Lab 3: Evolutionary and Genetic Algorithms

Bora ILCI & Kaan Emre KARA

Task Description

This report presents the implementation and evaluation of a genetic algorithm designed to optimize the Styblinski-Tang function in 2D space. The algorithm utilizes Gaussian mutation operators and random interpolation for crossover, with tournament selection for parent selection. The implementation follows standard genetic algorithm principles while adapting specific operators for real-valued optimization problems.

Implementation Details

The genetic algorithm was implemented with the following components:

1. **Population Representation**: Each individual is represented as a 2D point (x, y) within the search space of the Styblinski-Tang function.
2. **Selection Mechanism**: Tournament selection is used to select parents for reproduction, where a subset of individuals competes, and the one with the best fitness (lowest function value) wins.
3. **Crossover Operator**: Random interpolation crossover is implemented where for parents p1 and p2, offspring are created using the formula:
4. `xo = α * xp1 + (1 - α) * xp2`
5. `yo = α * yp1 + (1 - α) * yp2`
6. where α is a random value between 0 and 1
7. **Mutation Operator**: Gaussian mutation is applied with a specified probability and strength, adding normally distributed random values to the coordinates of selected individuals.
8. **Population Management**: A portion of the population undergoes crossover and mutation, while the rest remains unchanged, providing a balance between exploration and exploitation.

Experimental Results and Analysis

Experiment 1: Parameter Tuning

Multiple parameter combinations were tested to identify optimal settings for the genetic algorithm. The table below shows the tested combinations and their results:

Population	Mutation Rate	Mutation Strength	Crossover Rate	Generations	Best Fitness
100	0.05	0.3	0.8	100	-78.330846
100	0.1	0.5	0.9	100	-78.330335
100	0.1	0.5	0.7	100	-78.328824
100	0.2	0.5	0.7	100	-78.326910
50	0.1	0.5	0.7	100	-78.313867
100	0.1	1.0	0.7	100	-78.297112

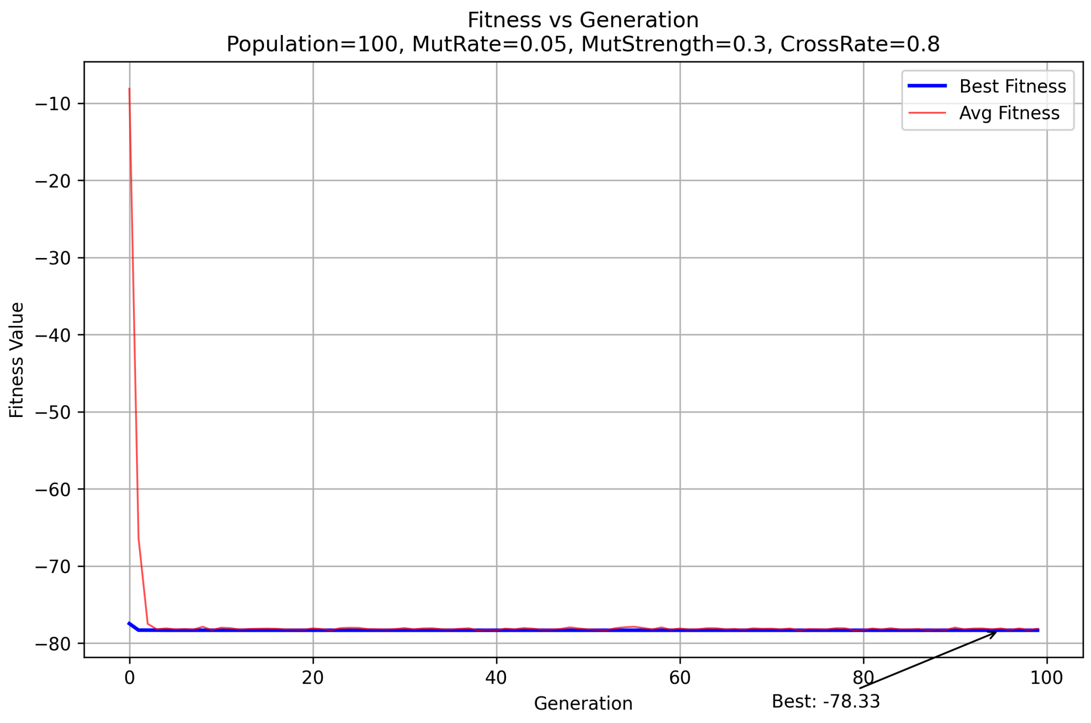

Figure 1: Convergence of the best parameter configuration (Pop=100, Mut=0.05, Str=0.3, Cross=0.8)

