PRACTICAL ASSIGNMENT

Shuzhou Zhao
3997545
s3997545@umail.leidenuniv.nl

Haowen Ran
4473590
s4473590@umail.leidenuniv.nl

December 24, 2024

1 Introduction

The study is split into two parts: the first applies standard genetic algorithm to the LABS and N-Queens problems ; the second implements evolution strategy to figure out the Katsuura problem from the Black-Box Optimization Benchmark (BBOB) suite. Key aspects such as operator design, parameter tuning, and performance evaluation metrics are discussed.

For hyper-parameter tuning, the steps include: Defining the hyper-parameter search space, multiple independent running, and select the most effective hyper-parameter combination after computing the average performance for each combination.

2 Algorithms

2.1 Genetic Algorithm Operators

2.1.1 operators

The GA relies on three main operators:

1. **Selection**: Roulette Wheel Selection is used, where individuals are selected with probabilities proportional to their fitness values. If all fit values are zero or negative, a uniform probability distribution is applied to ensure diversity.
2. **Crossover**: Single-point crossover is employed with a specified crossover rate r_c. Two parents produce two offspring by exchanging segments of their binary denotes at a randomly chosen crossover point.
3. **Mutation**: Bit-flip mutation is applied to the offspring with a mutation rate r_m. A random bit in the binary denote is flipped (from 0 to 1 or from 1 to 0).

2.1.2 parameter settings

The parameters used in the GA are as follows:

- **Population size** (P): 50 individuals.
- **Crossover rate** (r_c): 0.5.
- **Mutation rate** (r_m): 0.9.
- **Budget**: A maximum of 5000 fitness evaluations.
- **Elitism**: Ensures the best individuals are retained in subsequent generations.

2.2 Evolution Strategy

2.2.1 operators

1. **Initialization** Each individual in the population is randomly initialized. The variables of each individual are in $[-5, 5]$.
2. **Selection** Implement the random selection algorithm to randomly select individuals with a number of 'lambda_' from the parent population, allowing for repetitions. This is to provide candidate parents for recombination.
3. **Recombination** Implement the arithmetic crossover algorithm, each offspring is the weighted average of two random parents. This step is to generate better offspring individuals through gene recombination.
4. **Mutation** Implement Gaussian noise to perturb each child variable. The noise standard deviation is controlled by the parameter 'sigma'. This step is to increase population diversity and avoid local optimality.
5. **Survivor selection** Implement $\mu + \lambda$ strategy, merge the parent and offspring individuals, sort them according to fitness value, and select the best μ individuals to enter the next generation. Maintain the best individuals in the population and optimize them step by step.
6. **Boundary handling** Ensure that each variable is within the range of $[-5, 5]$ to prevent the variable from exceeding the domain.

2.2.2 parameter setting

- **Budget** 50000. The maximum number of function evaluations.
- **Dimension** 10.
- **Population size (mu)** 50. Individuals in the parent generation.
- **Offspring size (lambda_)** 50 Offspring produced per generation.
- **Mutation strength (sigma)** 0.1. The standard deviation of the variation.

3 Hyper-parameter Tuning

In this study, we explore a systematic approach to identify the optimal set of hyper-parameters for our GA implementation. The key hyper-parameters considered are:

- **Population size:** The number of individuals in each generation of the GA.
- **Mutation rate:** The probability of mutation occurring in offspring during reproduction.
- **Crossover rate:** The probability of crossover operations being performed between two parents to produce offspring.

The hyper-parameter tuning process involves evaluating combinations of these parameters across two benchmark problems: *LABS* and *N-Queens Problem*. Each problem is tested at specific dimensions, and the evaluation process follows these steps:

1. Define a hyper-parameter search space with candidate values for each parameter.
2. For each combination of parameters, perform five independent runs of the GA on each problem.
3. Record the performance score achieved in each run.
4. Compute the average performance across all runs for each parameter combination.
5. Select the hyper-parameter combination that maximizes the average score across problems and runs.

4 Experimental Results

4.1 part 1: LABS Problem and N-Queens Problem

The best-found fitness plot (Figure 1) indicates that while the genetic algorithm steadily improves the fitness over function evaluations, convergence occurs slower compared to the N-Queens problem. This slower convergence suggests that LABS has a more rugged search landscape or tighter constraints, making it harder for the algorithm to find optimal solutions.

