Universitat Rovira i Virgili

Assignment 2

Neural and Evolutionary Computation

Optimization with Genetic Algorithms

Biel Foradada Jiménez

February 1, 2026

UNIVERSITAT
ROVIRA i VIRGILI

Contents

1 Chromosome Description and Algorithm Adaptations

1.1 Chromosome Representation

Each chromosome is a permutation of job IDs, where each ID appears exactly n times (corresponding to the number of machines/operations per job). For example, in a problem with 3 jobs and 2 machines, a chromosome might look like:

$$[0, 1, 0, 2, 1, 2]$$

The k-th occurrence of a job ID in the list refers to the k-th operation of that specific job.

1.2 Genetic Operators and Adaptations

The techniques for crossover and mutation are applied to the childs of a generation with a probability defined by constants in the code `(CROSSOVER_PROB, MUTATION_PROB)`, which can be edited two tweak the results. Given that the mutation or the crossover is being applied, each of both types for mutation and crossover has a 50% probability of happening.

1.2.1 Selection Techniques

I used two distinct selection mechanisms:

- **Tournament Selection:** For each child to be created, three random individual chromosomes are sampled from the population. The one with the lowest total time is selected.
- **Elitism:** To ensure the best solutions are never lost due to random mutation or crossover, the two chromosomes with the lowest total time are copied into the next generation.

1.2.2 Crossover Techniques

Since standard crossover[1] could result in an invalid number of job occurrences, I used two specific crossover techniques for this problem:

- **Position-Based Crossover:** A random subset of positions is selected from the first parent and inherited by the child. The remaining empty spots are filled using job IDs from the second parent in the order they appear, always keeping track of the number of instances of the job to not exceed the limit.
- **Precedence Operation Crossover:** A subset of job IDs is selected. All instances of these jobs are copied from Parent 1 in their exact positions. The remaining gaps are filled using only the non-selected job IDs from Parent 2.

1.2.3 Mutation Techniques

I used two methods for the mutation techniques:

- **Swap Mutation:** Two random indices in the chromosome are selected, and their job IDs are swapped.
- **Inversion Mutation:** A random segment of the chromosome is selected and its order is entirely reversed.

[1] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

1.3 Population Size and Stationary State

I set the population size to 200, I chose this value only to balance out performance in computation time and solution quality. The stationary state can be reached in two ways, one is reaching the hard-coded limit of generations (which I set up to 600), or by the code tracking that the best total time of the population has not lowered in 100 generations, value which can be also modified.

2 Results of executions

I used three JSP problems of the provided link in the description of the assginment[2]. The first problem has 20 jobs and 5 machines, the second one 10 jobs and 10 machines, and the third one 20 jobs and 20 machines.

2.1 Testing Graph Results

Low diversity

For this test the parameters where set to:

- `MUTATION_PROB = 0.1`
- `CROSSOVER_PROB = 0.1`
- `POPULATION_SIZE = 50`

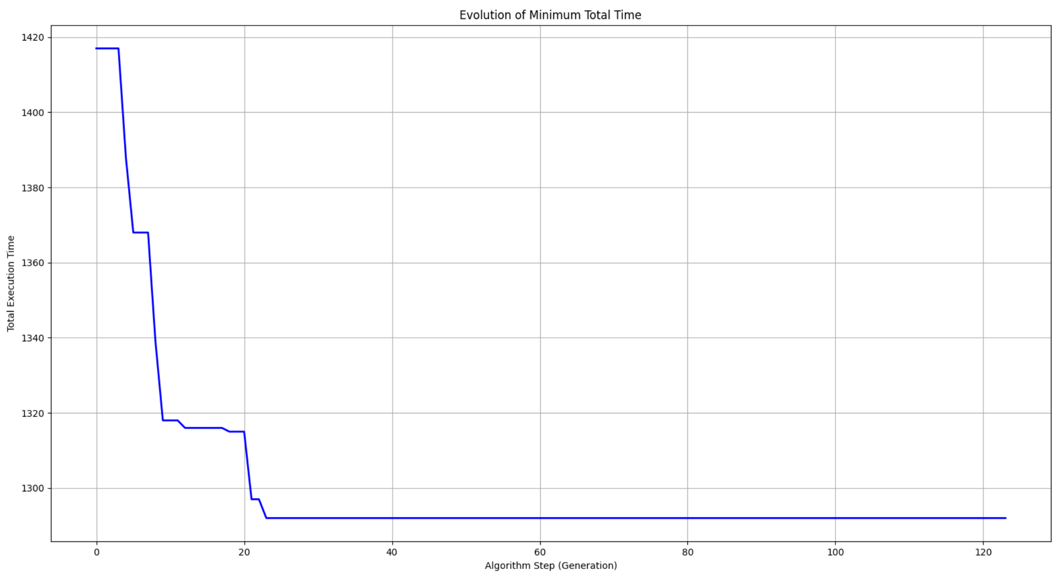

Figure 1: Evolution of the best time for the 20 jobs and 5 machines problem.

[2]https://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt

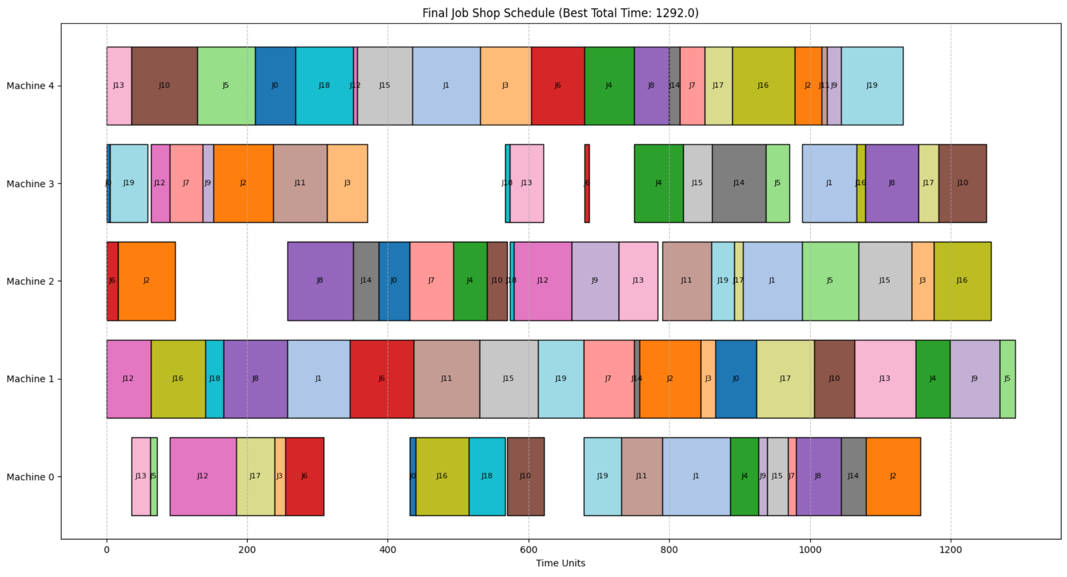

Figure 2: Gantt scheme for the 20 jobs and 5 machines problem.

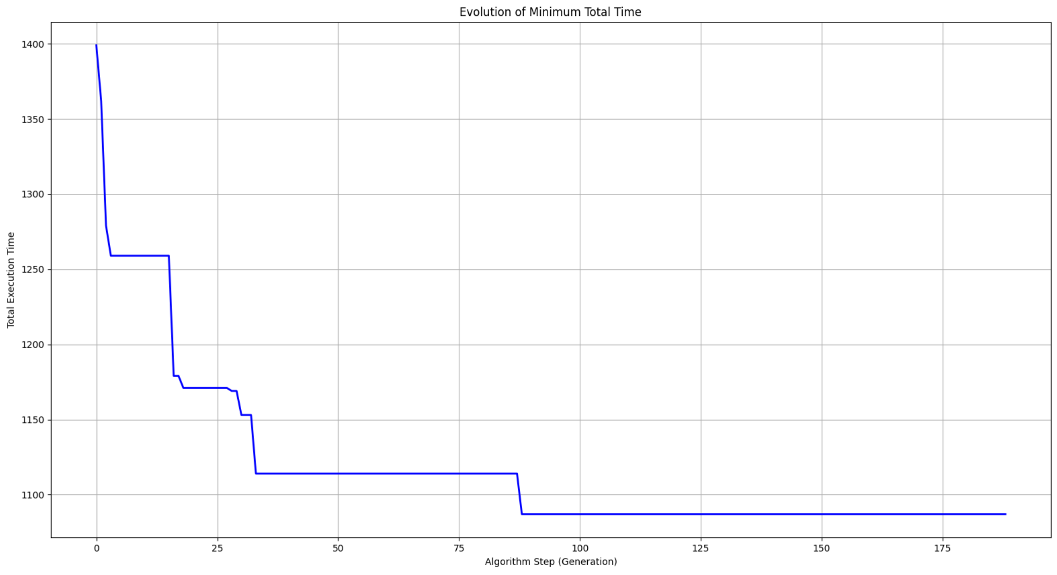

Figure 3: Evolution of the best time for the 10 jobs and 10 machines problem.

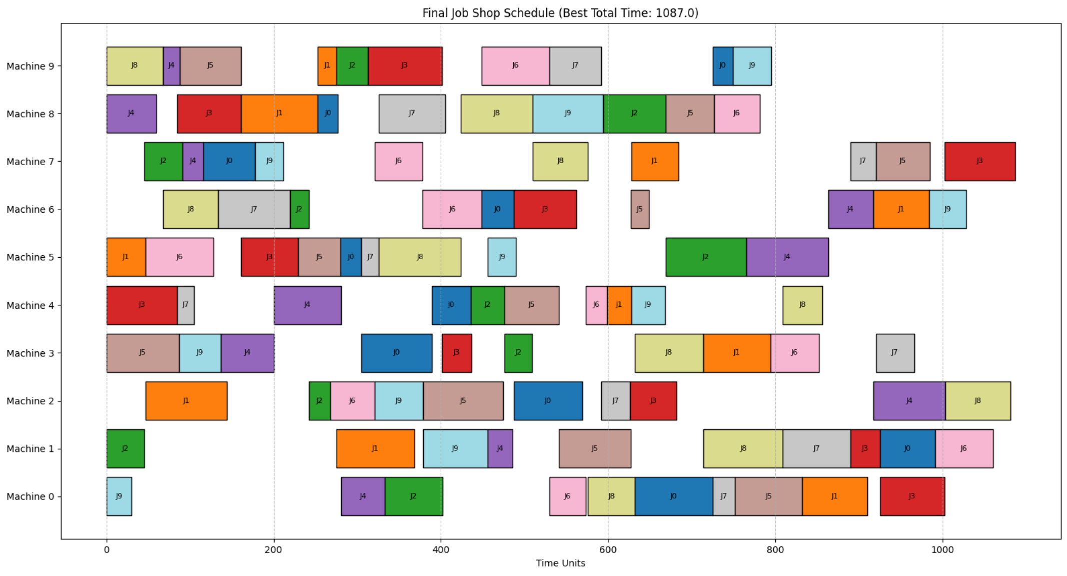

Figure 4: Gantt scheme for the 10 jobs and 10 machines problem.