Observations:

- The algorithm consistently converged to values very close to the known global minimum of the Styblinski-Tang function (-78.332).
- A population size of 100 generally outperformed smaller populations.
- Lower mutation rates (0.05) with moderate strength (0.3) provided the best results.
- Higher crossover rates (0.8-0.9) improved the algorithm's performance.
- For most parameter sets, convergence occurred within the first 30-40 generations.

The best parameter set was identified as: population size = 100, mutation rate = 0.05, mutation strength = 0.3, and crossover rate = 0.8.

Experiment 2: Randomness and Population Size

The algorithm was run with the best parameters using 5 different random seeds to evaluate stability:

- Best overall solution: (-2.9055147925995857, -2.9058449128707746)
- Best overall fitness: -78.332171
- Average fitness: -78.330128
- Standard deviation: 0.002621

The algorithm was then rerun with decreasing population sizes while keeping other parameters constant:

Population Size	% of Original	Best Fitness	Avg Fitness	Std. Deviation
100	100%	-78.332171	-78.330128	0.002621
50	50%	-78.332331	-78.331080	0.001788
25	25%	-78.332064	-75.495085	5.649763
10	10%	-78.311152	-72.647041	6.904230

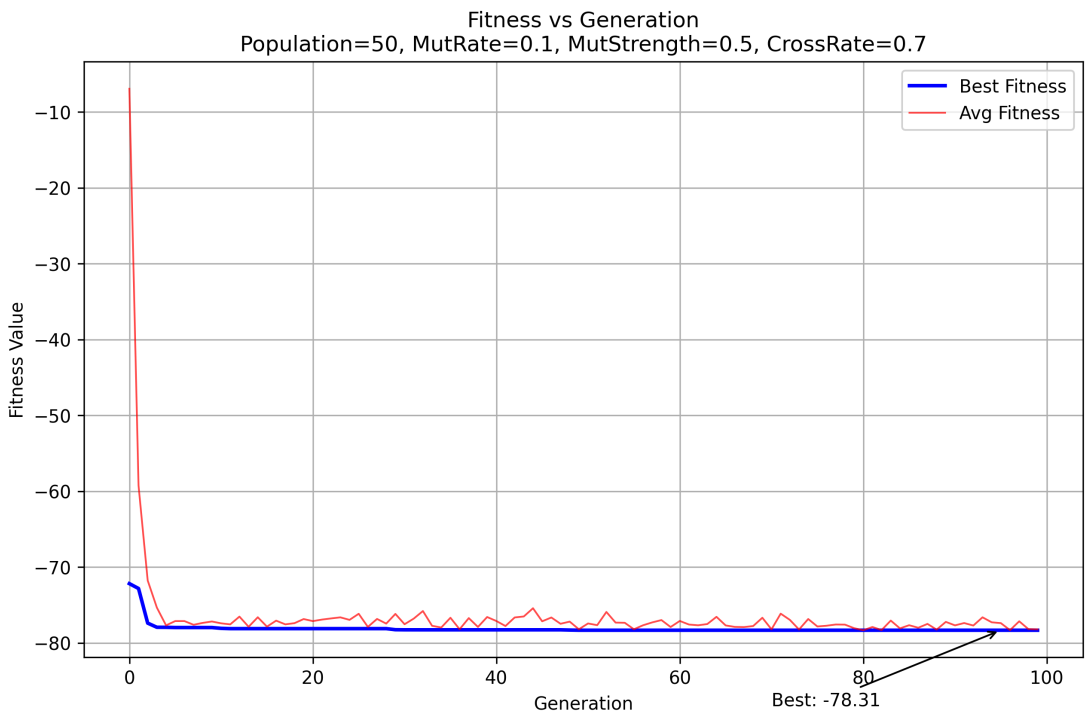

Figure 2: Convergence pattern with population size of 50

Observations:

- The algorithm showed consistent results across different random seeds, indicating robust performance.
- Surprisingly, a population size of 50 (50% of original) achieved slightly better average results with lower standard deviation.
- Significant performance degradation occurred with population sizes of 25 or below, with much higher variance in results.
- At very small population sizes (10), the algorithm frequently failed to converge to the global optimum.