The ECDF for LABS (Figure 3) shows that a smaller proportion of runs achieve the target performance within a given number of evaluations. This indicates that LABS not only presents difficulties in finding solutions but also in consistently solving the problem across multiple runs. Furthermore, the ERT curve for LABS (Figure 2) is higher, suggesting that the algorithm requires more evaluations to reach the same performance level, further emphasizing the problem's complexity.

The N-Queens problem, on the other hand, demonstrates a more favorable performance profile. The best-found fitness plot (Figure 4) shows a consistent improvement with fewer evaluations needed to achieve convergence. This suggests that the solution space of the N-Queens problem is less rugged or that the algorithm is better suited to solving this problem.

The ECDF for N-Queens (Figure 6) rises more quickly compared to LABS, indicating higher success rates across runs and with fewer function evaluations. Additionally, the ERT values for N-Queens (Figure 5) are lower, highlighting the efficiency of the genetic algorithm in navigating its solution space. Overall, the algorithm performs significantly better on the N-Queens problem compared to LABS.

Furthermore, the AUC (Area Under the Curve) values for these ECDF plots highlight the differences in performance. The N-Queens Problem problem achieves an **AUC value** of **0.873**, while the LABS problem has a lower **AUC value** of **0.731**. This quantitative measure reinforces the observation that the N-Queens Problem problem demonstrates higher overall performance in terms of success rate and efficiency.

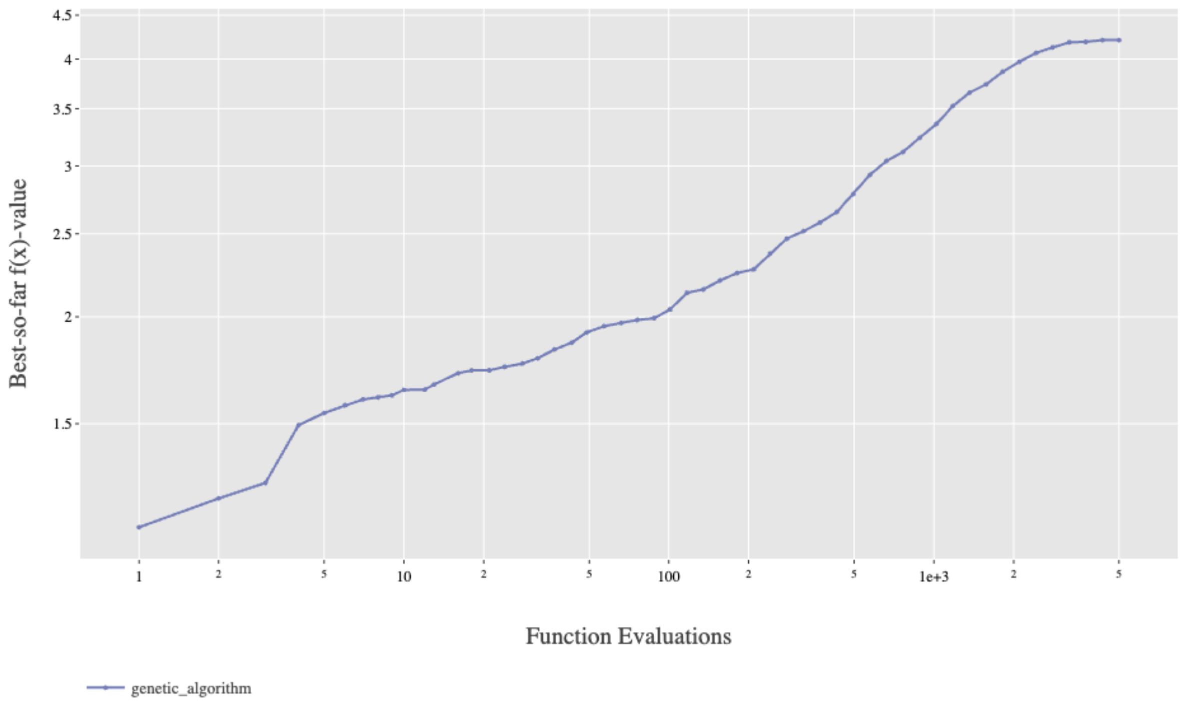

Figure 1: Average best-found fitness for LABS Problem.