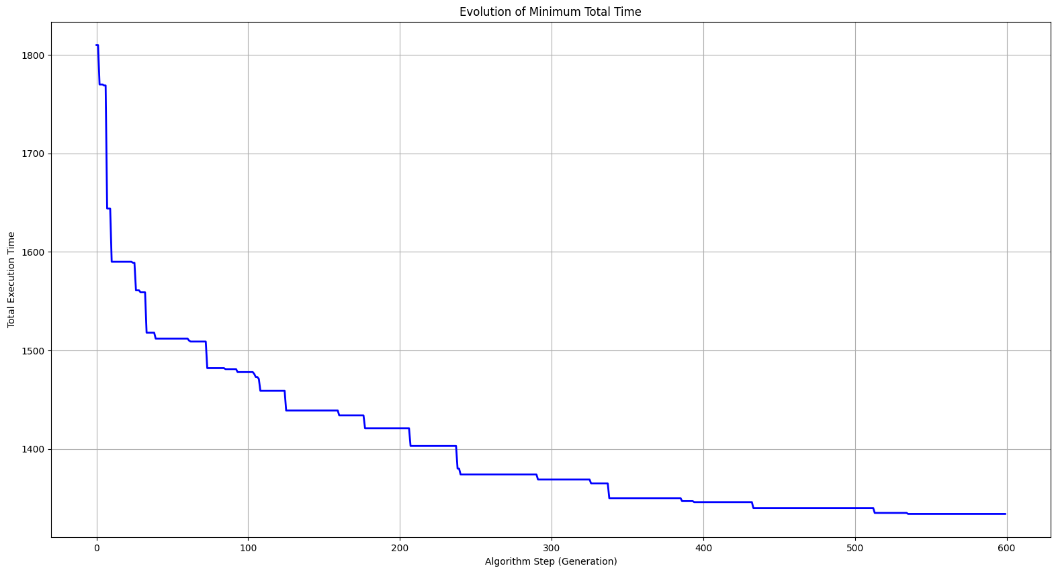

Figure 5: Evolution of the best time for the 20 jobs and 20 machines problem.

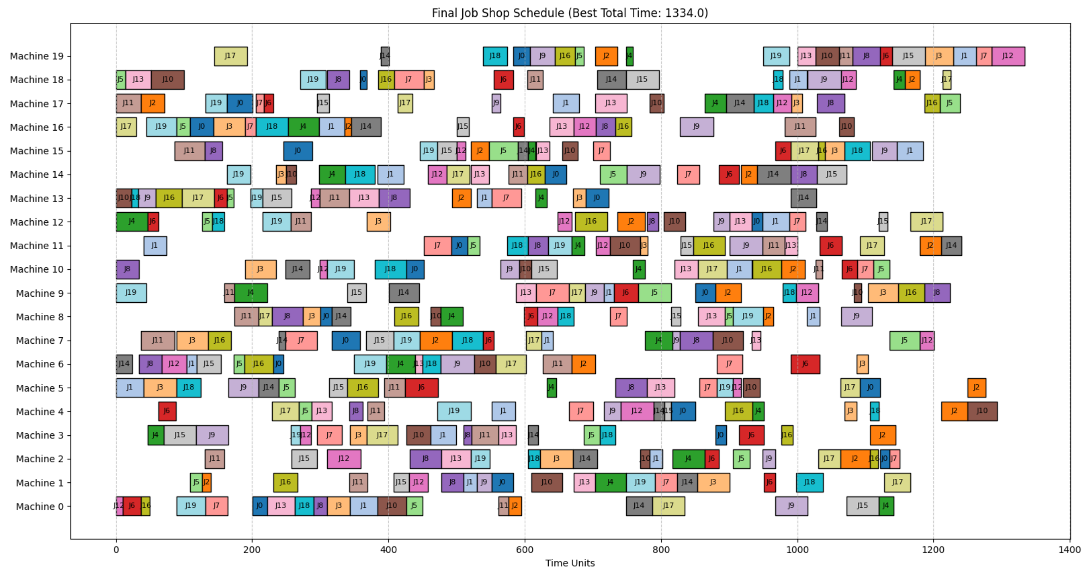

Figure 6: Gantt scheme for the 20 jobs and 20 machines problem.

Baseline

For this test the parameters where set to:

- `MUTATION_PROB = 0.4`
- `CROSSOVER_PROB = 0.2`
- `POPULATION_SIZE = 200`

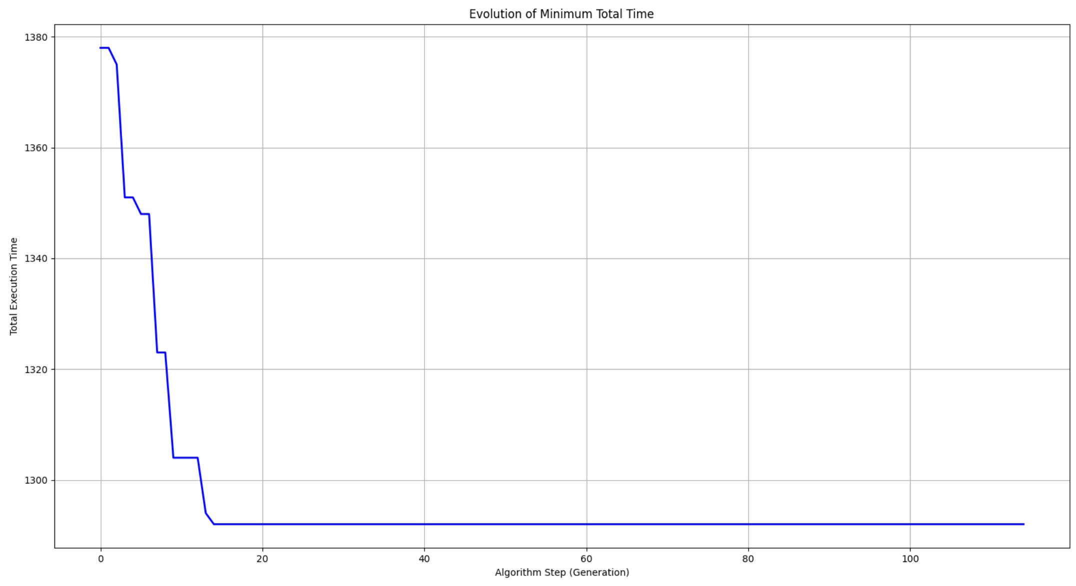

Figure 7: Evolution of the best time for the 20 jobs and 5 machines problem.

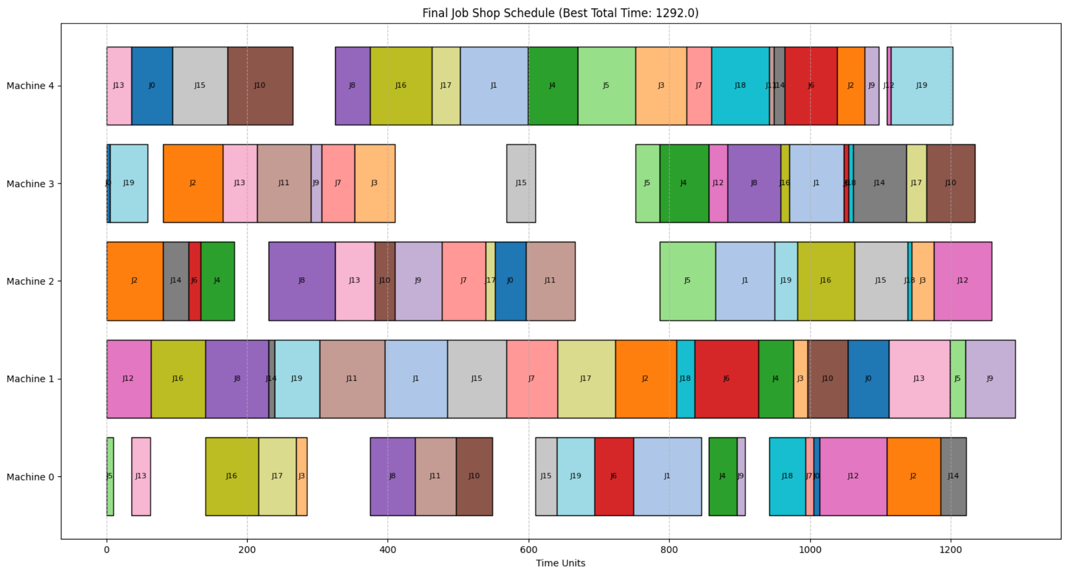

Figure 8: Gantt scheme for the 20 jobs and 5 machines problem.