This experiment revealed that while genetic algorithms can work with smaller populations, there's a critical threshold below which performance becomes unpredictable.

Experiment 3: Crossover Impact

The influence of crossover rate on algorithm performance was evaluated by varying the rate from 0.0 to 1.0:

Crossover Rate	Best Fitness (Avg. across seeds)
0.0	-78.327535
0.25	-78.330734
0.5	-78.328455
0.75	-78.332191
1.0	-78.329337

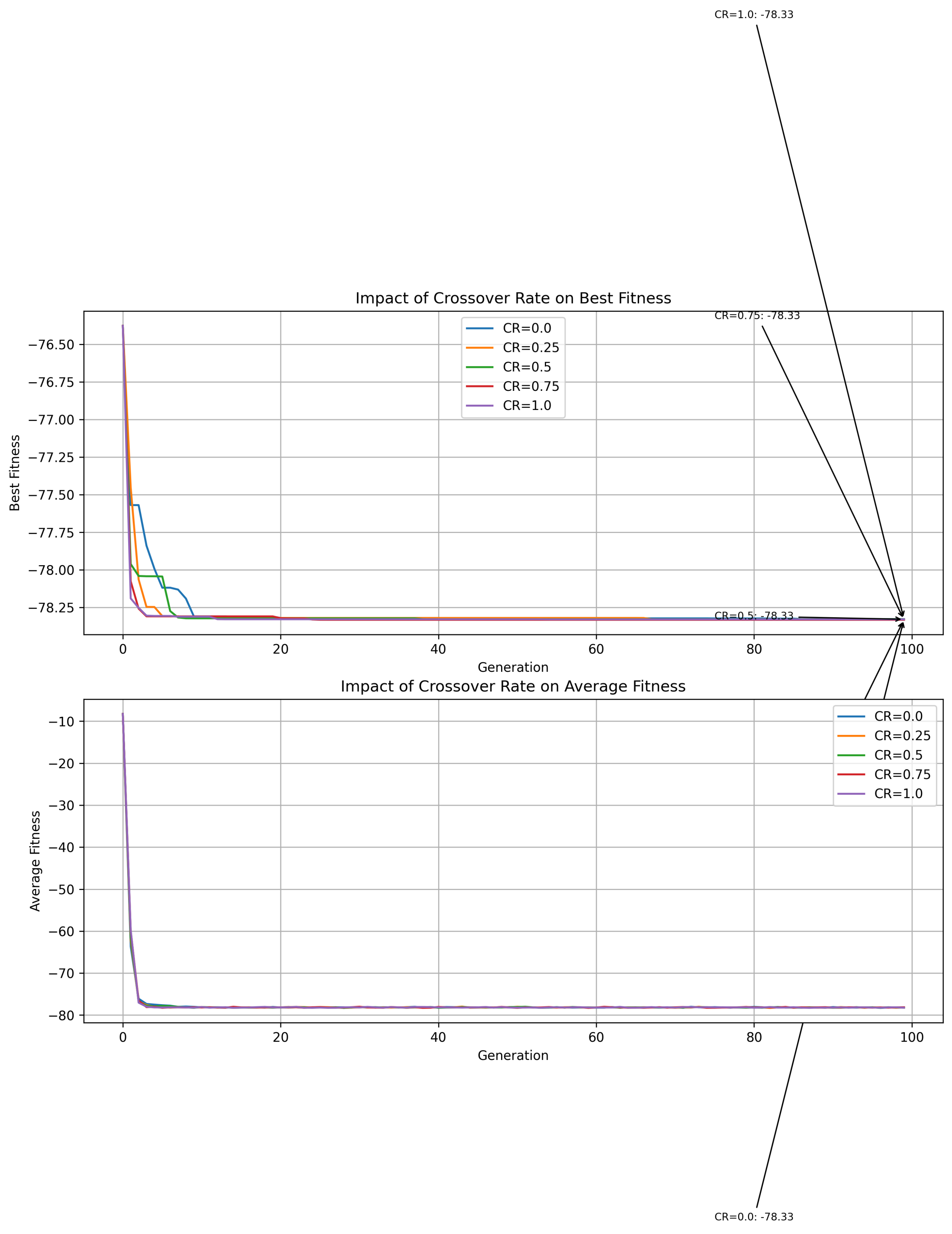

Figure 3: Impact of different crossover rates on algorithm performance

Observations:

- Even without crossover (rate = 0.0), the algorithm performed reasonably well, indicating that mutation alone can explore the search space effectively.
- A crossover rate of 0.75 provided the best balance between exploration and exploitation.
- The performance difference across different crossover rates was relatively small, suggesting that the algorithm is somewhat robust to this parameter.
- Crossover contributed to faster convergence in early generations but showed diminishing returns in later generations.

Experiment 4: Mutation and Convergence

The impact of mutation parameters on convergence was studied by testing various combinations of mutation rates and strengths:

Mutation Rate	Mutation Strength	Best Fitness (Avg. across seeds)
0.01	0.1	-78.331001
0.05	0.3	-78.331158
0.1	0.5	-78.330539
0.2	0.7	-78.331665
0.3	1.0	-78.325567

