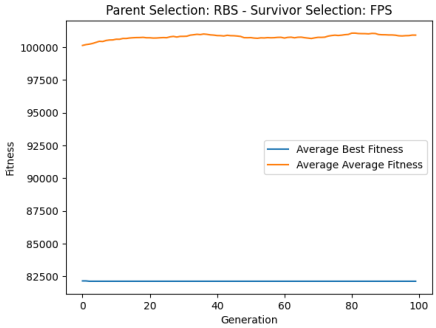


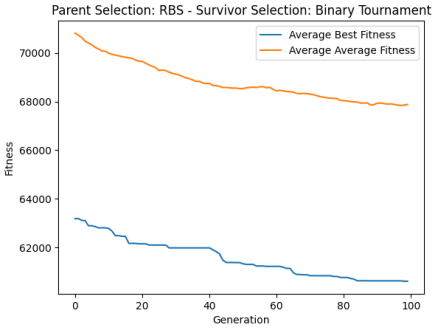


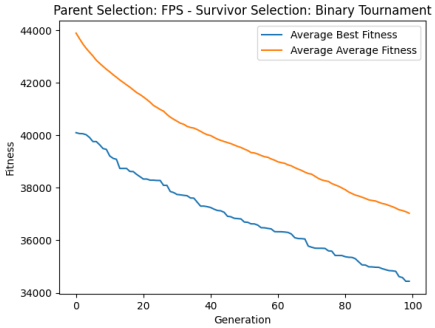

Parent Selection: RBS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

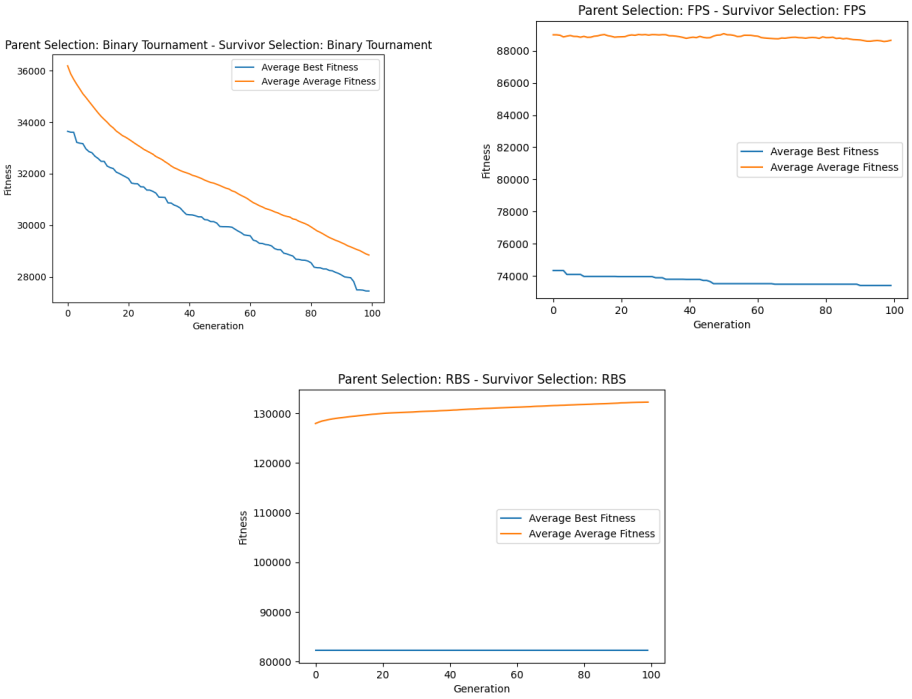

Parent Selection: Binary Tournament - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: Binary Tournament - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: FPS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation



Parent	Survivor	Best Fitness	Avg Fitness
FPS	Truncation	18825	19579
RBS	Truncation	30923	33951
Binary	Truncation	16402	17012
Random	Truncation	18982	19785
Random	Random	76273	93141
FPS	Random	75818	91473
RBS	Random	80408	102385
Binary	Random	61136	75898
Binary	FPS	58780	73307
RBS	FPS	82132	101072
RBS	Binary	56811	65055
FPS	Binary	22914	23800
Binary	Binary	18289	18983
FPS	FPS	73026	88930
RBS	RBS	82251	135452

Table 1: Avg ASF and Avg BSF values for various selection combinations

1.5 Analysis and Optimal Solution

1.5.1 Analysis

- Binary - Truncation: This combination resulted in the lowest best fitness (16402) and average fitness (17012). Binary tournament selection tends to maintain a good balance between exploration (searching new areas in the solution space) and exploitation (refining the current best solutions), which can lead to effective search. Truncation selection for survivors aggressively focuses on the best individuals, which might have helped push the performance.

- RBS - RBS: This combination resulted in the highest best fitness (82251) and average fitness (135452). Rank-Based Selection (RBS) gives every individual a chance to be selected, which can maintain diversity but might also maintain lower quality solutions. Using RBS for both parent and survivor selection might have resulted in too much emphasis on maintaining diversity and not enough on exploiting the best solutions.

- FPS -Truncation and Random -Truncation: These combinations have similar best and average fitness values (18825 and 18982 for best fitness, 19579 and 19785 for average fitness, respectively). This suggests that the parent selection scheme (FPS or Random) might not have had a significant impact on the performance when used with truncation survivor selection.

- Random- Random, FPS- Random, and RBS- Random: These combinations resulted in relatively high best and average fitness values. This suggests that using random selection for survivors, regardless of the parent selection scheme, might not be effective for this problem. Random survivor selection can introduce randomness into the survivor pool, which might hinder the algorithm's ability to converge to good solutions.

- Binary- FPS, RBS -FPS, and FPS- FPS: These combinations show that using FPS for survivor selection can lead to varied results depending on the parent selection scheme. Binary FPS resulted in relatively low best and average fitness values, suggesting that it might be a good combination. However, RBS FPS and FPS FPS resulted in high fitness values, suggesting that they might not be effective.

- RBS- Binary, FPS- Binary, and Binary- Binary: These combinations show that using binary selection for survivors can also lead to varied results. RBS Binary and Binary Binary resulted in relatively low fitness values, while FPS Binary resulted in higher fitness values. This suggests that the effectiveness of binary survivor selection might depend on the parent selection scheme.

1.5.2 Optimal Solution

Analysing the data in the table above tells us that we are getting the best result when we choose Binary Tournament and Truncation selection schemes. To further check how much can we optimize our solution, we further modified the parameters to get an optimal solution using these selection schemes. The new parameters that we chose were -

- Offspring Count = 1000
- No Of Generations = 1000
- Mutation Rate = 0.5
- No Of Iterations = 3
- Population Size = 1000
- Parent Scheme: Binary Tournament
- Survivor Scheme: Truncation

Through this we were able to get a best fitness of **13979** and our best chromosome was [65, 20, 63, 62, 59, 36, 8, 6, 4, 7, 45, 57, 60, 33, 24, 17, 14, 11, 9, 10, 12, 31, 38, 41, 46, 44, 35, 30, 32, 43, 40, 34, 27, 22, 18, 21, 28, 29, 37, 39, 51, 47, 56, 53, 52, 54, 55, 50, 42, 49, 67, 73, 70, 74, 75, 78, 91, 102, 109, 113, 122, 147, 151, 158, 159, 152, 153, 150, 154, 157, 165, 168, 167, 166, 160, 148, 143, 133, 129, 135, 136, 131, 121, 117, 116, 108, 107, 105, 96, 95, 88, 83, 81, 84, 100, 97, 92, 79, 77, 26, 25, 13, 16, 23, 71, 93, 110, 112, 115, 120, 123, 124, 128, 164, 163, 161, 149, 146, 139, 138, 142, 137, 134, 132, 144, 141, 119, 126, 125, 140, 145, 130, 111, 94, 99, 101, 104, 114, 127, 156, 169, 172, 179, 174, 173, 175, 187, 190, 192, 191, 193, 185, 171, 170, 180, 178, 184, 183, 186, 176, 182, 194, 189, 188, 181, 177, 162, 155, 118, 106, 103, 87, 80, 76, 72, 69, 64, 68, 66, 61, 58, 48, 19, 15, 5, 3, 2, 1, 82, 89, 90, 98, 85, 86].