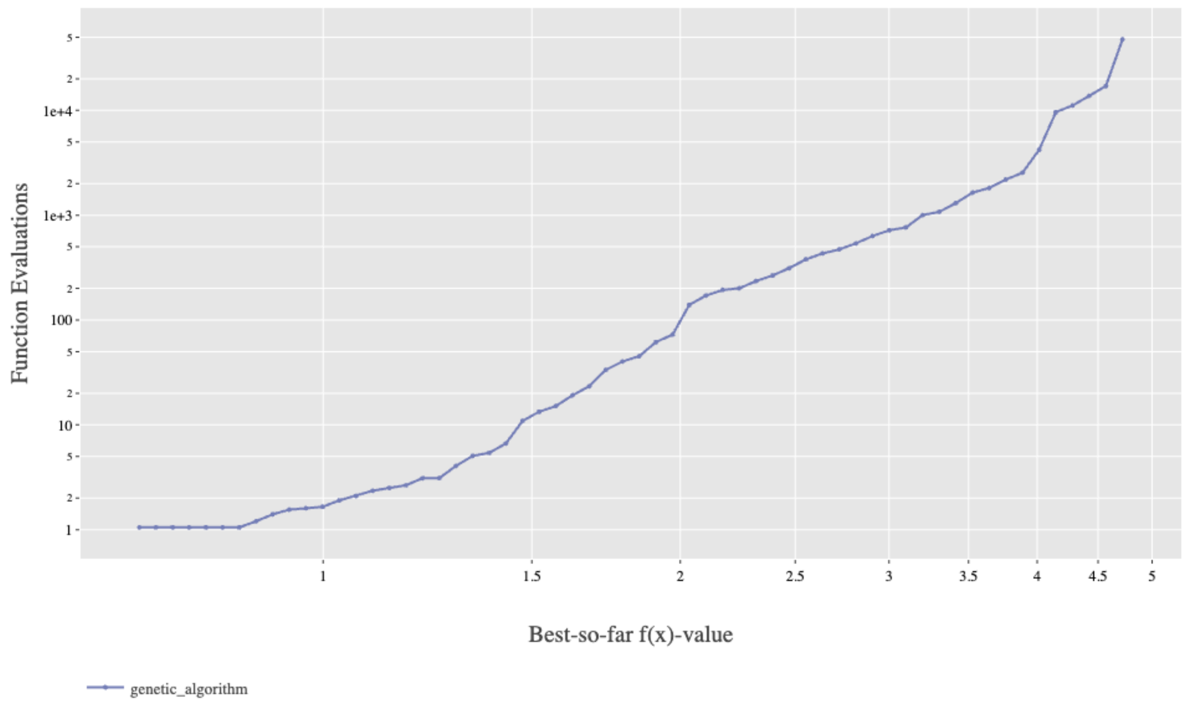

Figure 2: ERT (Expected Running Time) for LABS.