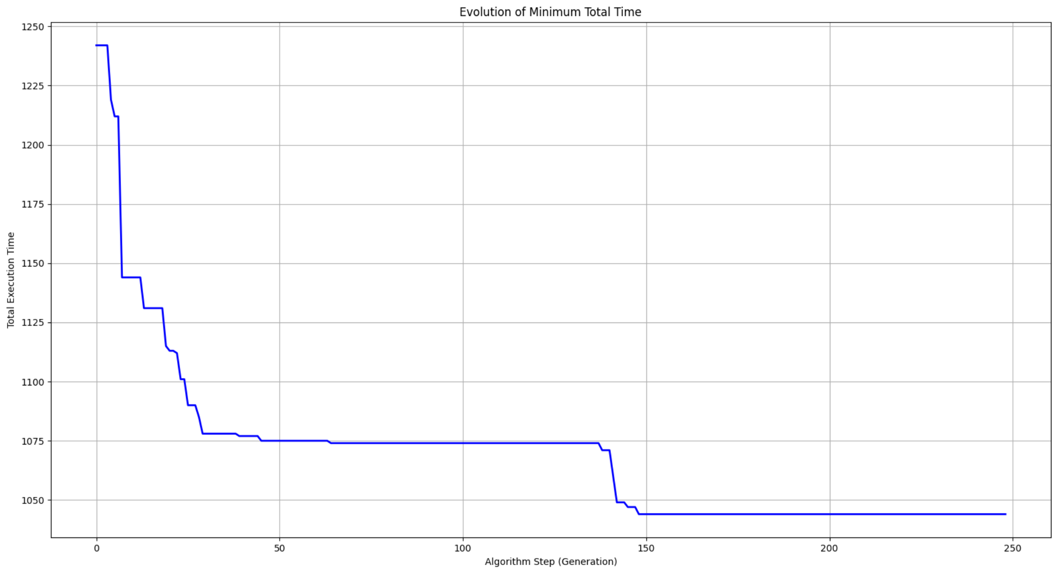

Figure 9: Evolution of the best time for the 10 jobs and 10 machines problem.

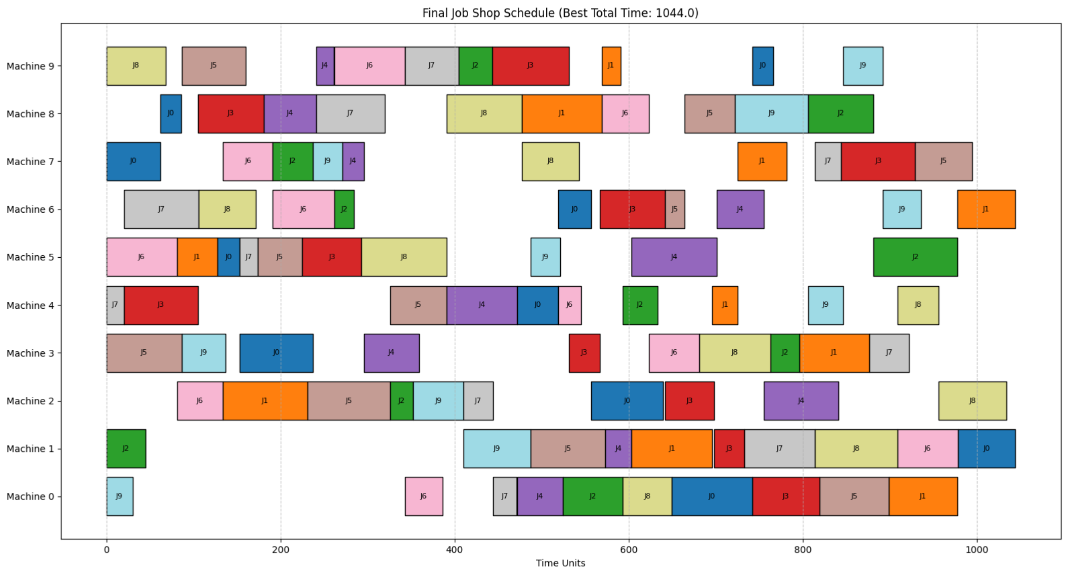

Figure 10: Gantt scheme for the 10 jobs and 10 machines problem.

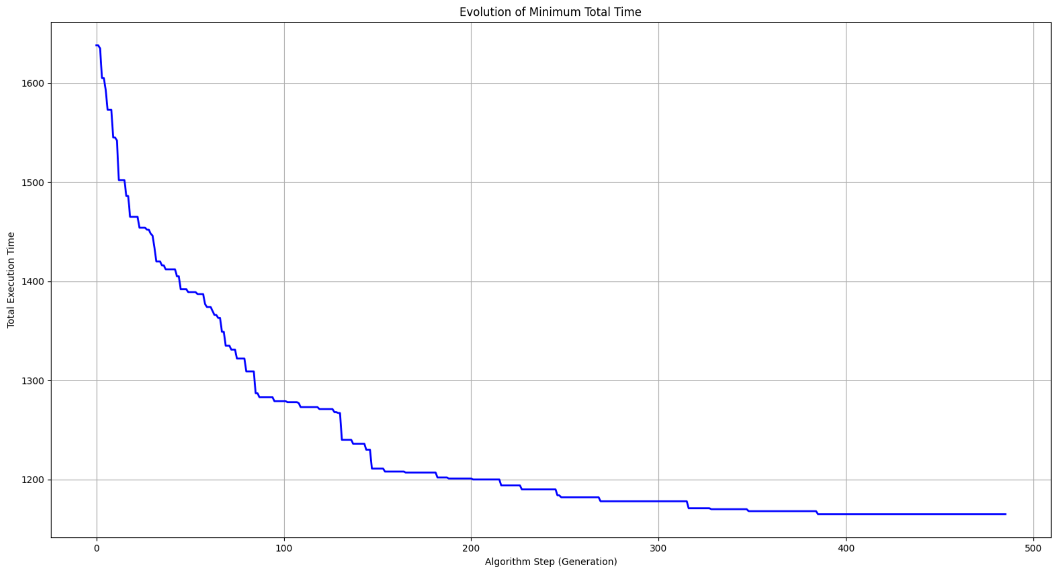

Figure 11: Evolution of the best time for the 20 jobs and 20 machines problem.

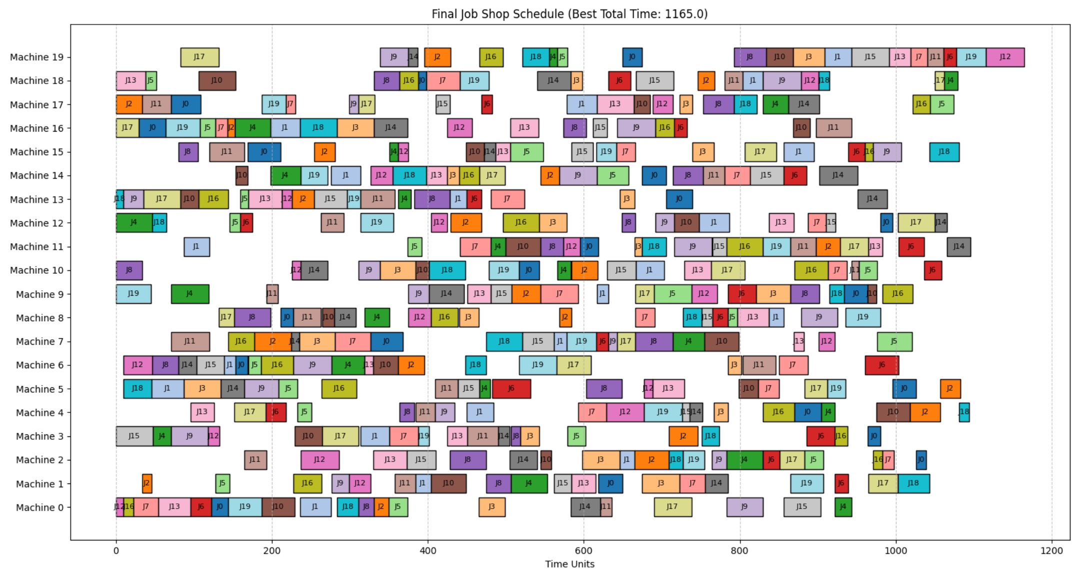

Figure 12: Gantt scheme for the 20 jobs and 20 machines problem.

High Exploration

For this test the parameters where set to:

- `MUTATION_PROB = 0.8`
- `CROSSOVER_PROB = 0.4`
- `POPULATION_SIZE = 200`

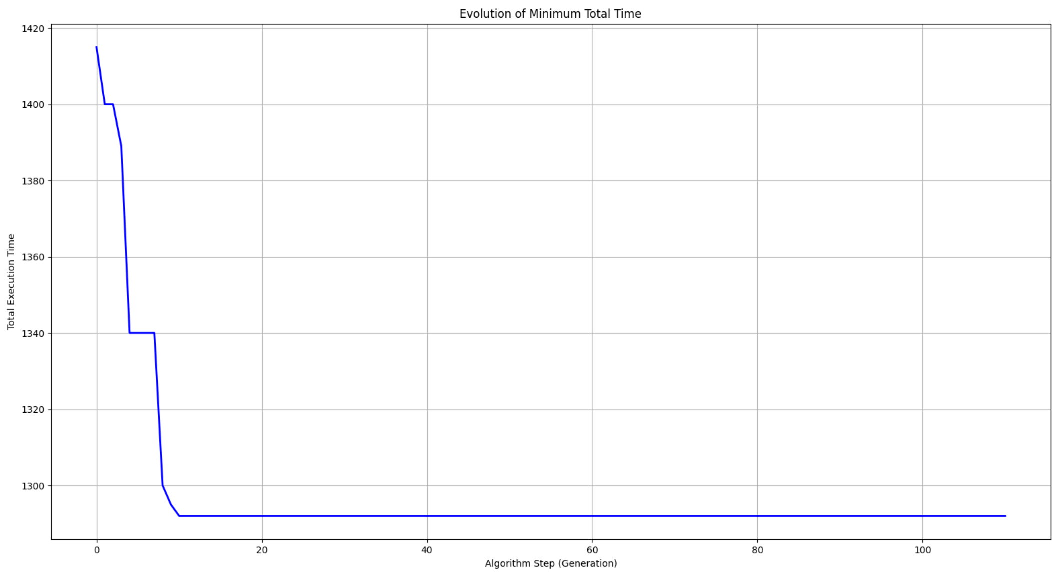

Figure 13: Evolution of the best time for the 20 jobs and 5 machines problem.