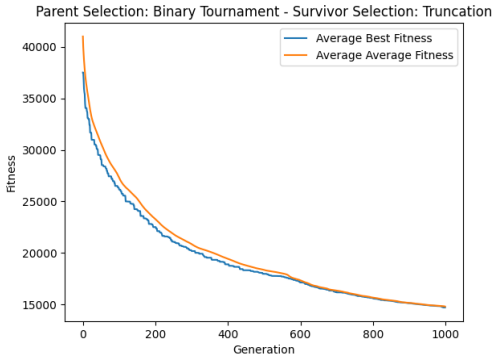


Figure 1: Avg BSF and Avg ASF graph for optimal solution

2 Problem 2: Job-Shop Scheduling

2.1 Problem

The job shop scheduling problem is an optimization problem where:

- A set of n jobs $\{J_1, J_2, \ldots, J_n\}$ to be scheduled.
- Each job J_i consists of an ordered sequence of operations $\{O_{i1}, O_{i2}, \ldots, O_{im}\}$, where m may vary for each job.
- A set of k machines $\{M_1, M_2, \ldots, M_k\}$, where each operation O_{ij} is to be processed on a specific machine M_l for a specified duration d_{ijl}.
- Precedence constraints indicating that each task within a job must be completed before the next task in the job can begin.
- Machine constraints indicating that each machine can only process one task at a time.

Objective: Minimize the makespan, which is the total time required to complete all jobs.

2.2 Chromosome Representation

The chromosome is represented as a permutation of job IDs. Each gene in the chromosome corresponds to a job ID, indicating that an operation from the specified job should be executed next.

- The length of a chromosome is equal to the number of machines multiplied by the number of jobs, ensuring that each operation has a designated position in the schedule.
- The placement of job IDs reflects the order in which operations from the various jobs will be scheduled.
- The job IDs in a chromosome are permuted and can repeat up to the number of operations required for that job, adhering to the constraint that each job's operations must be completed in a specific sequence but can be interleaved with operations from other jobs.
- This permutation-based representation is suitable for genetic algorithms, as it allows for the straightforward application of crossover and mutation operators to explore the search space of possible schedules.

2.3 Fitness Function

The primary goal of this fitness function is to minimize the makespan, which is the total time required to complete all jobs. The makespan is a critical measure

of schedule efficiency in job shop scheduling problems.

Makespan Calculation: For each chromosome (schedule) in the population, the fitness function simulates the execution of jobs according to the order specified by the chromosome. It keeps track of the completion times for each job and the availability times of each machine. Operations are scheduled as soon as both the job and machine are available, adhering to the job's operation sequence. The makespan for a chromosome is the maximum completion time across all jobs.

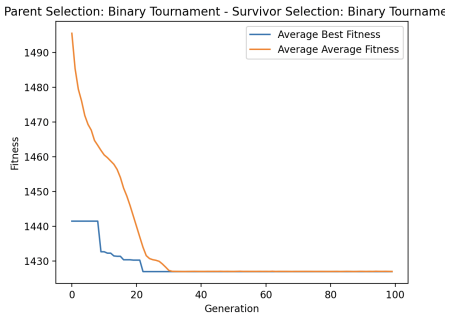
The fitness of a chromosome is inversely related to its makespan. A lower makespan indicates a more efficient schedule and thus a fitter chromosome. The fitness function returns a list of makespan values for the entire population, which can be used to select and breed the fittest individuals in subsequent generations of the genetic algorithm.

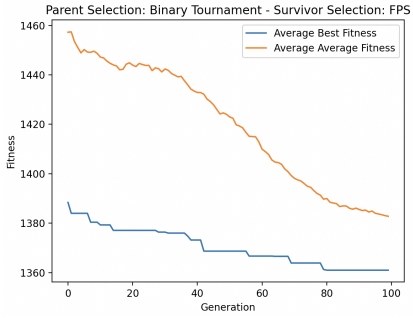
2.4 Plots

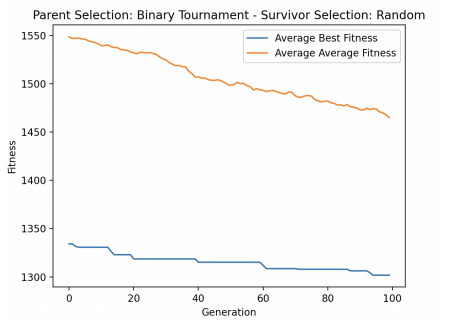
2.5 Instance 1

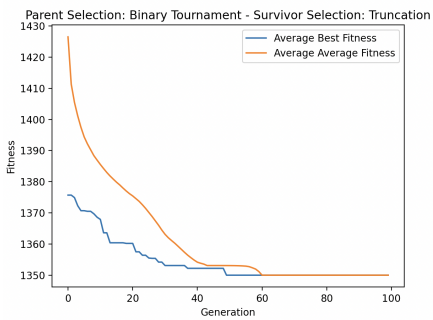
Parent	Survivor	Best Fitness	Avg Fitness
FPS	Truncation	1327	1329.288
RBS	Truncation	1321	1321.57
Binary	Truncation	1350	1352.433
Random	Truncation	1351	1352.693
Random	Random	1447	2510.985
FPS	Random	1499	2327.063
RBS	Random	1599	6865.3655
Binary	Random	1275	1275
Binary	FPS	1361	1363.6035
RBS	FPS	1455	3076.877
RBS	Binary	1332	1356.7895
FPS	Binary	1372	1373.3415
Binary	Binary	1427	1428.4175
FPS	FPS	1294	1433.88
RBS	RBS	1616	5021.7845

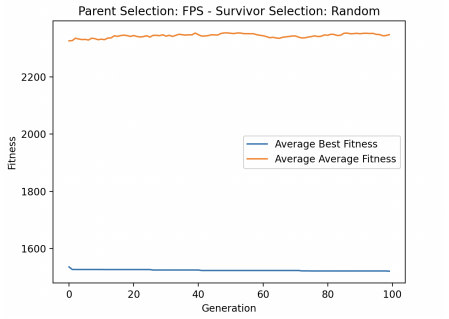
Table 2: Instance 2 - comparision of different combinations of selection schemes

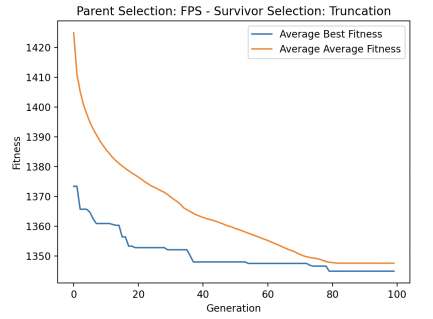

Parent Selection: Binary Tournament - Survivor Selection: Binary Tourname
Average Best Fitness
Average Average Fitness
Fitness
Generation

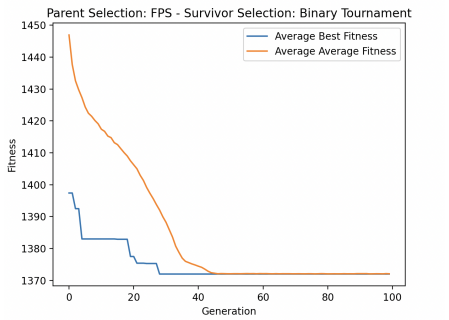

Parent Selection: Binary Tournament - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