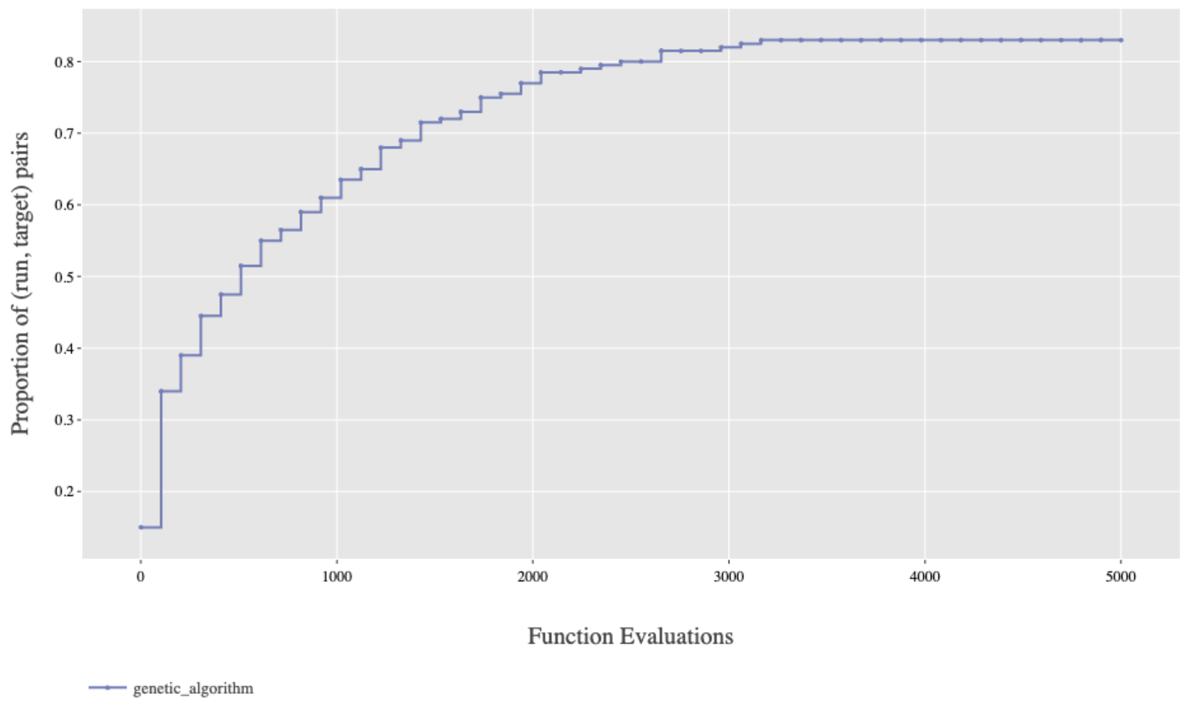

Figure 3: ECDF (Empirical Cumulative Distribution Function) for LABS.

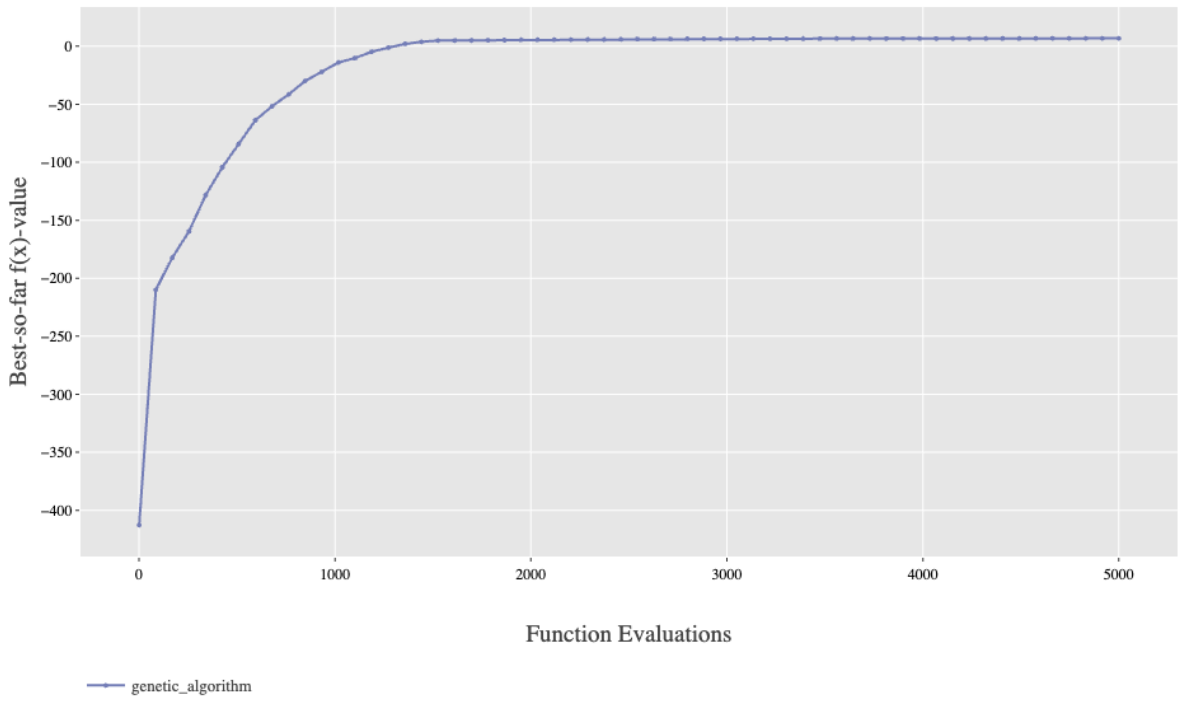

Figure 4: Average best-found fitness for N-Queens Problem.