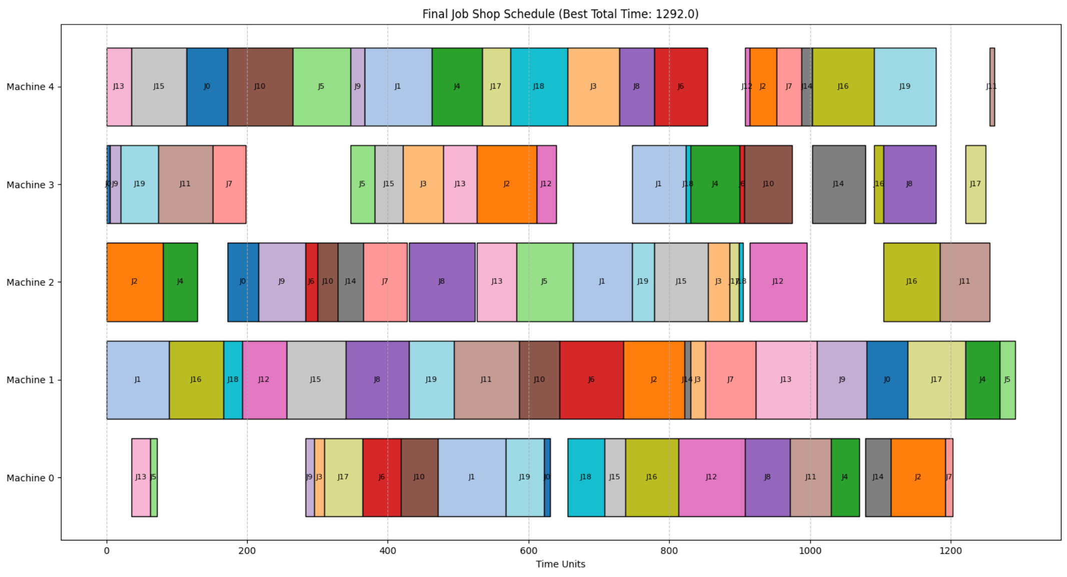

Figure 14: Gantt scheme for the 20 jobs and 5 machines problem.

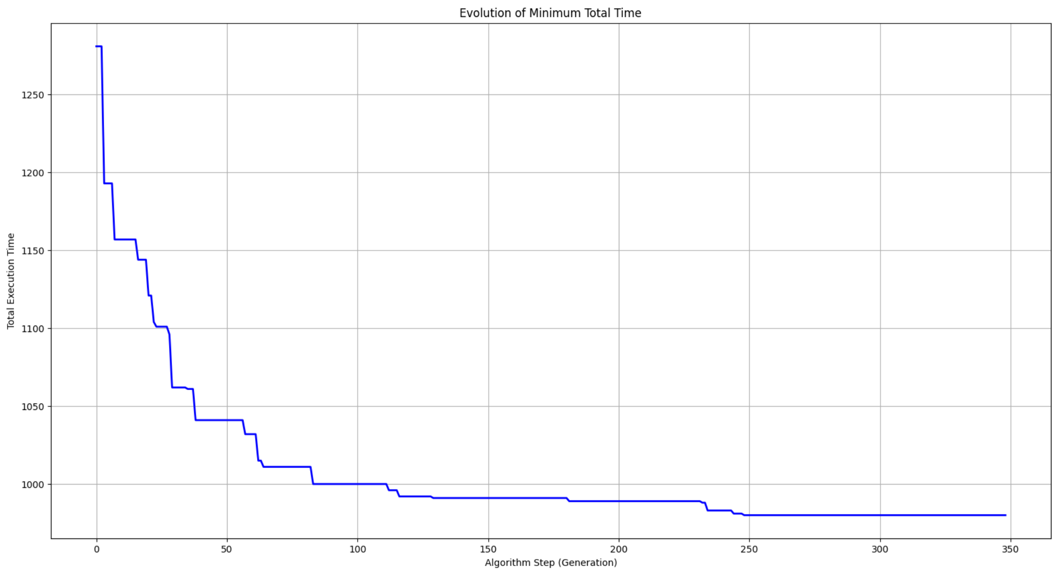

Figure 15: Evolution of the best time for the 10 jobs and 10 machines problem.

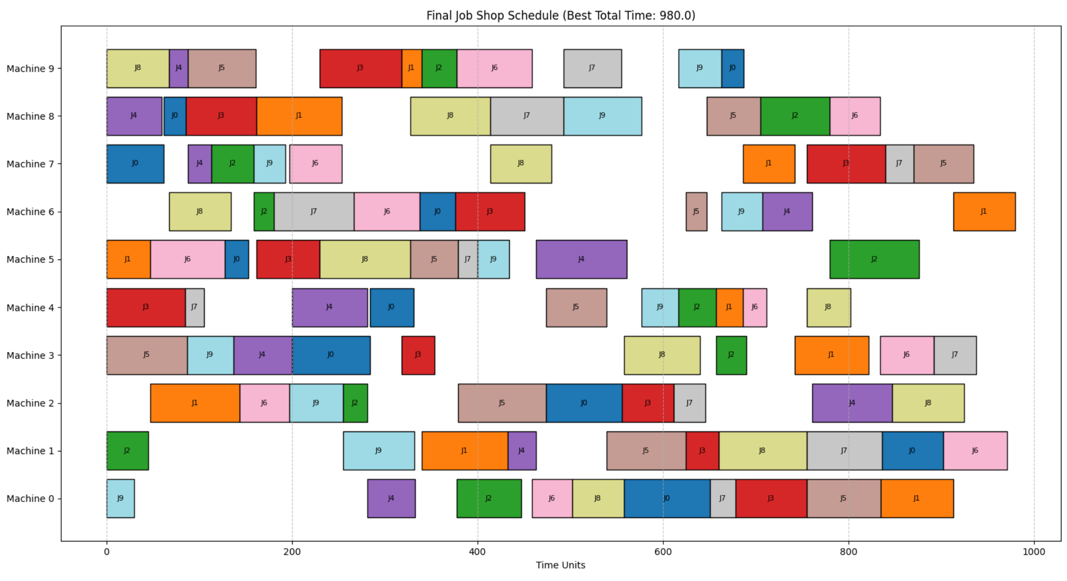

Figure 16: Gantt scheme for the 10 jobs and 10 machines problem.

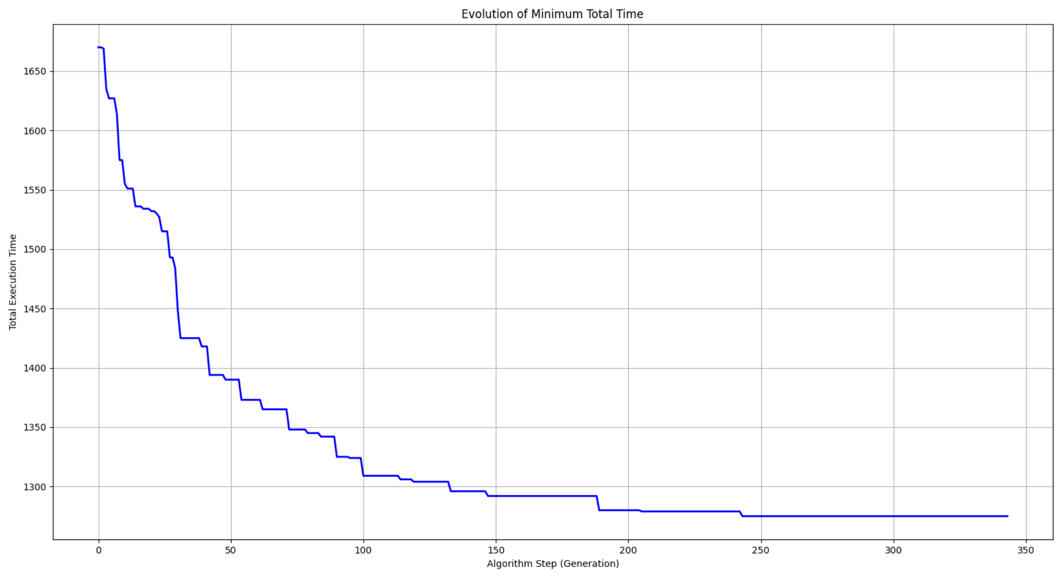

Figure 17: Evolution of the best time for the 20 jobs and 20 machines problem.

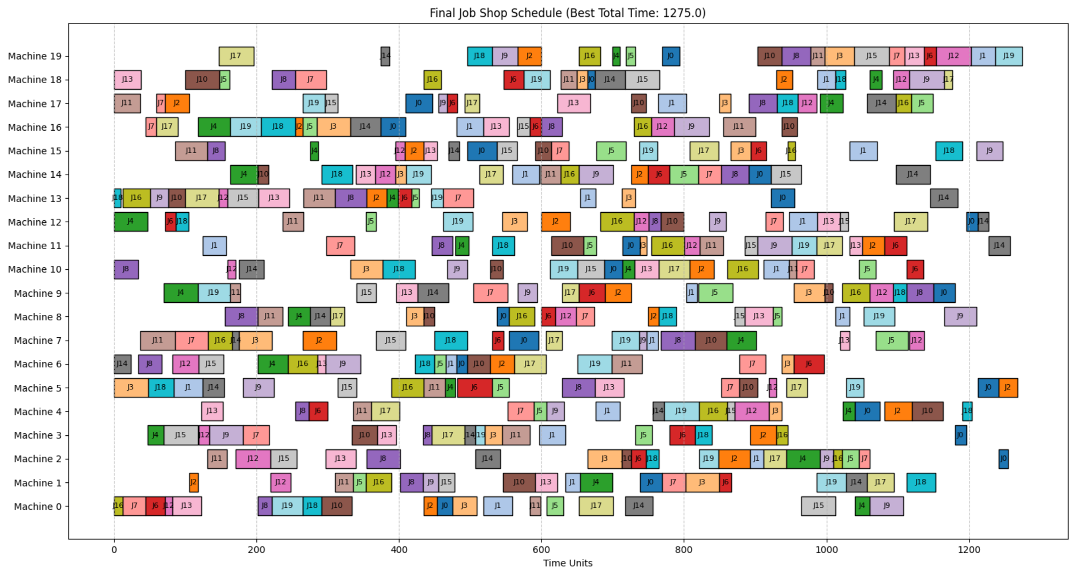

Figure 18: Gantt scheme for the 20 jobs and 20 machines problem.