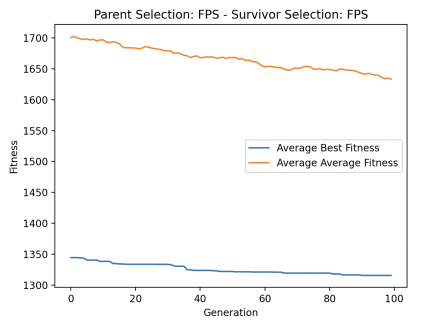

Parent Selection: Binary Tournament - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

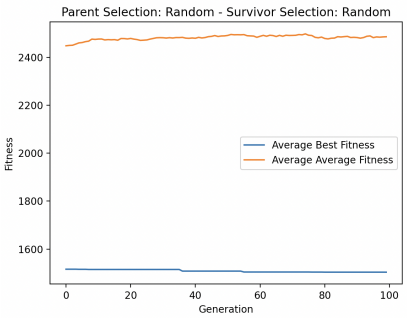

Parent Selection: Binary Tournament - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

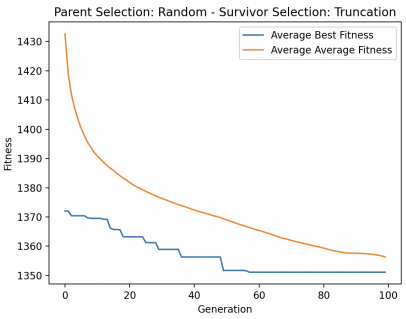

Parent Selection: FPS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

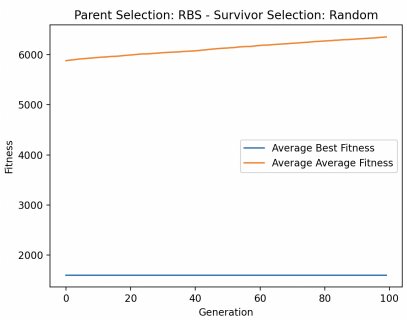

Parent Selection: FPS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

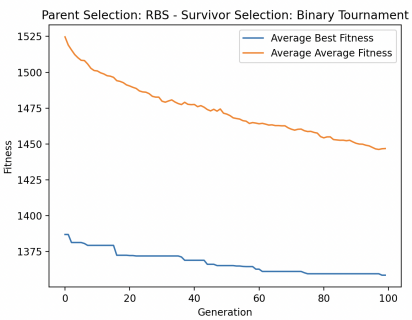

Parent Selection: FPS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation

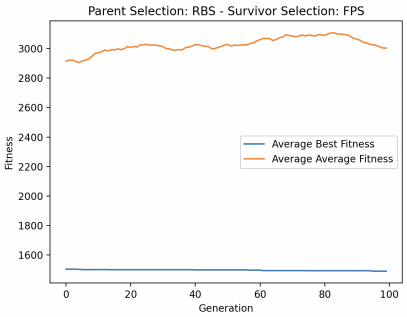

Parent Selection: FPS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

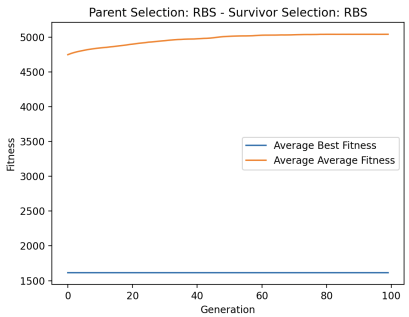

Parent Selection: Random - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

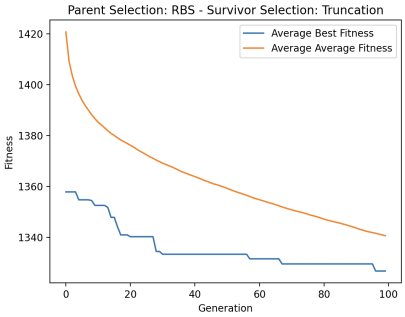

Parent Selection: Random - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

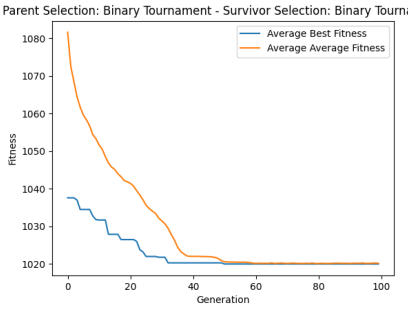

Parent Selection: RBS - Survivor Selection: RBS
Average Best Fitness
Average Average Fitness
Fitness
Generation

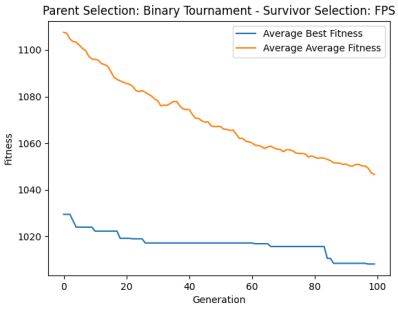

Parent Selection: RBS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

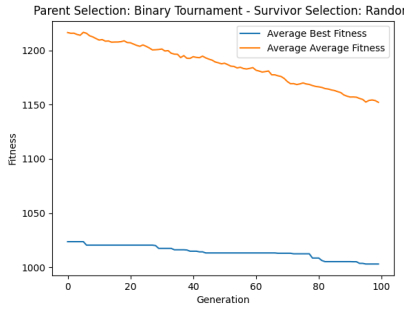
2.6 Instance 2

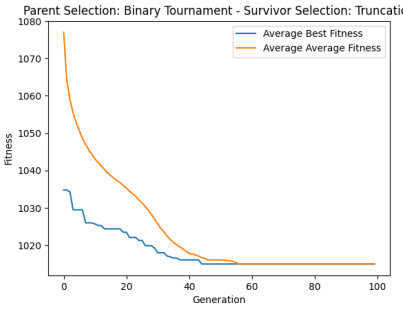
Parent	Survivor	Best Fitness	Avg Fitness
Binary	Random	976	976.73
FPS	FPS	997	1047.107
FPS	Truncation	1012	1012.0125
RBS	Truncation	1006	1006.028
Binary	Truncation	1015	1015.58
Random	Truncation	1008	1010
Random	Random	1141	1948.177
FPS	Random	1096	1797.11
RBS	Random	1204	5162
Binary	FPS	1008	1009.711
RBS	FPS	1047	2312.75
RBS	Binary	1001	1001.99
FPS	Binary	1030	1031.855
Binary	Binary	1026	1026
RBS	RBS	1197	4053.75

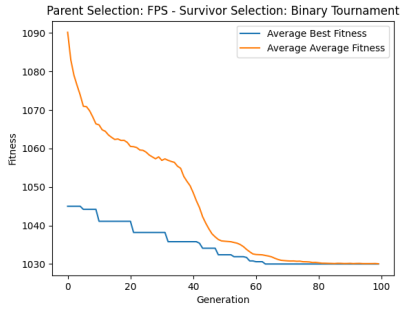
Table 3: Instance 2 - comparision of different combinations of selection schemes

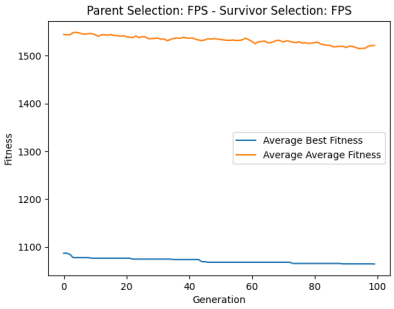

Parent Selection: Binary Tournament - Survivor Selection: Binary Tournamen
Average Best Fitness
Average Average Fitness
Fitness
Generation