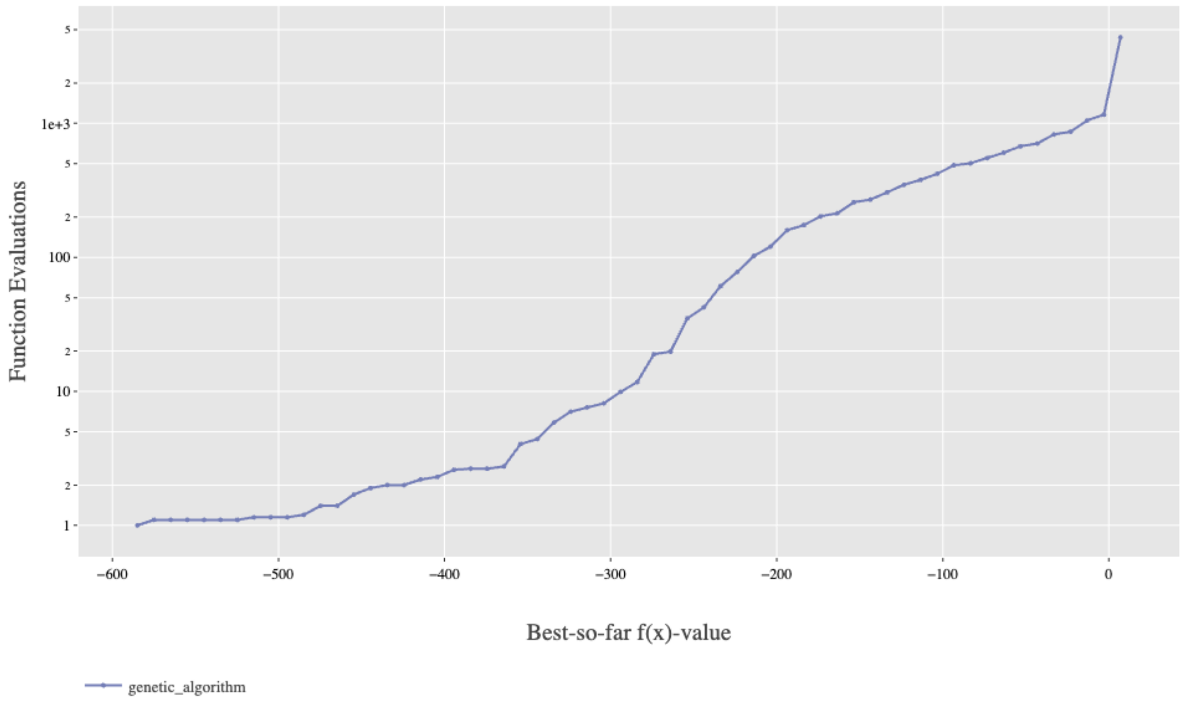

Figure 5: ERT (Expected Running Time) for N-Queens Problem.