Brute Force

For this test the parameters where set to:

- `MUTATION_PROB = 0.3`
- `CROSSOVER_PROB = 0.3`
- `POPULATION_SIZE = 500`

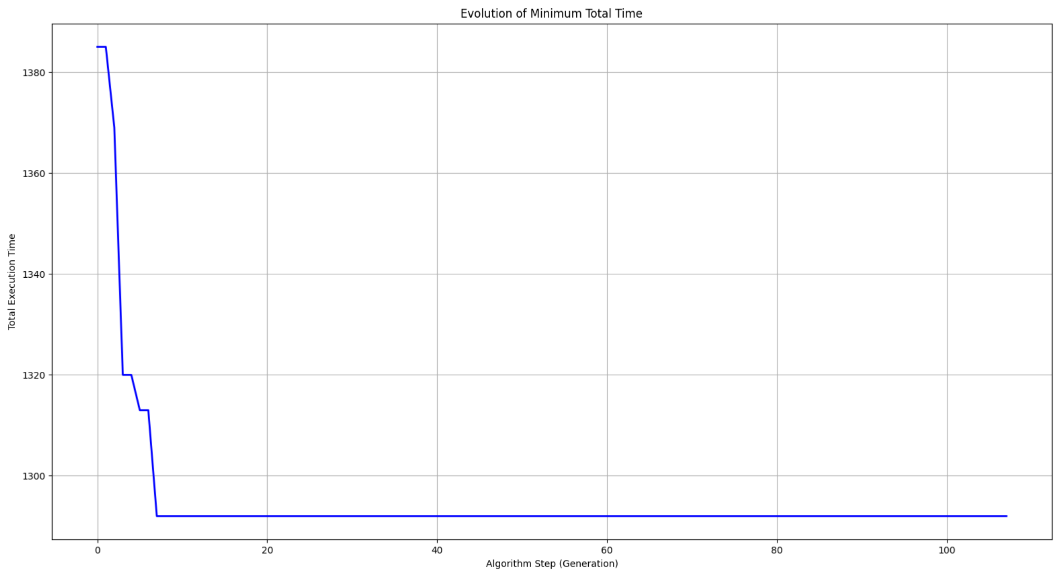

Figure 19: Evolution of the best time for the 20 jobs and 5 machines problem.

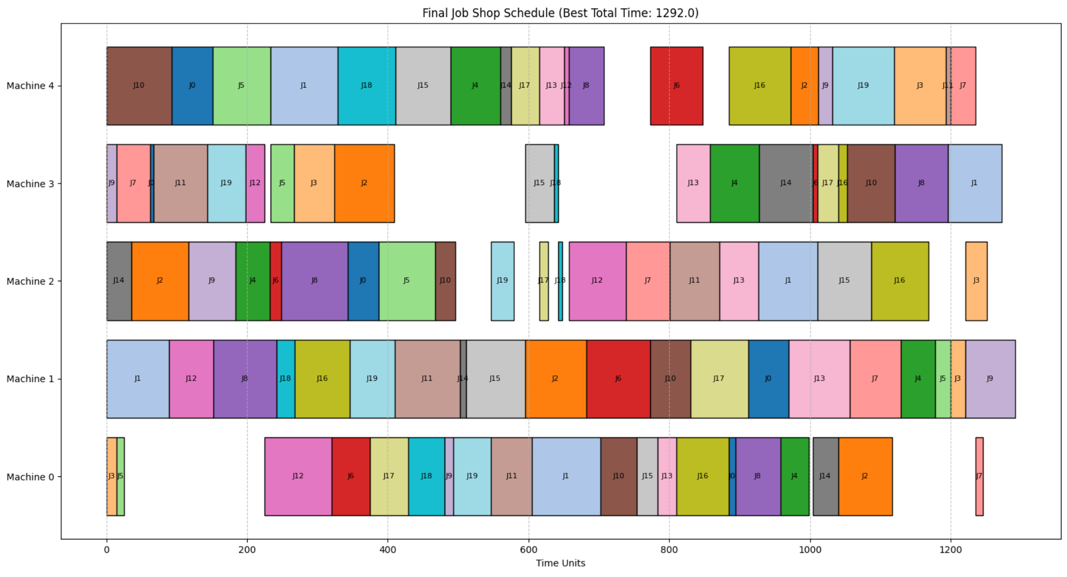

Figure 20: Gantt scheme for the 20 jobs and 5 machines problem.

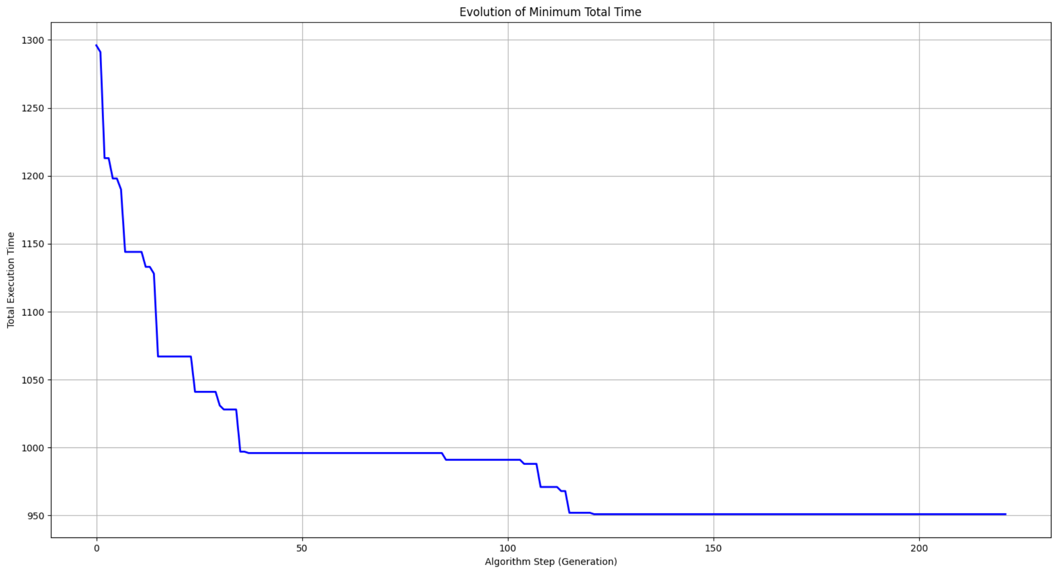

Figure 21: Evolution of the best time for the 10 jobs and 10 machines problem.

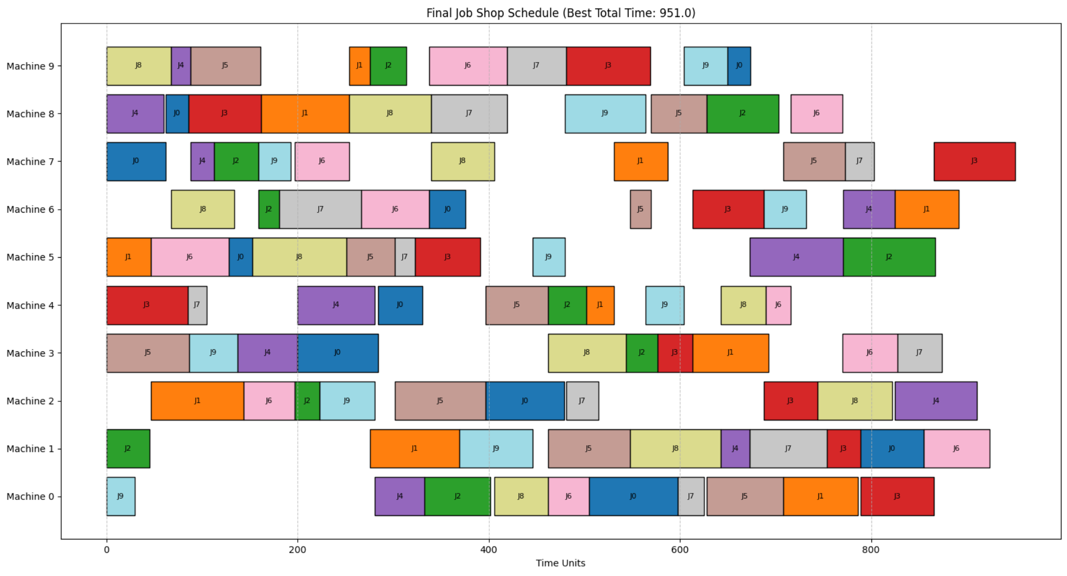

Figure 22: Gantt scheme for the 10 jobs and 10 machines problem.

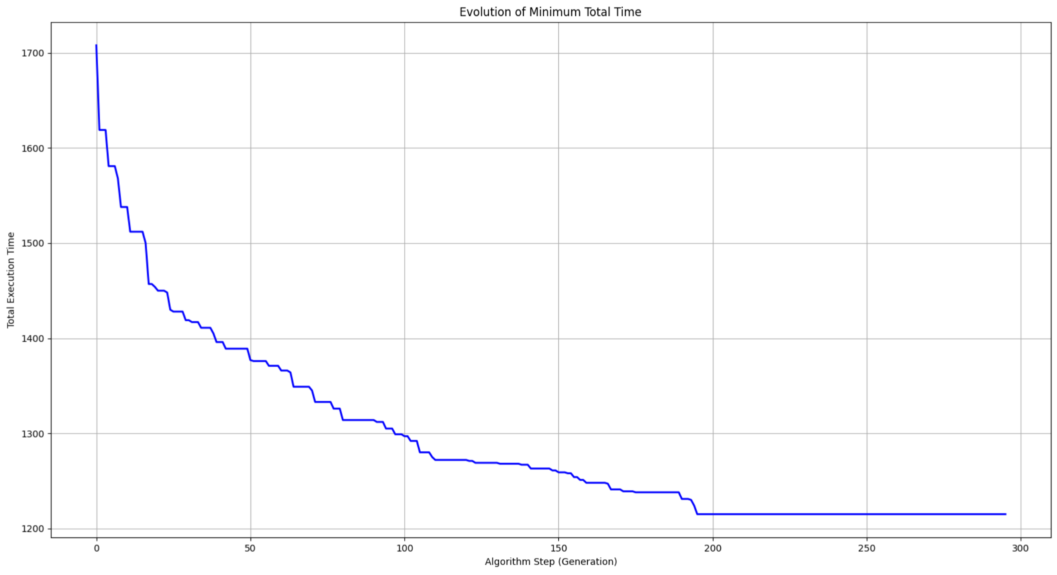

Figure 23: Evolution of the best time for the 20 jobs and 20 machines problem.