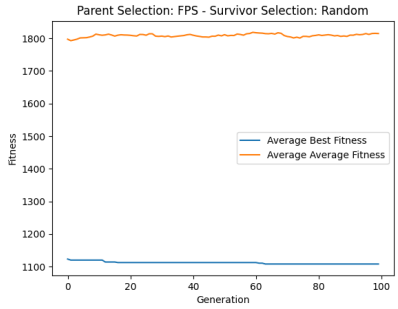

Parent Selection: Binary Tournament - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

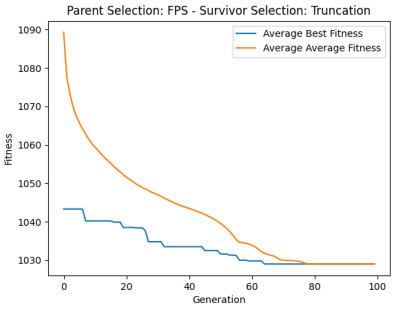

Parent Selection: Binary Tournament - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

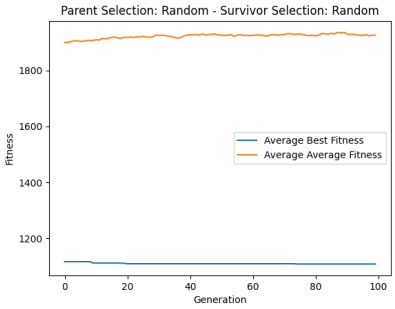

Parent Selection: Binary Tournament - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

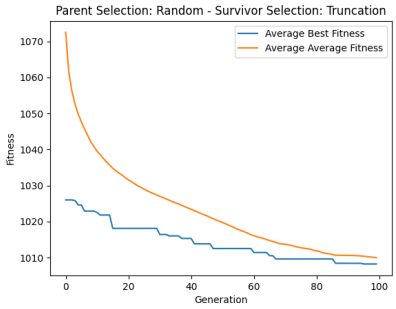

Parent Selection: FPS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation

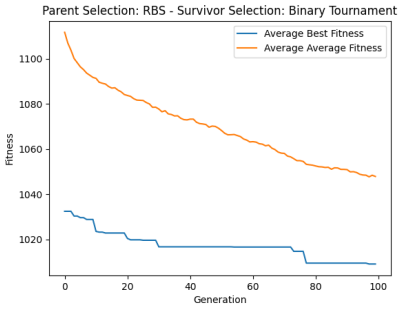

Parent Selection: FPS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

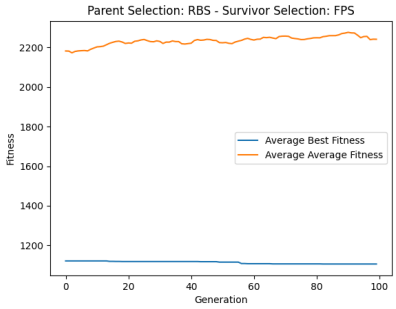

Parent Selection: FPS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

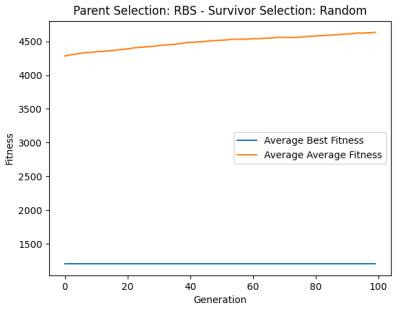

Parent Selection: FPS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

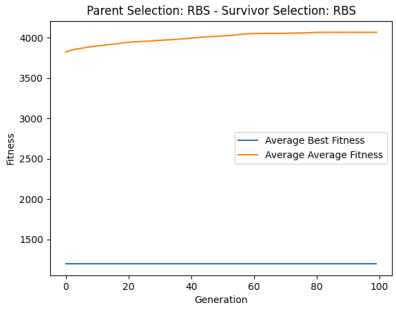

Parent Selection: Random - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

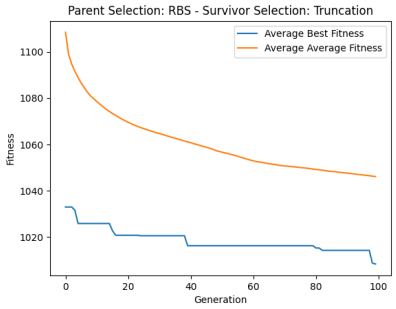

Parent Selection: Random - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

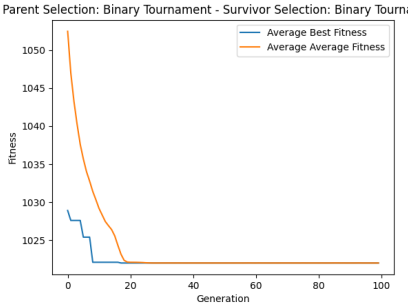

Parent Selection: RBS - Survivor Selection: RBS
Average Best Fitness
Average Average Fitness
Fitness
Generation

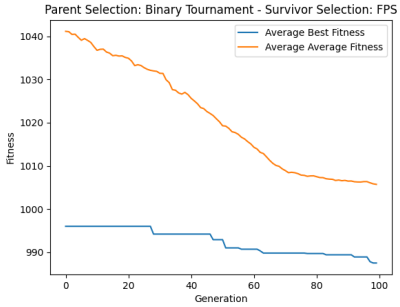

Parent Selection: RBS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

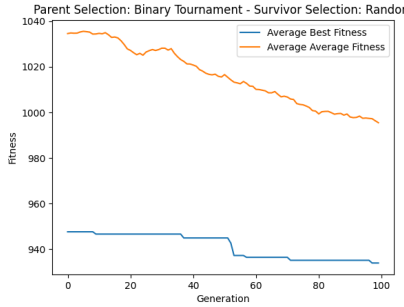
2.7 Instance 3

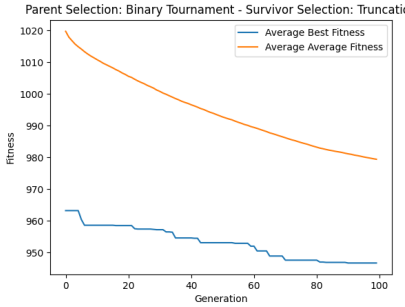
Parent	Survivor	Best Fitness	Avg Fitness
Binary	Random	927	1091.129
FPS	FPS	972	999.96
FPS	Truncation	929	1014.33
RBS	Truncation	945	1087
Binary	Truncation	930	979
Random	Truncation	877	982
Random	Random	1048	1752
FPS	Random	1004	1660
RBS	Random	1092	2536
Binary	FPS	975	1005
RBS	FPS	956	1096
RBS	Binary	1057	1057
FPS	Binary	1041	1041
Binary	Binary	1022	1022
RBS	RBS	1082	2572

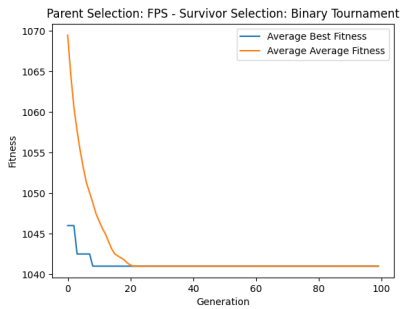
Table 4: Instance 3 - comparision of different combinations of selection schemes

