Bărnuț Cristiana Florina Diana
Group: 921

Benchmark Optimization Functions Using Genetic Algorithms

1. Introduction

Genetic Algorithms (GAs) are optimization methods inspired by natural selection. They evolve a population of solutions through selection, crossover, and mutation to find optimal values for complex problems. This project tests different GA configurations—binary vs. real encoding, crossover types, and mutation rates—on benchmark functions. We analyze results using plots and statistical tests to compare performance and draw conclusions about the best setups.

2. Functions Description

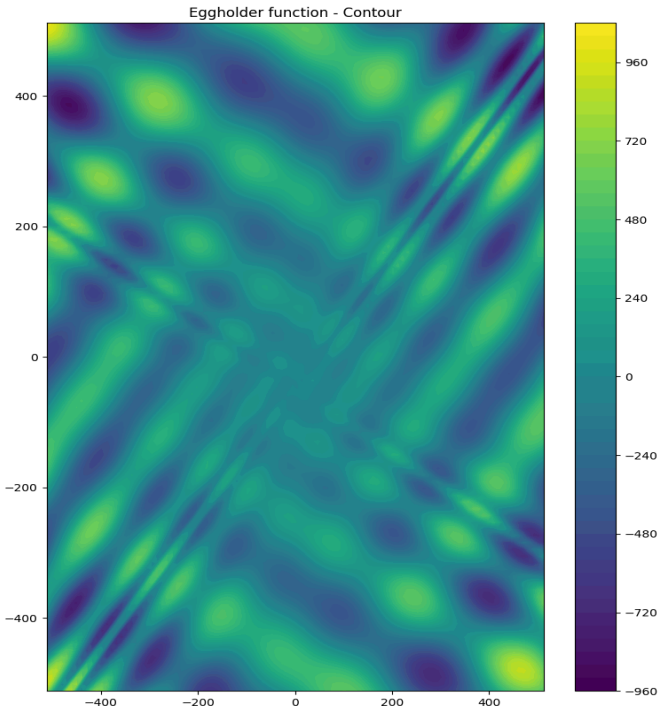
2.1 Eggholder Function

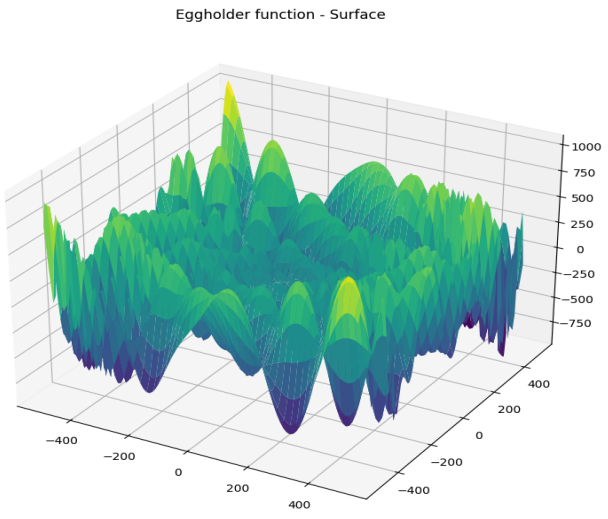
The Eggholder function is a highly multimodal function with many local minima, making it difficult for optimization algorithms to find the global minimum. It is defined as:

$$f(X) = -(y + 47)sin(\sqrt{|y + \frac{x}{2} + 47|}) - x\,sin(\sqrt{|x - (y + 47)|})$$

where x and y are real-valued inputs typically constrained in the range[−512,512].

The function has a very irregular surface with sharp ridges and deep valleys. The global minimum is approximately −959.6407 at around (512,404.2319). Due to its complexity, the Eggholder function provides a rigorous test of an algorithm's ability to escape local optima and explore the search space efficiently.