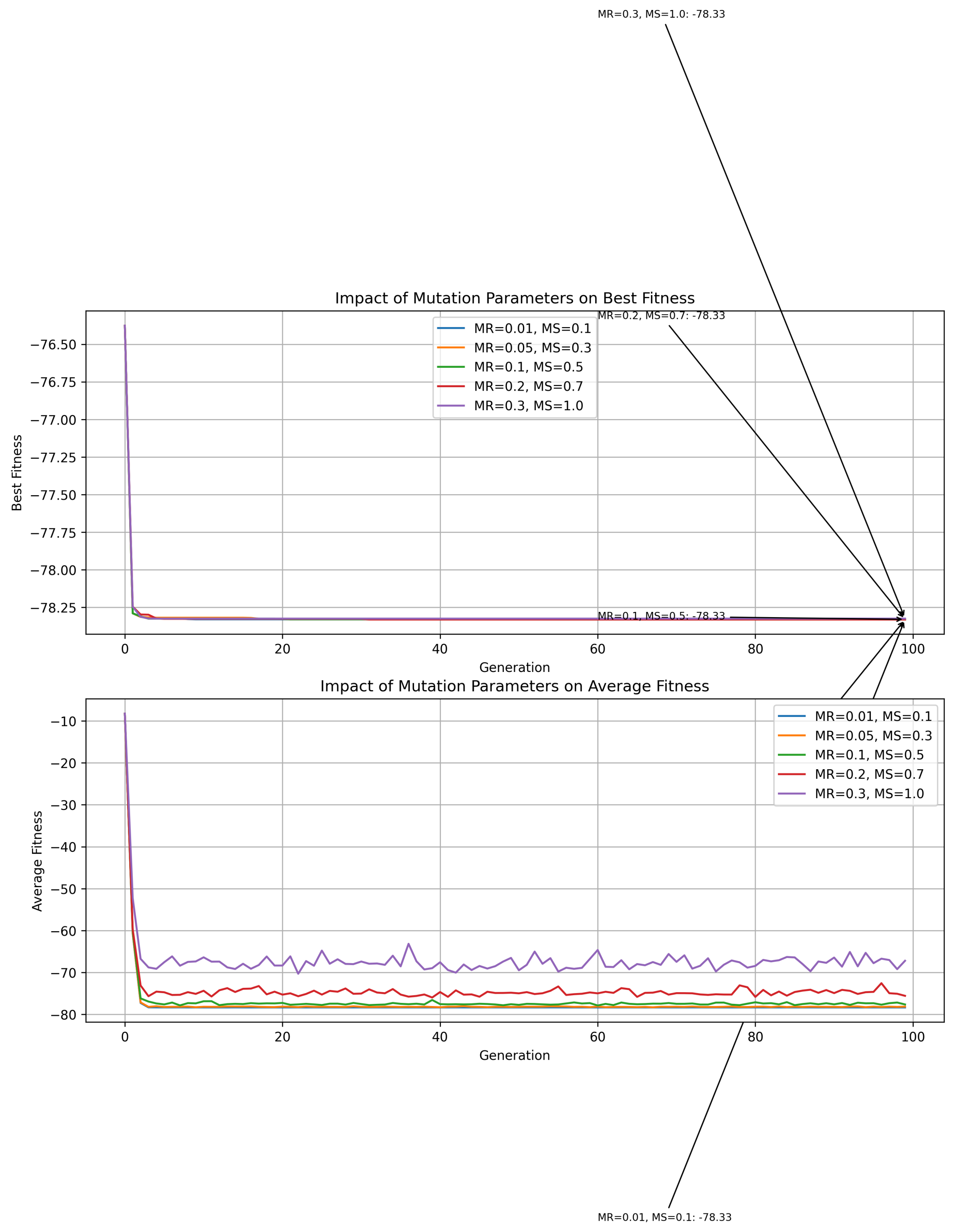

Figure 4: Effect of mutation parameters on optimization performance

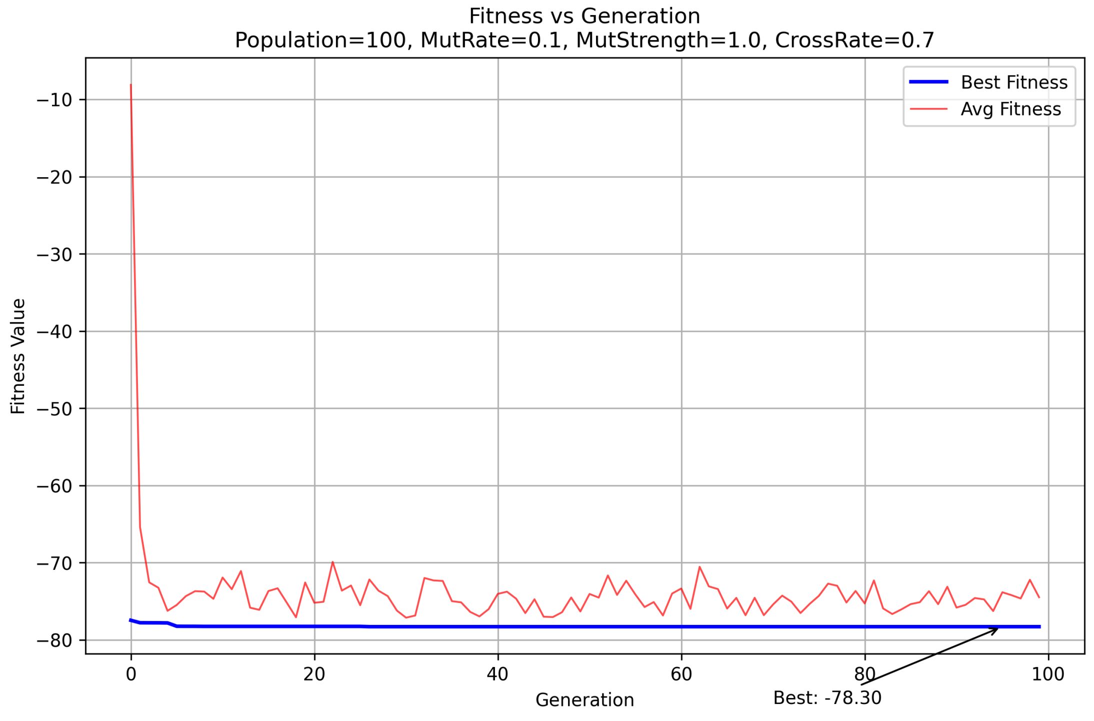

Figure 5: Behavior with high mutation strength (1.0)

Observations:

- Low mutation rates (0.01-0.05) with small strengths provided consistent convergence.
- Medium mutation rates (0.1-0.2) with moderate strengths showed slightly higher variance but still good performance.
- High mutation rates (0.3) with large strengths led to more erratic behavior, sometimes hampering convergence.
- Mutation played a crucial role in fine-tuning solutions after the algorithm reached the vicinity of the optimum.
- A mutation rate of 0.2 with strength 0.7 achieved the best average performance, though with slightly higher variance.