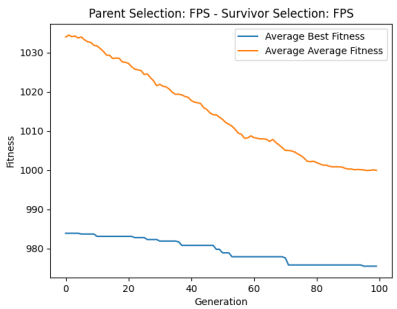

Parent Selection: Binary Tournament - Survivor Selection: Binary Tournamen
Average Best Fitness
Average Average Fitness
Fitness
Generation

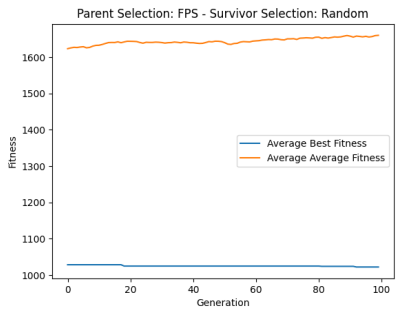

Parent Selection: Binary Tournament - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

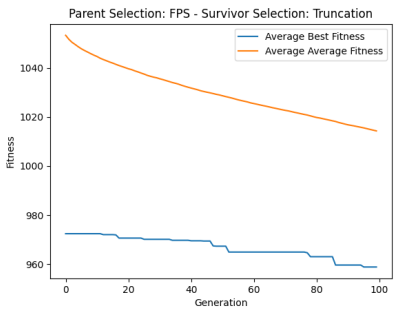

Parent Selection: Binary Tournament - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

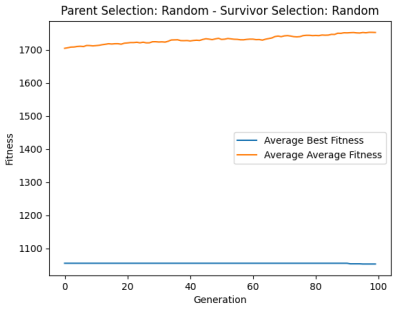

Parent Selection: Binary Tournament - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

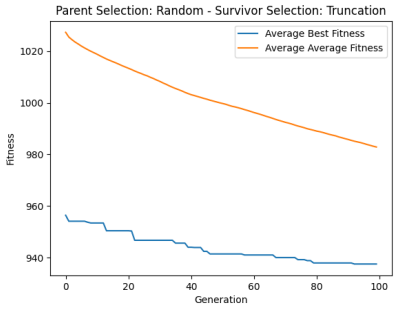

Parent Selection: FPS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation

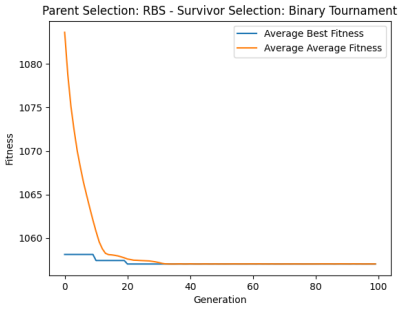

Parent Selection: FPS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation

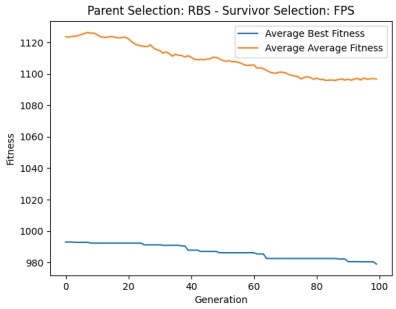

Parent Selection: FPS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

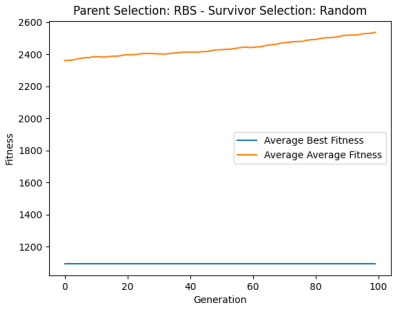

Parent Selection: FPS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

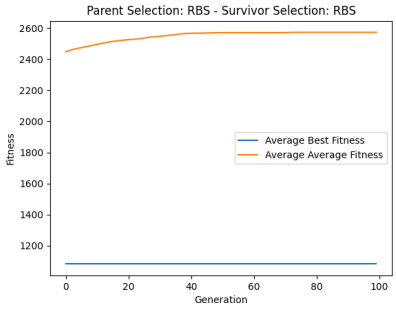

Parent Selection: Random - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation

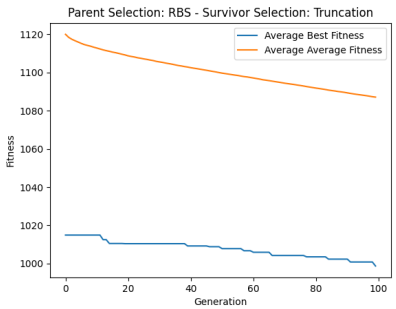

Parent Selection: Random - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Binary Tournament
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: FPS
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: Random
Average Best Fitness
Average Average Fitness
Fitness
Generation


Parent Selection: RBS - Survivor Selection: RBS
Average Best Fitness
Average Average Fitness
Fitness
Generation

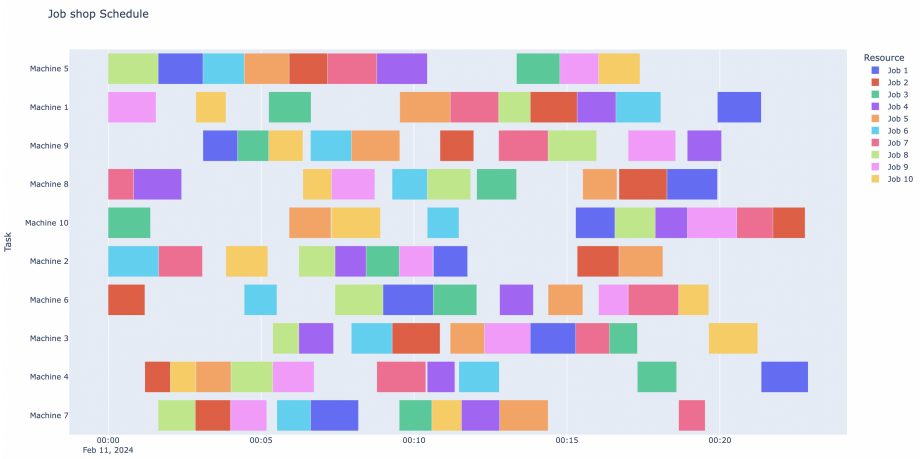

Parent Selection: RBS - Survivor Selection: Truncation
Average Best Fitness
Average Average Fitness
Fitness
Generation

2.8 Optimal Value and optimal Solution

Configuration:
offspringCount = 1000
numGenerations = 100
mutationRate = 0.5
numIterations = 10
populationSize = 1000