Bărnuț Cristiana Florina Diana
Group: 921

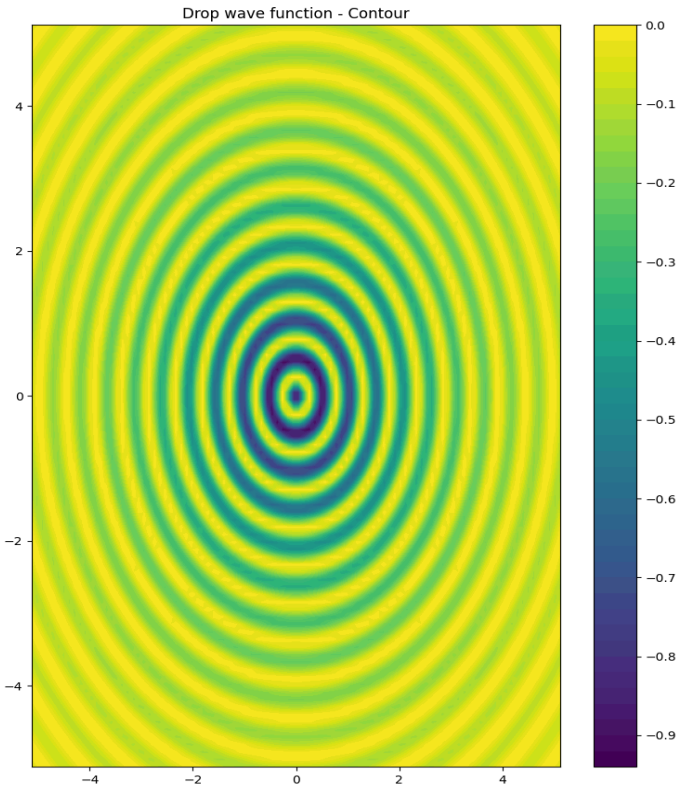
2.2 Drop-Wave Function

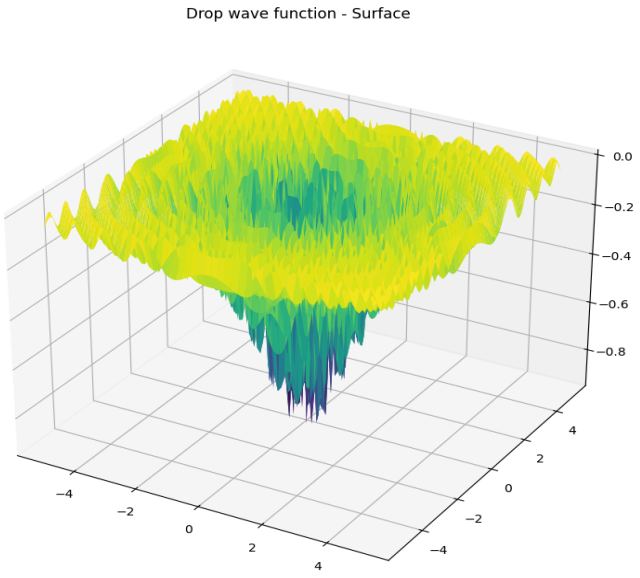
The Drop-Wave function is another challenging multimodal test function characterized by a series of ripples superimposed on a bowl-shaped surface. It is defined as:

$$f(x,y) = -\frac{1+cos(12\sqrt{x^2+y^2})}{0.5(x^2+y^2)+2}$$

with the typical input domain of [−5.12, 5.12] for both x and y.

This function has a global minimum of −1 at the origin (0,0) but includes many local minima caused by the cosine term. The Drop-Wave function tests the GA's precision and robustness in locating the global minimum amidst many deceptive local traps.





3. Generic Algorithms Configuration

3.1 Representation

Two types of solution representations were considered:

- Real-valued representation: Individuals are represented as vectors of real numbers within defined bounds for each variable.

- Binary representation: Individuals are represented as binary strings, where each variable is encoded as a fixed-length binary substring. The binary strings are decoded into real values before fitness evaluation.

3.2 Population Initialization

- The population consists of 50 individuals (default).

- For real-valued representation, each individual's variables are initialized randomly within their respective bounds using uniform sampling.

- For binary representation, each individual is initialized as a random sequence of bits of length equal to the number of variables multiplied by the number of bits per variable (default 16 bits per variable).

3.3 Selection Method

- Tournament selection with tournament size k=3 is used to select parents for reproduction.

- This method randomly samples k individuals and selects the one with the best fitness among them, promoting selection pressure while maintaining diversity.

3.4 Crossover Operators

- For binary representation:

    - 1-point crossover: A random crossover point is selected, and segments are swapped between two parents to generate offspring.

    - 2-point crossover: Two crossover points are selected, and the segment between these points is exchanged between parents.

- For real-valued representation:

    - Arithmetic crossover: Offspring are generated by averaging corresponding variables of the parents.

    - BLX-α (Blend crossover): Offspring variables are sampled uniformly from an extended interval around parent values, controlled by parameter α=0.5, promoting exploration.

3.5 Mutation Operators

- For binary representation:
    - Each bit has a probability of mutation defined by the mutation rate (default 0.01), where it flips from 0 to 1 or vice versa.

- For real-valued representation:
    - Each variable may be mutated by adding Gaussian noise scaled to 10% of the variable's domain width, with probability equal to the mutation rate.