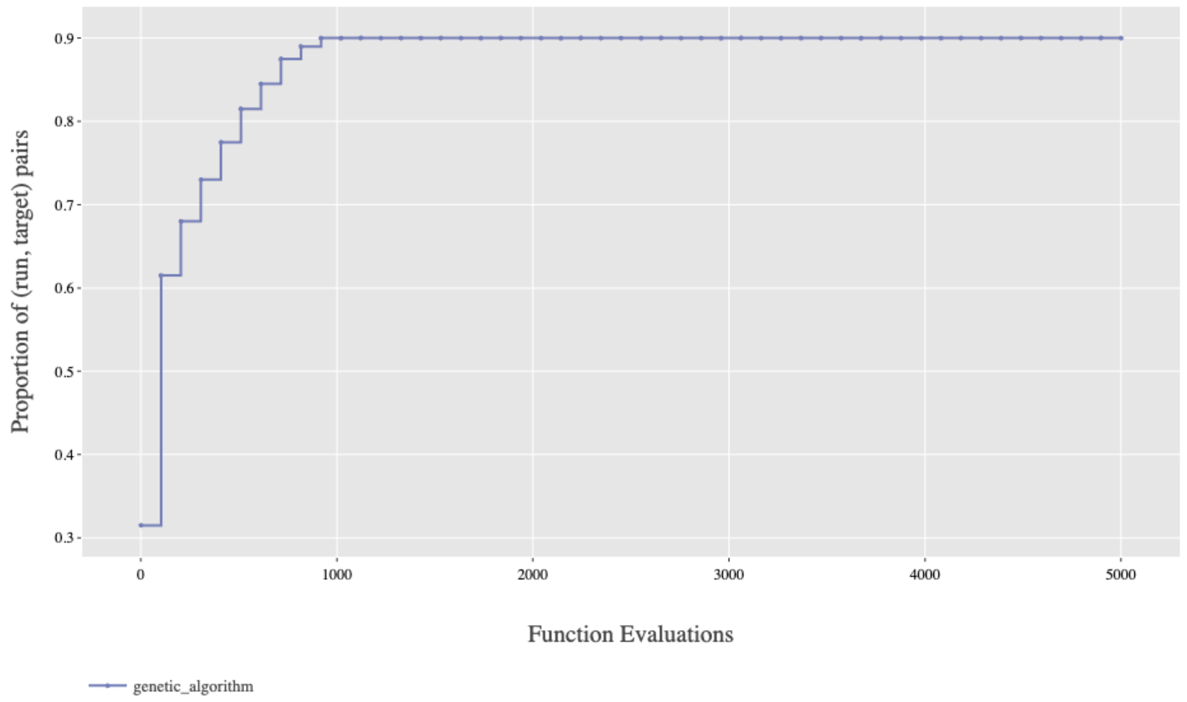

Figure 6: ECDF (Empirical Cumulative Distribution Function) for N-Queens Problem.

4.2 Hyper-parameter Tuning

Table 1: Evaluated Hyper-parameters and Scores for LABS and N-Queens Problem

Population Size	Mutation Rate	Crossover Rate	Avg. Score
50	0.3	0.3	5.1365
50	0.3	0.5	4.9350
50	0.3	0.9	4.9948
50	0.9	0.3	5.2722
50	0.9	0.5	**5.6905**
50	0.9	0.9	5.3036
100	0.3	0.3	5.0547
100	0.3	0.5	5.2000
100	0.3	0.9	4.7712
100	0.9	0.3	5.3324
100	0.9	0.5	5.4194
100	0.9	0.9	5.3984
200	0.3	0.3	3.4557
200	0.3	0.5	4.5706
200	0.3	0.9	4.5127
200	0.9	0.3	4.1016
200	0.9	0.5	4.3046
200	0.9	0.9	4.1924

The **best-performing hyperparameters** for both problems were:

- **Population Size:** 50
- **Mutation Rate:** 0.9
- **Crossover Rate:** 0.5

Certain hyper-parameter combinations, such as a population size of 200 paired with high mutation and crossover rates, resulted in lower average scores. This outcome is likely due to inefficiencies in managing a large search space, which may have caused slower convergence or over-diversification, hindering the algorithm's ability to refine promising solutions.

Nevertheless, the genetic algorithm's inherent mechanisms—selection, mutation, and crossover—impart a degree of robustness, enabling relatively stable performance even when hyper-parameters are not fully optimized. The dimensions of the problems (50 for LABS and 49 for N-Queens) are challenging but manageable, allowing most configurations to explore the solution space effectively without excessive resource demands. Additionally, the tested hyper-parameter ranges, such as crossover rates of 0.3, 0.7, and 0.9, were within reasonable bounds, avoiding extreme values that might otherwise cause significant performance divergence.

The observed consistency across hyper-parameter combinations underscores the robustness of the genetic algorithm and its capacity to deliver reliable performance across a range of parameter settings.

4.3 part 2: Katsuura Problem

The best-found fitness plot (Figure 7) shows the improvement of the fitness value over the number of function evaluations. The fitness improves steadily as the number of evaluations increases, with the most significant improvements occurring during the early stages of the optimization process. As the algorithm progresses, the rate of improvement decreases, indicating convergence toward an optimal solution. The logarithmic scaling of the evaluation axis highlights the high computational cost associated with achieving further fitness improvements, particularly in the later stages.

The ECDF plot (Figure 9) illustrates the proportion of runs that achieve the target performance within a given number of function evaluations. The ECDF curve starts at approximately 50%, indicating that half of the runs reach the initial target early in the optimization process. The curve rises steadily, flattening at around 90%, meaning most runs successfully reach the target performance. The area under the curve (**AUC**) for the ECDF is calculated as **0.908**, reflecting the high success rate and efficiency of the evolution strategy across runs. However, the gradual flattening of the curve suggests that achieving the target performance requires significant computational effort as the problem becomes increasingly challenging.