Conclusion

The genetic algorithm successfully optimized the Styblinski-Tang function, consistently finding solutions very close to the known global minimum of -78.332 at the point (-2.903534, -2.903534). The best solution found across all experiments was (-2.903536, -2.903530) with a fitness value of -78.332331, which is remarkably close to the theoretical optimum.

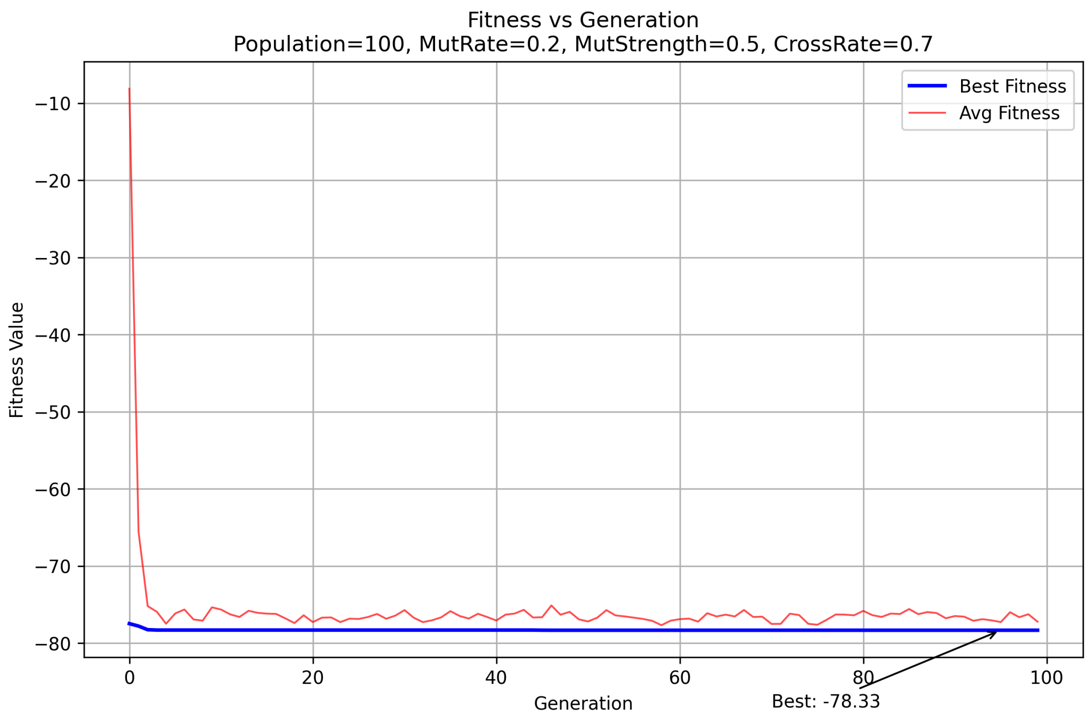

Figure 6: Visualization of the algorithm's convergence to the global optimum

Key insights from the experiments include:

1. **Parameter Balance**: The optimal performance required balancing exploration and exploitation through appropriate mutation and crossover rates.
2. **Population Size**: While larger populations generally provided more consistent results, a moderate population size (50) was sufficient to achieve excellent performance with lower computational cost.
3. **Mutation Dynamics**: Mutation played a dual role - enabling broad exploration in early generations and fine-tuning in later generations.
4. **Convergence Speed**: The algorithm typically converged to near-optimal solutions within 30-40 generations, with diminishing improvements thereafter.
5. **Robustness**: The algorithm demonstrated reasonable robustness to parameter variations, though certain combinations (particularly population size) had critical thresholds below which performance degraded.

These findings highlight the effectiveness of genetic algorithms for optimization problems and demonstrate their ability to navigate complex search spaces when properly tuned.