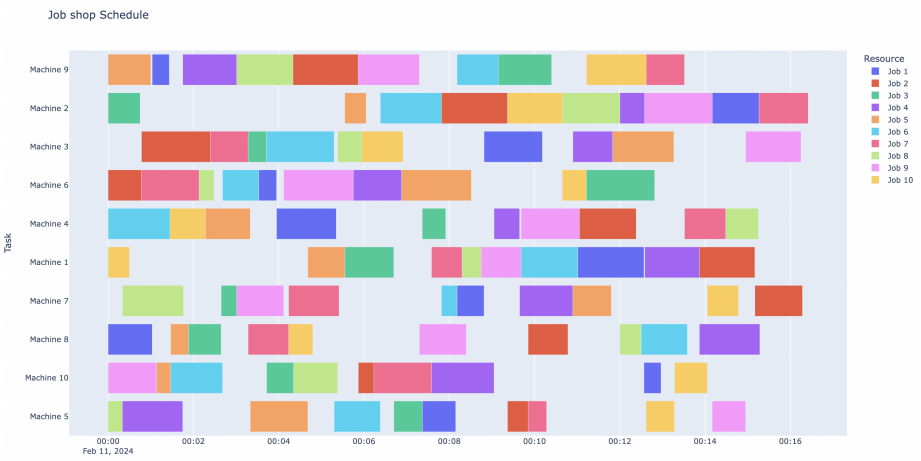
2.8.1 Instance 1

Optimal Fitness value: 1275
Parent Scheme: Binary Tournament
Survivor Scheme: Random
Best Chromosome: [5, 2, 3, 8, 7, 8, 2, 7, 6, 9, 4, 1, 4, 9, 9, 5, 0, 1, 3, 1, 2, 9, 7, 4, 5, 3, 8, 7, 9, 5, 6, 7, 4, 1, 0, 5, 1, 1, 0, 8, 9, 3, 7, 0, 6, 3, 2, 5, 8, 3, 0, 6, 4, 7, 5, 8, 7, 2, 2, 3, 4, 2, 6, 0, 8, 1, 4, 7, 8, 6, 5, 3, 7, 1, 0, 9, 6, 4, 6, 5, 2, 9, 8, 0, 3, 1, 2, 8, 1, 9, 0, 4, 4, 6, 2, 5, 9, 6, 3, 0]



2.8.2 Instance 2

Optimal Fitness value: 976
Parent Scheme: Binary Tournament
Survivor Scheme: Random
Best Chromosome: [7, 3, 1, 9, 0, 1, 4, 9, 2, 0, 3, 6, 8, 2, 0, 9, 8, 4, 2, 9, 5, 4, 6, 1, 8, 9, 7, 4, 4, 8, 5, 0, 2, 0, 5, 1, 3, 7, 9, 2, 4, 6, 0, 2, 8, 1, 8, 5, 7, 1, 3, 4, 0, 1, 5, 7, 0, 6, 2, 7, 3, 7, 4, 5, 5, 5, 4, 6, 8, 3, 9, 6, 8, 6, 3, 4, 7, 9, 5, 1, 0, 6, 8, 9, 7, 5, 1, 3, 2, 8, 3, 6, 2, 1, 0, 6, 3, 2, 7, 9]



2.8.3 Instance 3

Optimal Fitness value: 877
Parent Scheme: Random
Survivor Scheme: Truncation
Best Chromosome: [0, 1, 4, 15, 1, 0, 18, 2, 5, 6, 4, 6, 10, 11, 12, 7, 11, 9, 0, 7, 9, 7, 6, 5, 17, 12, 17, 8, 13, 15, 1, 15, 8, 5, 9, 17, 9, 4, 17, 12, 10, 0, 17, 2, 11, 18, 15, 14, 9, 2, 15, 18, 3, 14, 8, 8, 13, 13, 5, 6, 6, 19, 16, 10, 14, 16, 3, 19, 8, 11, 16, 18, 1, 7, 18, 6, 7, 17, 19, 17, 19, 11, 4, 1, 17, 2, 4, 14, 18, 18, 5, 0, 16, 6, 19, 17, 6, 9, 11, 11, 13, 6, 14, 8, 11, 9, 8, 9, 4, 10, 8, 1, 2, 9, 3, 11, 14, 13, 12, 8, 19, 1, 5, 2, 1, 14, 16, 12, 3, 4, 7, 5, 10, 3, 12, 9, 7, 10, 2, 1, 3, 3, 18, 16, 18, 19, 19, 12, 17, 18, 14, 18, 13, 8, 3, 16, 0, 4, 2, 18, 16, 1, 0, 5, 18, 13, 12, 17, 11, 4, 8, 4, 10, 2, 10, 10, 14, 7, 14, 8, 10, 7, 1, 7, 15, 11, 11, 16, 13, 3, 7, 0, 15, 8, 16, 2, 7, 15, 5, 4, 16, 14, 12, 5, 0, 19, 19, 6, 3, 12, 10, 5, 3, 16, 2, 0, 14, 8, 4, 9, 13, 1, 6, 8, 18, 17, 15, 14, 13, 0, 0, 14, 17, 2, 14, 7, 2, 6, 3, 13, 11, 6, 0, 16, 16, 6, 19, 7, 9, 12, 15, 3, 10, 4, 2, 5, 17, 9, 5, 9, 2, 9, 13, 1, 15, 19, 19, 5, 12, 19, 10, 13, 1, 3, 17, 15, 13, 4, 15, 13, 7, 0, 6, 4, 19, 5, 11, 11, 0, 15, 12, 10, 12, 15, 18, 16, 3, 12, 1, 10]

2.9 Analysis

2.9.1 Performance under Different Problem Sizes:

For smaller problems (10 jobs and 10 machines) with equal number of jobs and machines, RBS combined with Truncation and Binary Tournament with Random provided some of the best fitness values. As the problem size increased (20 jobs and 15 machines) with unequal number of jobs and machines, Binary with Truncation emerged as more effective, suggesting that these methods are scalable and adapt well to increased complexity.

2.9.2 Influence of Random Selection:

The use of Random selection for both parents and survivors (Random - Random) tends to result in poorer performance in terms of best and average fitness. This is expected, as random selection does not effectively exploit and explore the solution space.

2.9.3 Parent and Survivor Selection Combination:

The combination of Binary selection for parents and Truncation for survivors consistently yielded good results across all datasets. This suggests that a more deterministic approach to survivor selection (i.e., choosing the best individuals) complements the stochastic nature of binary tournament selection well.
FPS and Truncation also performed well, particularly in the first and second datasets. This indicates that a balance between selection pressure (from FPS) and preserving elite individuals (through truncation) is effective for these problem sizes.

2.9.4 Impact of Selection Mechanisms:

The choice of parent and survivor selection strategies has a significant impact on both the best and average fitness across all datasets. Fitness-based selection mechanisms, like Fitness Proportional Selection (FPS) and Rank-Based Selection (RBS), generally perform variably, with their effectiveness heavily influenced by the survivor selection strategy.

2.9.5 Variability in Average Fitness:

High variability in average fitness, especially when Random is involved in the survivor selection, indicates a lack of convergence towards optimal solutions. This is evident in the significantly higher average fitness values in some configurations, suggesting a wide spread in the quality of solutions.

2.9.6 Different configurations of parameters:

Increasing the population size and offspring size yielded in better results as there was an increased diversity. With more off springs, the exploration increased. Increasing the number of generations also yielded in better results as it gives the population more time to evolve to optimal solutions.

3 Problem 3: Evolutionary Art

3.1 Problem

Objective: The core objective is to leverage our evolutionary algorithm to evolve an image, in our case the iconic "Mona Lisa." The process involves the initiation of a population of random 'chromosome' representations of images, which then undergo iterative selection, crossover, and mutation with the aim of minimizing the difference between the evolved image and the original image.

3.2 Chromosome Representation

In the context of this question, chromosome represent a potential solution. Specifically, each chromosome is an image, itself a collection of polygons with varying vertices, color, and position. The representation takes the form of a dictionary in Python, where each dictionary key corresponds to an attribute relevant to the image:

- 'height' and 'width' define the dimensions of the image.
- 'fitness' holds the fitness score, a measure of how close this image is to the target image
- 'image' is the actual PIL Image object consisting of polygons that form the chromosome's visual representation.

- 'array' consists of the pixel data converted to a NumPy array for fitness computation.

This abstract representation is essential for the genetic algorithm as it allows manipulation through crossover and mutation functions and facilitates the evaluation of likeness to the target image.

3.3 Fitness Function

The fitness function quantitatively evaluates how close a given solution (chromosome) is to the optimal solution. In this code, the fitness function computes the visual difference between the chromosome's image and the target image using the CIE76 delta E color difference function, provided by the 'colour' library – this is a scientifically grounded metric used to compare color differences between images.