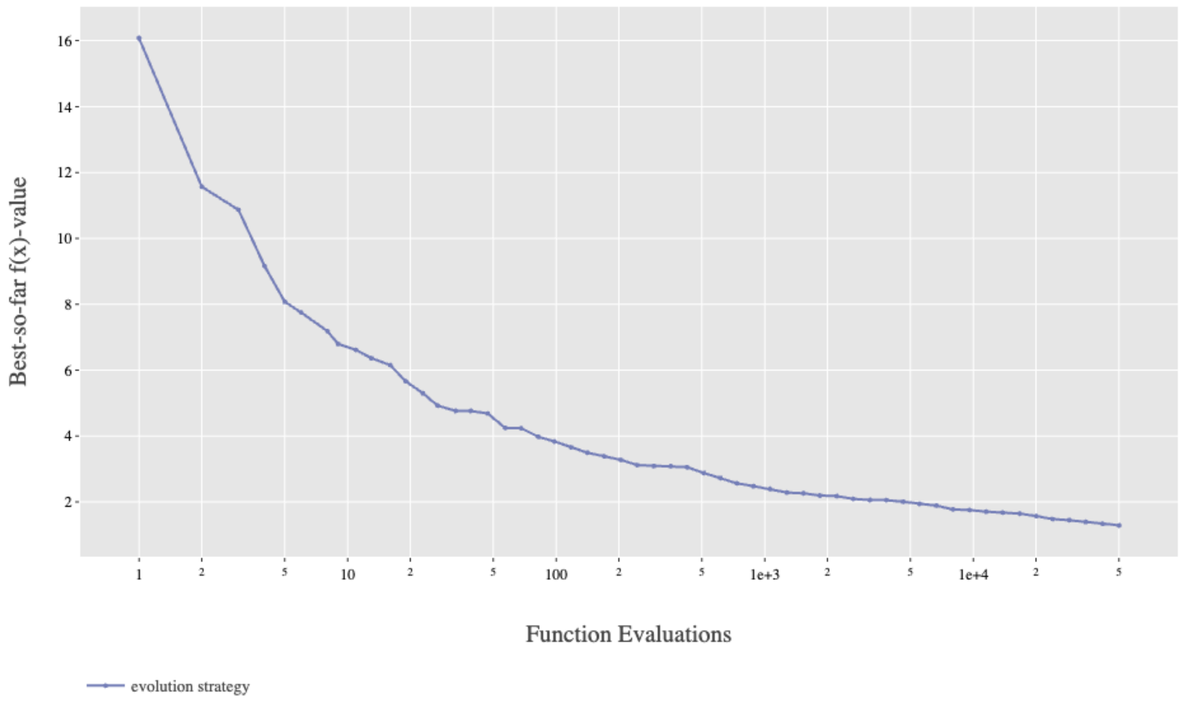

Figure 7: Average best-found fitness for Katsuura Problem.