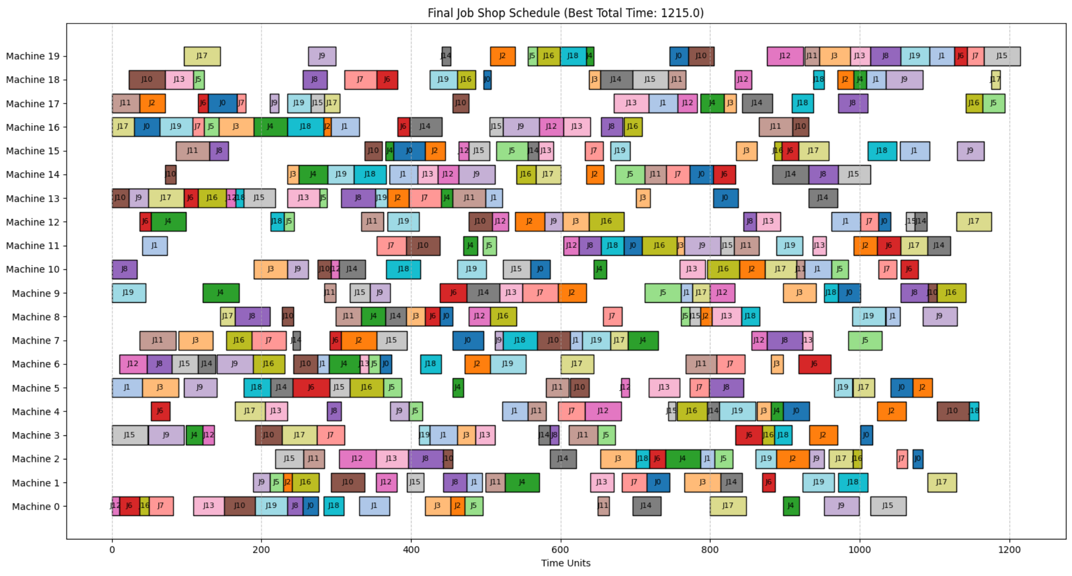

Figure 24: Gantt scheme for the 20 jobs and 20 machines problem.

Crossover Heavy

For this test the parameters where set to:

- `MUTATION_PROB = 0.1`
- `CROSSOVER_PROB = 0.8`
- `POPULATION_SIZE = 200`

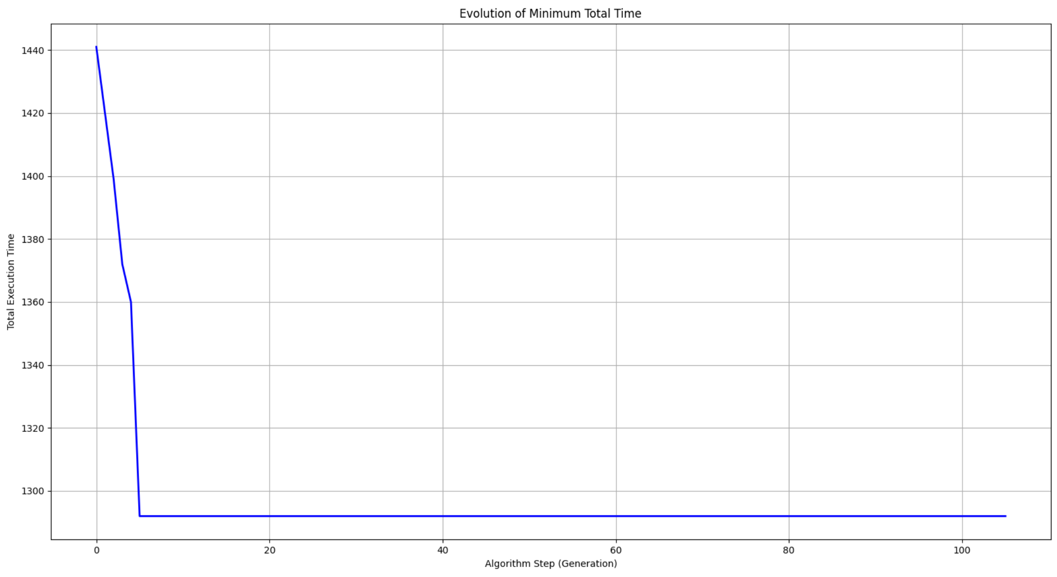

Figure 25: Evolution of the best time for the 20 jobs and 5 machines problem.

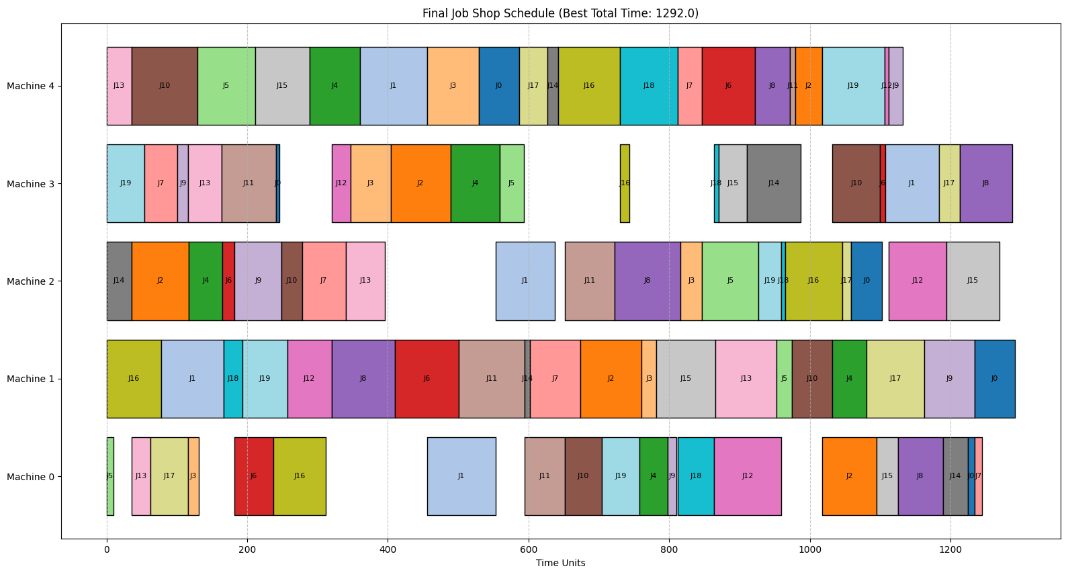

Figure 26: Gantt scheme for the 20 jobs and 5 machines problem.

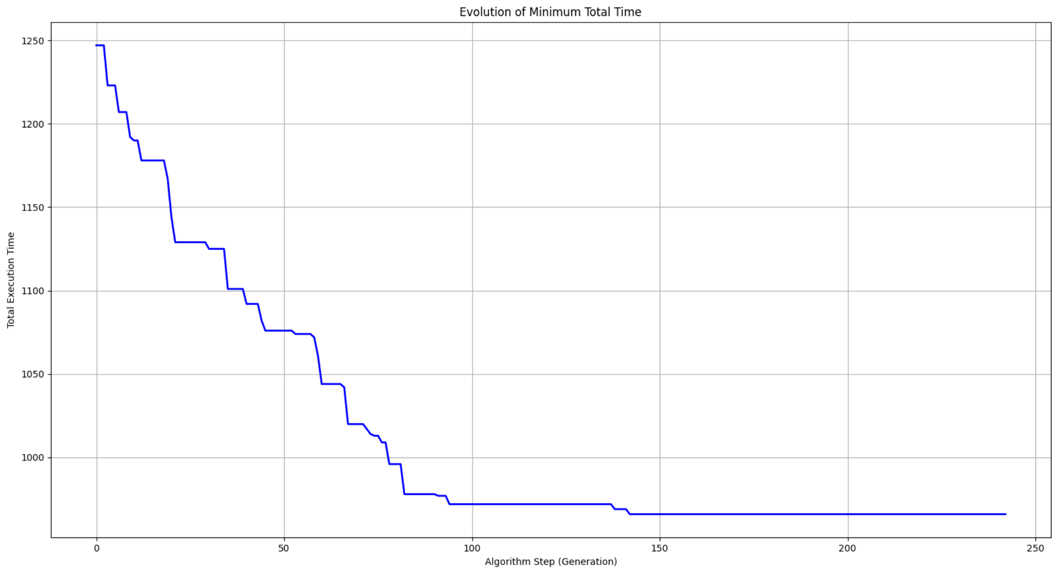

Figure 27: Evolution of the best time for the 10 jobs and 10 machines problem.

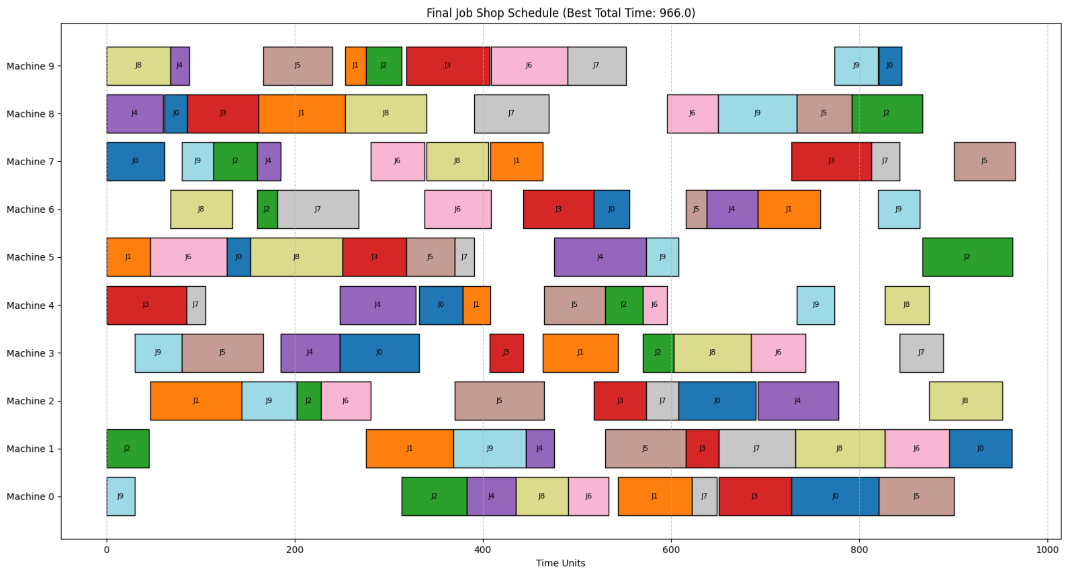

Figure 28: Gantt scheme for the 10 jobs and 10 machines problem.