The rationale for using the delta E CIE76 formula for the fitness function lies in its ability to mimic human perception of color differences, which is crucial when the goal is to approximate a human-made painting like the "Mona Lisa." (**reference:** https://medium.com/@sebastian.charmot/genetic-algorithm-for-image-recreation-4ca546454aaa) A lower fitness score indicates a higher resemblance to the target, guiding the evolutionary process towards more accurate representations as the generations progress. The use of this fitness function ensures the selection pressure during the genetic algorithm's run pushes the population to enhance the visual fidelity relative to the "Mona Lisa."

3.4 Results

3.4.1 Parameters:

- Population Size: 100
- Number of Polygons: 50
- Number of Vertices: 3
- Number of Generations: 10,000

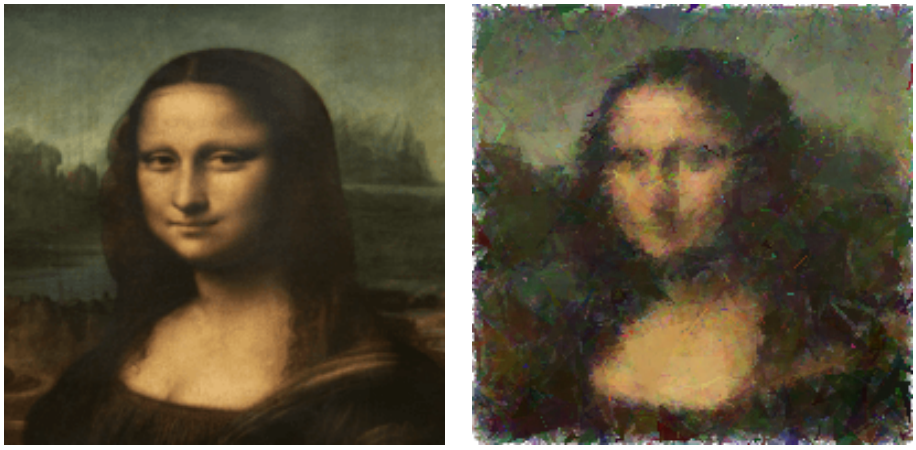


Figure 2: Target Image vs Evolved Image (according to the parameters mentioned above)

3.4.2 Analysis and Evolution Visualization

Implementing a genetic algorithm to approximate the intricate details of the "Mona Lisa" through image evolution presented a formidable challenge, due to the broad search space and nuanced color variations. We employed a larger population and increased generational iterations to refine the produced images; these modifications enhanced the outcomes but demanded higher computational time. Crucially, the adoption of elitism—retaining a subset of top-performing chromosomes in each generation—ensured that high-quality traits persisted, fostering incremental improvements without sacrificing diversity. Our Mutation and Crossover operators, as well as the function to initialize chromosomes was carefully experimented using preexisting techniques to come up with our own modified functions to yield better results. We experimented with different selection schemes and found tournament selection (with a tournament size of 4) to be particularly effective, striking a balance between exploring the search space and honing in on promising solutions, thus successfully progressing toward a recognizably evolved "Mona Lisa."