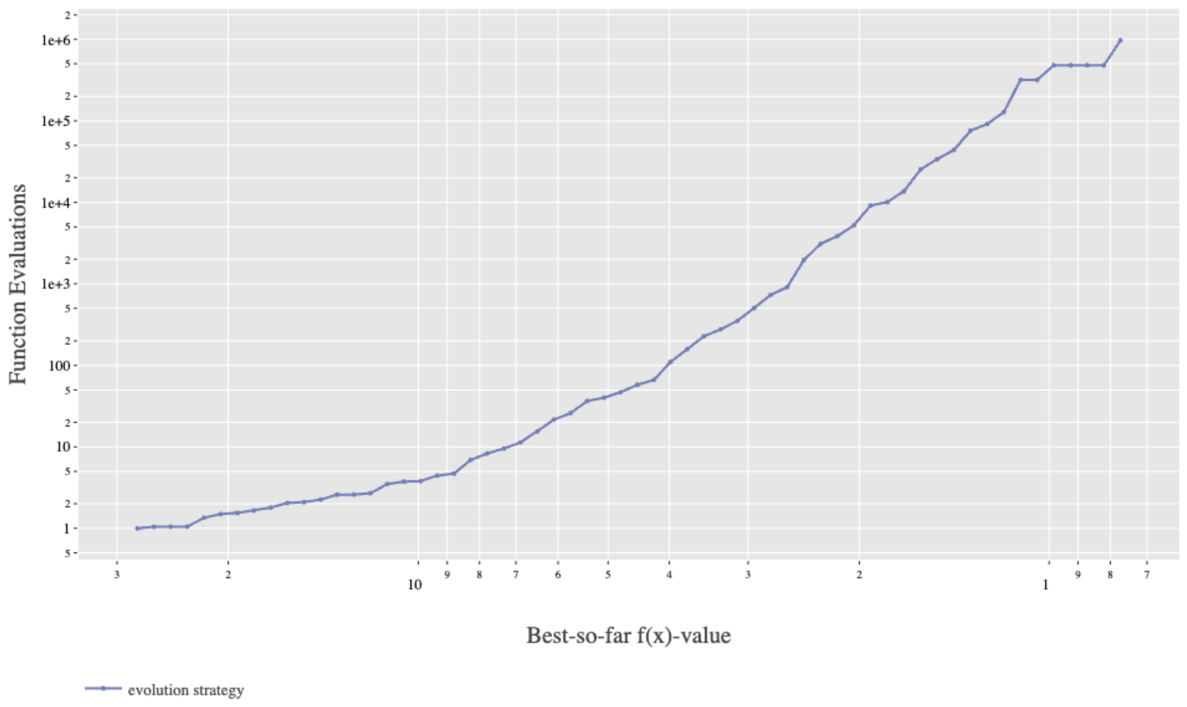

Figure 8: ERT (Expected Running Time) for Katsuura Problem.

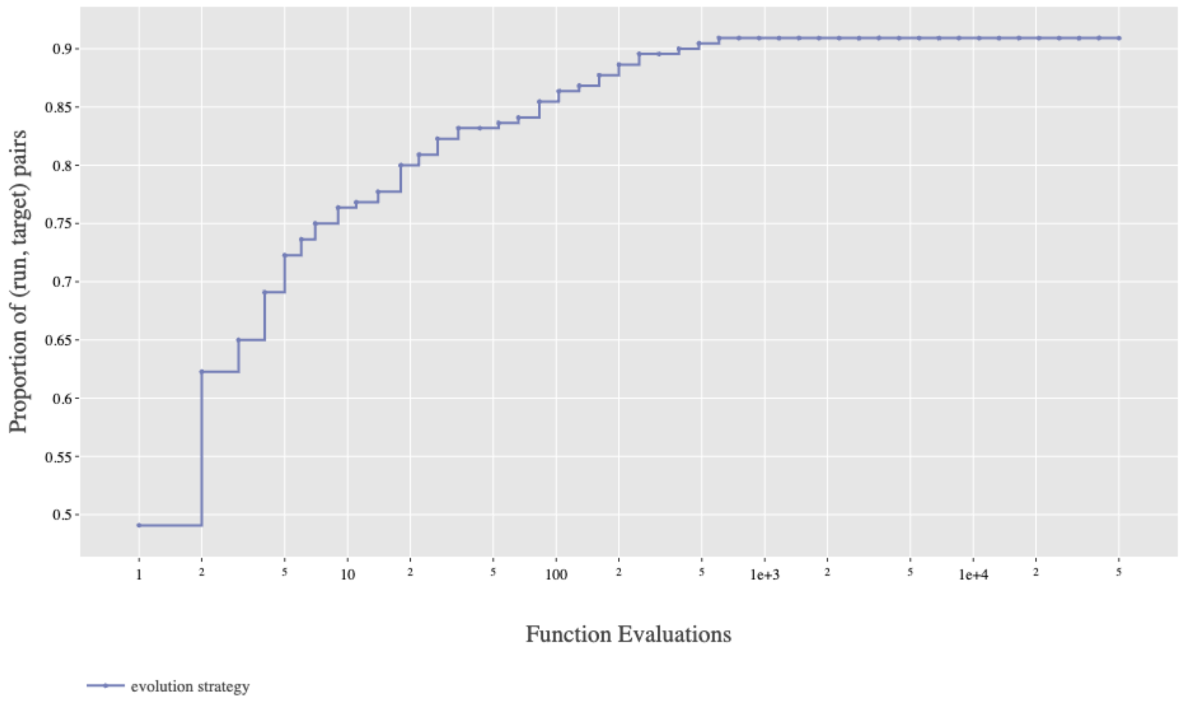

Figure 9: ECDF (Empirical Cumulative Distribution Function) for Katsuura Problem.

5 Discussion and Conclusion

Genetic Algorithms (GAs) demonstrated strong performance on the N-Queens problem, leveraging its smooth solution landscape, but encountered difficulties with the rugged landscape of LABS. Evolution Strategies (ES) excelled on the Katsuura problem, highlighting their robust convergence and adaptability in high-dimensional optimization tasks. Both algorithms exhibited robustness under varying parameter settings, achieving reasonable performance even without extensive tuning, underscoring their versatility as general-purpose optimization tools. Notably, GAs displayed consistent performance across different configurations during parameter tuning, further reaffirming their applicability across diverse problem domains.