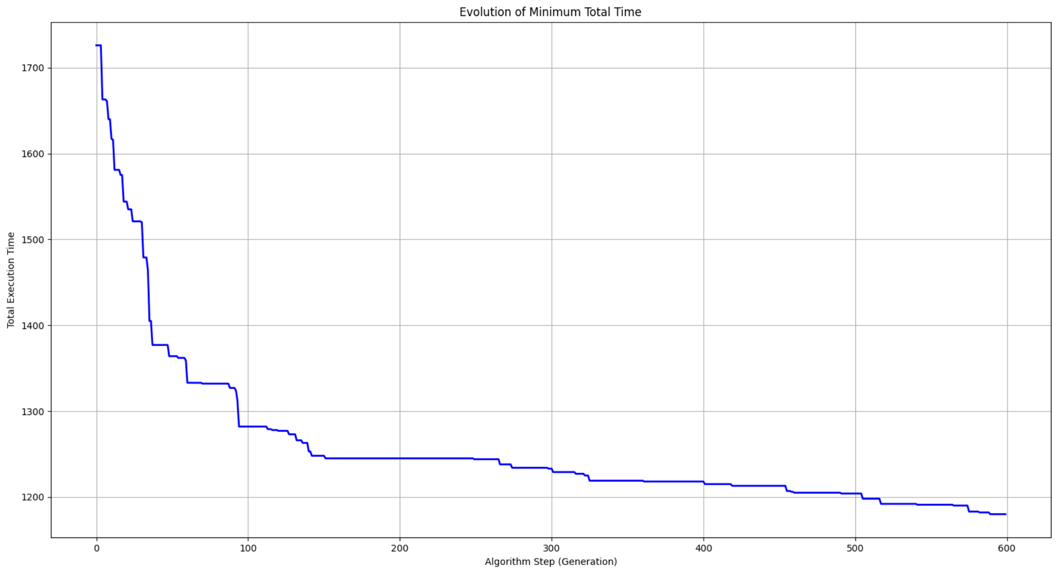

Figure 29: Evolution of the best time for the 20 jobs and 20 machines problem.

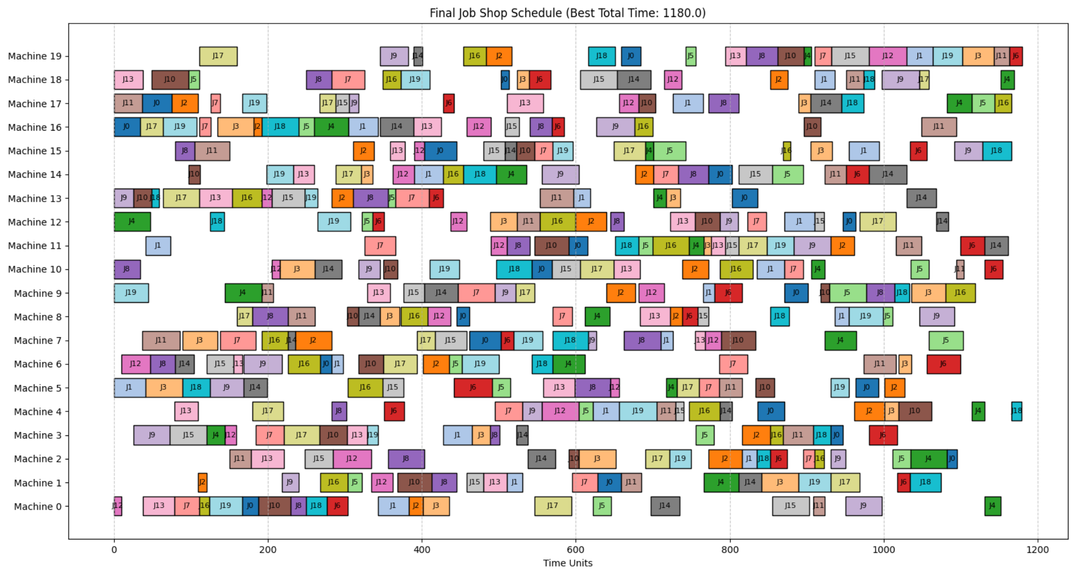

Figure 30: Gantt scheme for the 20 jobs and 20 machines problem.

Mutation Heavy

For this test the parameters where set to:

- `MUTATION_PROB = 0.8`
- `CROSSOVER_PROB = 0.1`
- `POPULATION_SIZE = 200`

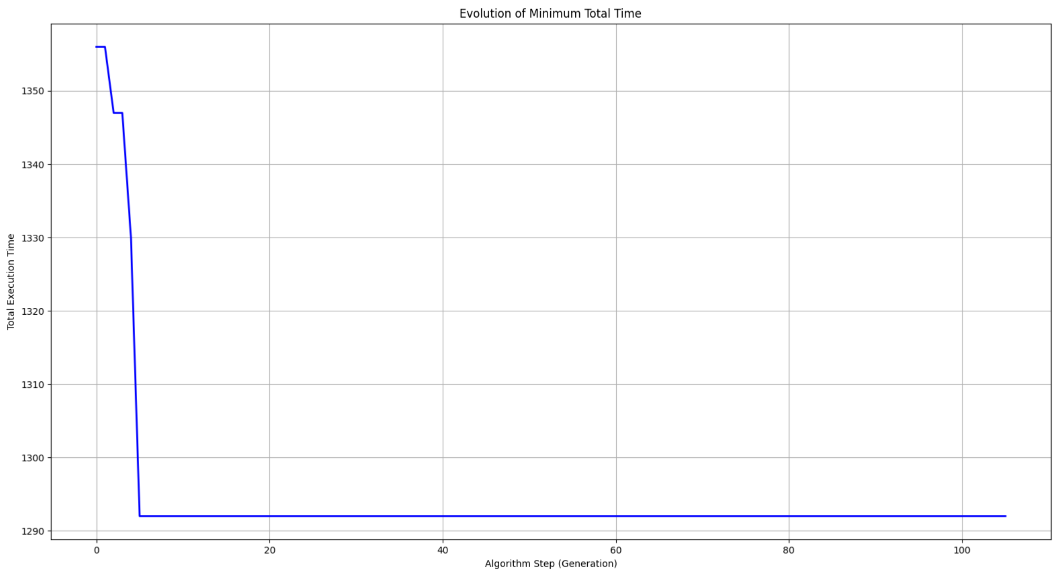

Figure 31: Evolution of the best time for the 20 jobs and 5 machines problem.

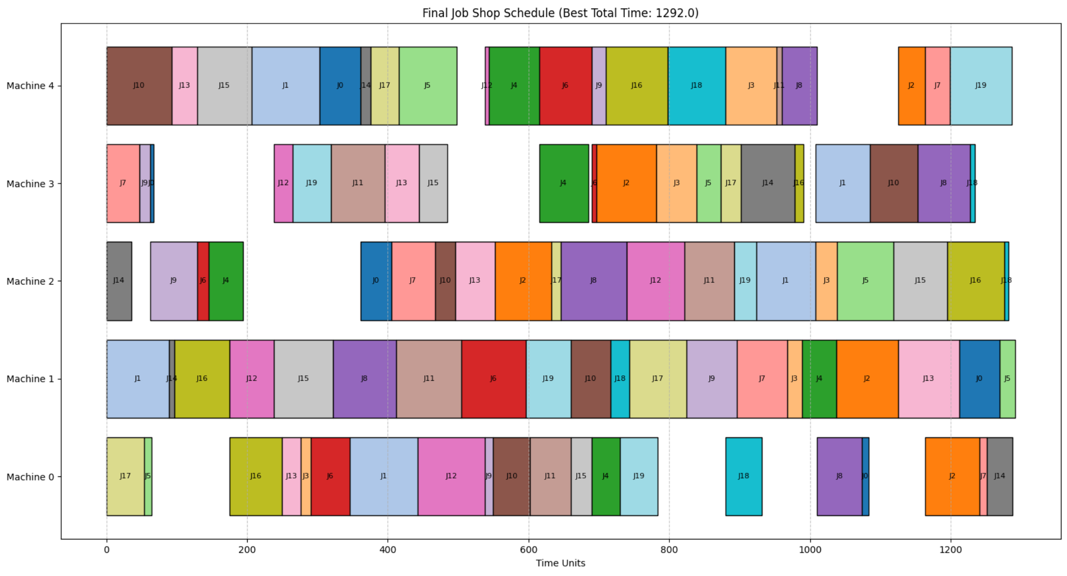

Figure 32: Gantt scheme for the 20 jobs and 5 machines problem.

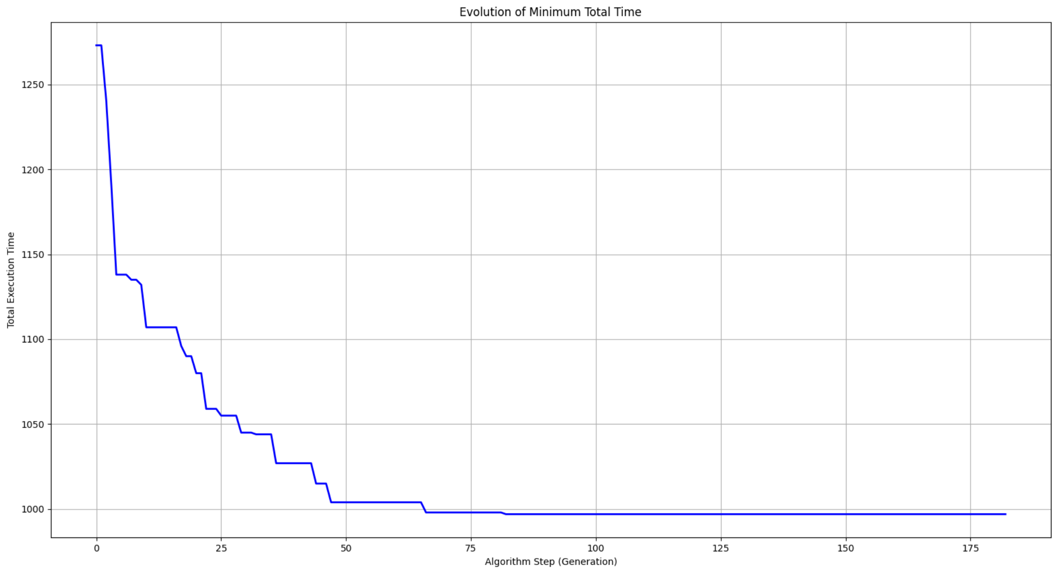

Figure 33: Evolution of the best time for the 10 jobs and 10 machines problem.