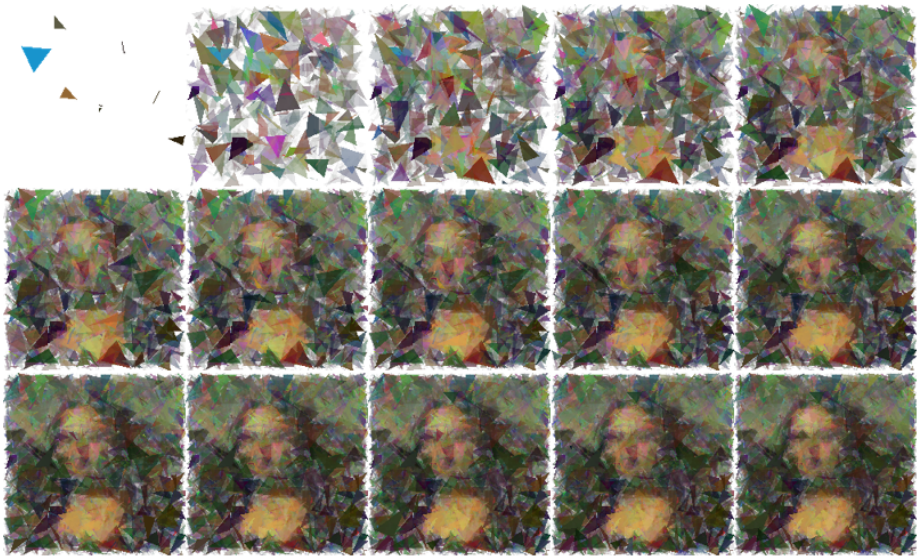


Figure 3: A - Mona Lisa Evolution - Every 100 generations - Left to Right

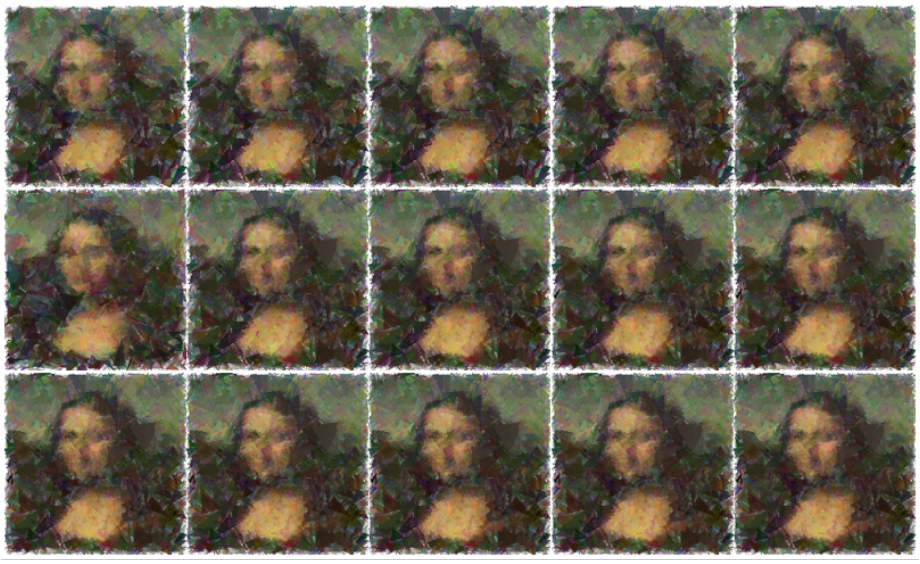


Figure 4: B - Mona Lisa Evolution - Every 100 generations - Left to Right

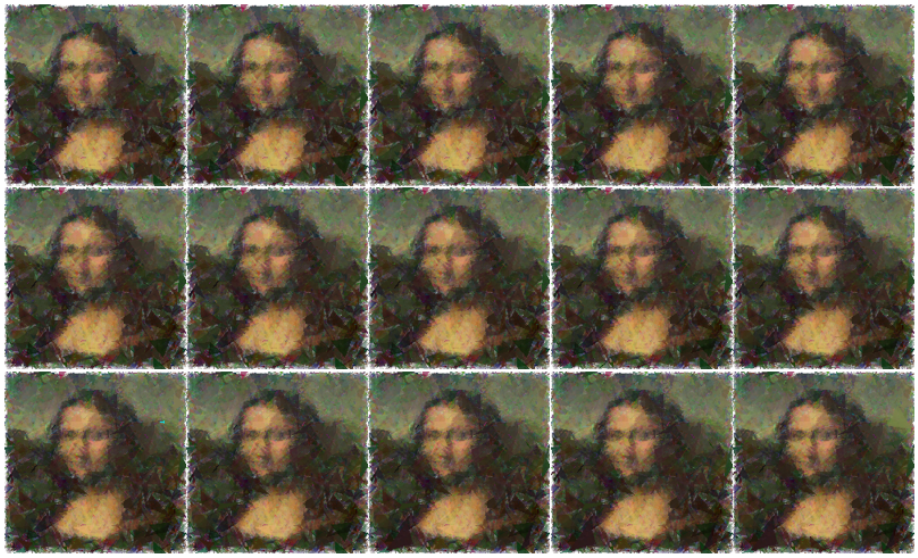


Figure 5: C - Mona Lisa Evolution - Every 100 generations - Left to Right

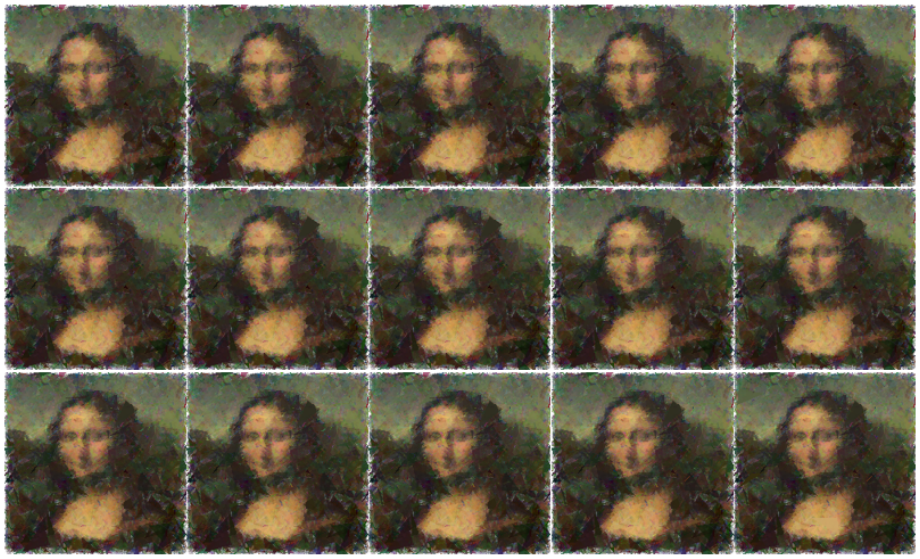


Figure 6: D - Mona Lisa Evolution - Every 100 generations - Left to Right

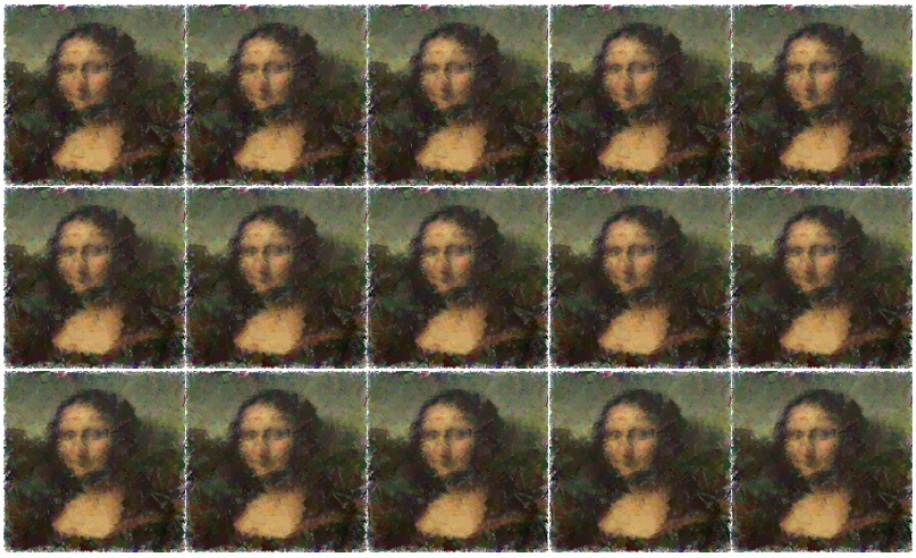


Figure 7: E - Mona Lisa Evolution - Every 100 generations - Left to Right

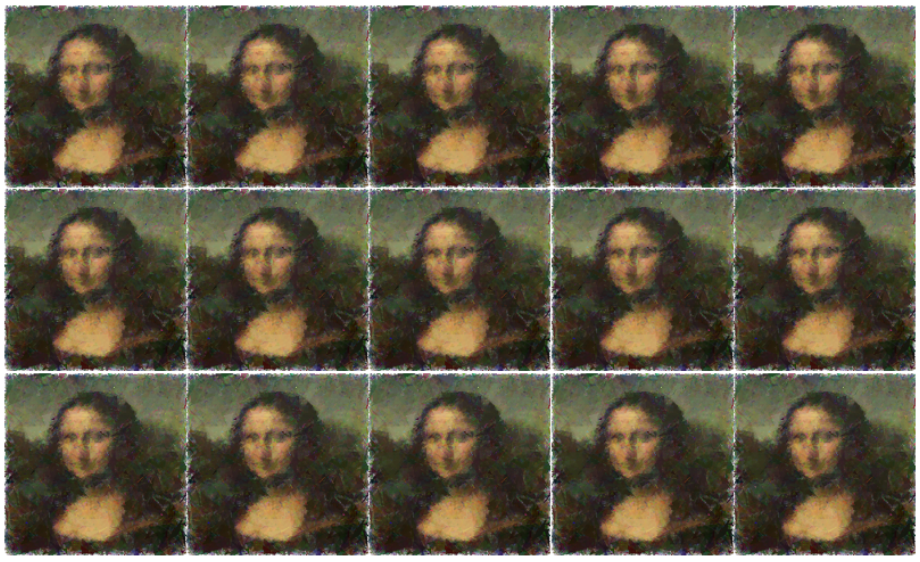


Figure 8: F - Mona Lisa Evolution - Every 100 generations - Left to Right

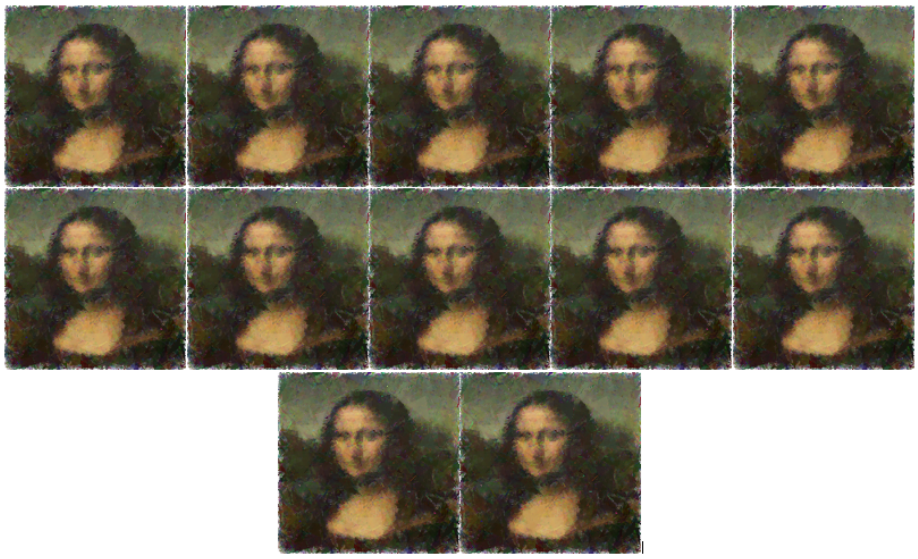


Figure 9: G - Mona Lisa Evolution - Every 100 generations - Left to Right