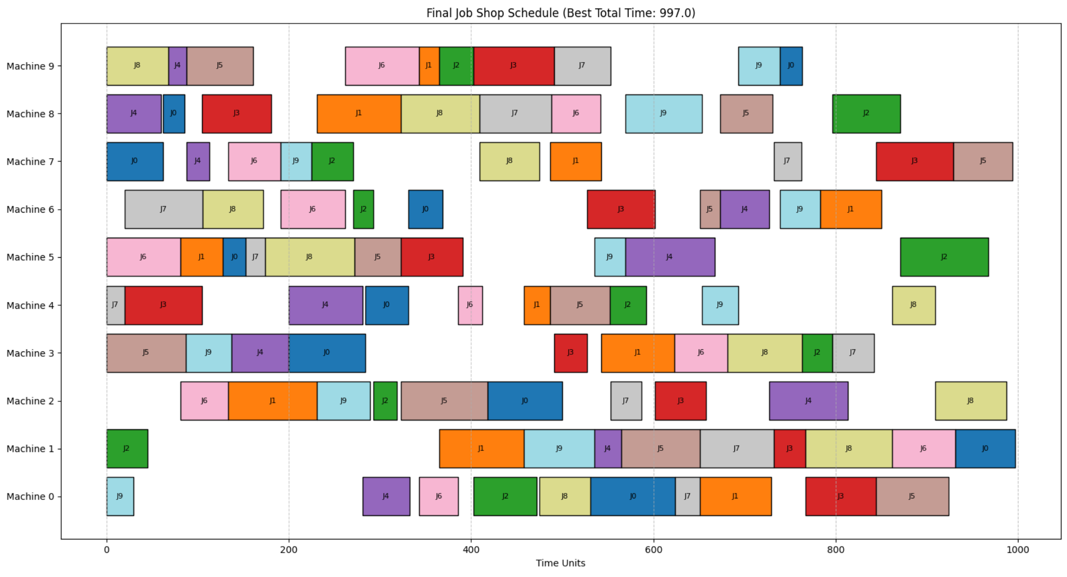

Figure 34: Gantt scheme for the 10 jobs and 10 machines problem.

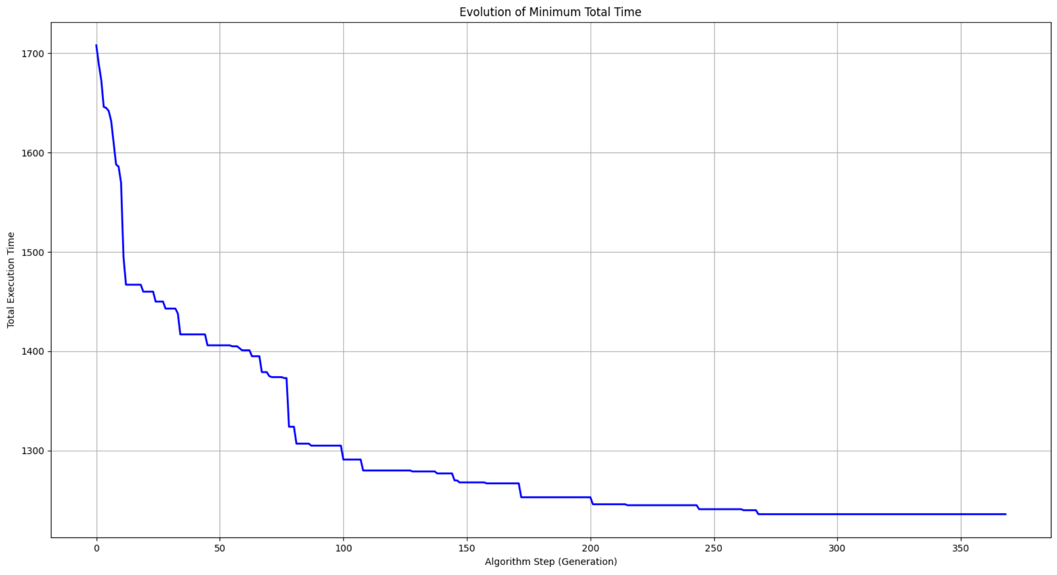

Figure 35: Evolution of the best time for the 20 jobs and 20 machines problem.

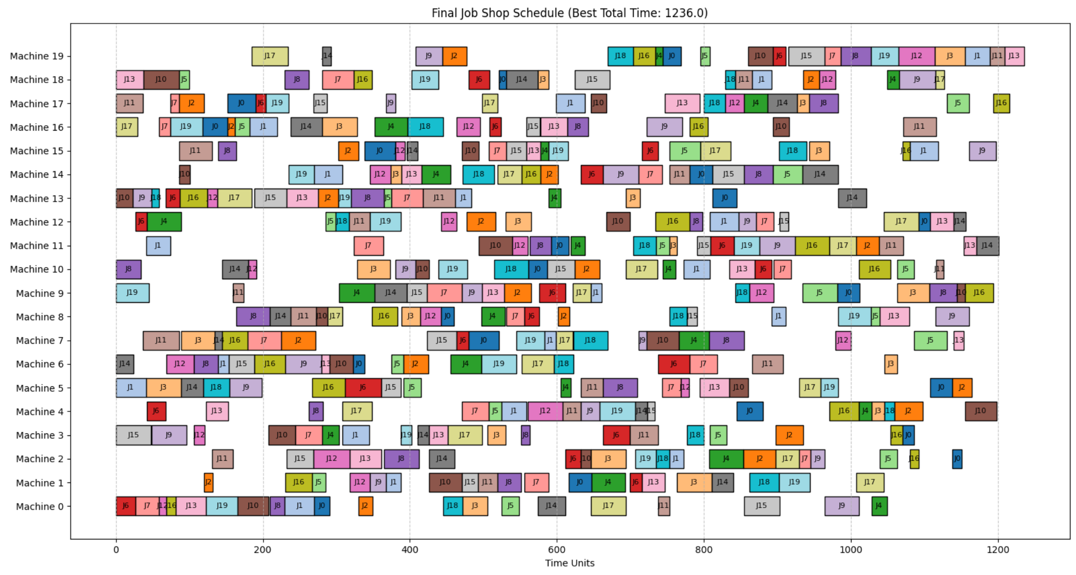

Figure 36: Gantt scheme for the 20 jobs and 20 machines problem.

2.2 Summary Results

The following table summarizes the minimum total time obtained across three different problem instances (20x5, 10x10, and 20x20) using the six parameter combinations described in the previous section.

Configuration	**20 Jobs × 5 Mach.**	**10 Jobs × 10 Mach.**	**20 Jobs × 20 Mach.**
Low Diversity	1292	1087	1334
Baseline	1292	1044	**1165**
High Exploration	1292	980	1275
Brute Force	1292	**951**	1215
Crossover Heavy	1292	966	1180
Mutation Heavy	1292	997	1236

Table 1: Total time results for different parameter combinations across three problem instances.

As observed in the table above, for the problem with 5 machines, all experiments arrived to the same time solution. For the 10x10 problem it was the brute force solution, where a big population size was combined with smaller probability of mutations (Figure 21 and 22). For the problem of 20x20, the baseline parameters were the ones that provided the best result (Figure 11 and 12).

2.3 Files used

2.3.1 20 jobs and 5 machines

```
20 5
3  5 4 58 2 44 0  9 1 58
1 89 4 96 0 97 2 84 3 77
2 81 3 85 1 87 4 39 0 77
```

```
0 15 3 57 4 73 1 21 2 31
2 48 4 71 3 70 0 40 1 49
0 10 4 82 3 34 2 80 1 22
2 17 0 55 1 91 4 75 3  7
3 47 2 62 1 72 4 35 0 11
1 90 2 94 4 50 0 64 3 75
3 15 2 67 0 12 4 20 1 71
4 93 2 29 0 52 1 57 3 68
3 77 1 93 0 58 2 70 4  7
1 63 3 27 0 95 4  6 2 82
4 36 0 26 3 48 2 56 1 87
2 36 1  8 4 15 3 76 0 36
4 78 1 84 3 41 0 30 2 76
1 78 0 75 4 88 3 13 2 81
0 54 4 40 2 13 1 82 3 29
1 26 4 82 0 52 3  6 2  6
3 54 1 64 0 54 2 32 4 88
```

2.3.2 10 jobs and 10 machines

```
10 10
7 62 8 24 5 25 3 84 4 47 6 38 2 82 0 93 9 24 1 66
5 47 2 97 8 92 9 22 1 93 4 29 7 56 3 80 0 78 6 67
1 45 7 46 6 22 2 26 9 38 0 69 4 40 3 33 8 75 5 96
4 85 8 76 5 68 9 88 3 36 6 75 2 56 1 35 0 77 7 85
8 60 9 20 7 25 3 63 4 81 0 52 1 30 5 98 6 54 2 86
3 87 9 73 5 51 2 95 4 65 1 86 6 22 8 58 0 80 7 65
5 81 2 53 7 57 6 71 9 81 0 43 4 26 8 54 3 58 1 69
4 20 6 86 5 21 8 79 9 62 2 34 0 27 1 81 7 30 3 46
9 68 6 66 5 98 8 86 7 66 0 56 3 82 1 95 4 47 2 78
0 30 3 50 7 34 2 58 1 77 5 34 8 84 4 40 9 46 6 44
```

2.3.3 20 jobs and 20 machines

```
*This content is located in the file jobs_ran.txt, due to not fitting
inside the document*
```

3 Link GitHub

The link to the GitHub repository is: https://github.com/BiForadada/NEC_2