3.6 Other Parameters

- Population size: 50 individuals.
- Number of generations: 100 iterations.
- Crossover rate: 0.8, the probability that crossover will be applied to selected parents.
- Mutation rate: 0.01, the probability of mutating each gene or variable.
- Number of bits per variable (binary): 16.

3.7 Algorithm Flow

The GA proceeds as follows:

1. Initialize a population of individuals.
2. Decode individuals if using binary representation.
3. Evaluate fitness for all individuals.
4. Select parents using tournament selection.
5. Apply crossover to produce offspring with crossover rate.
6. Apply mutation to offspring.
7. Form the next generation from offspring.
8. Track and update the best solution found.
9. Repeat for the specified number of generations.

4. Experimental results

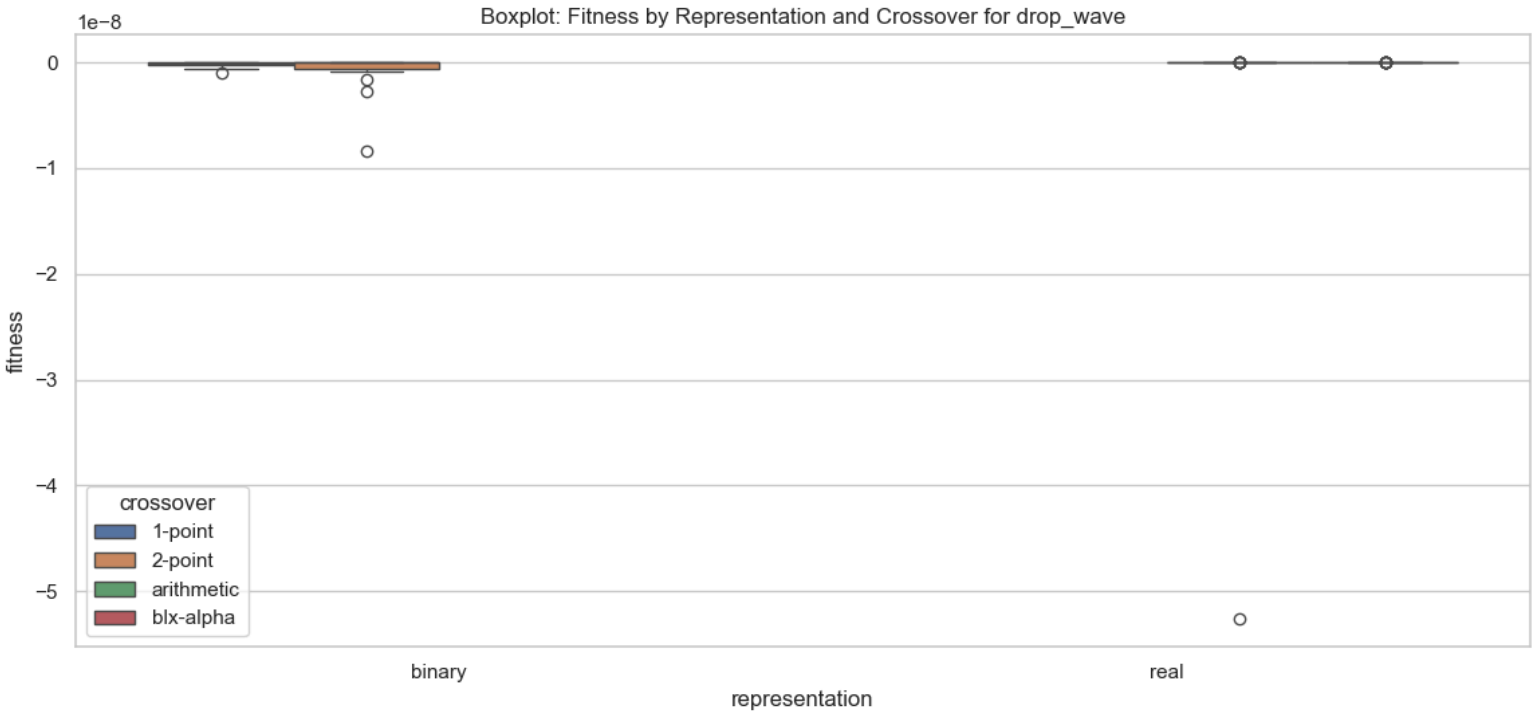
Bărnuț Cristiana Florina Diana
Group: 921

The fitness values obtained from multiple independent runs for each GA configuration on both benchmark functions are summarized below.

1. Drop-Wave Function:

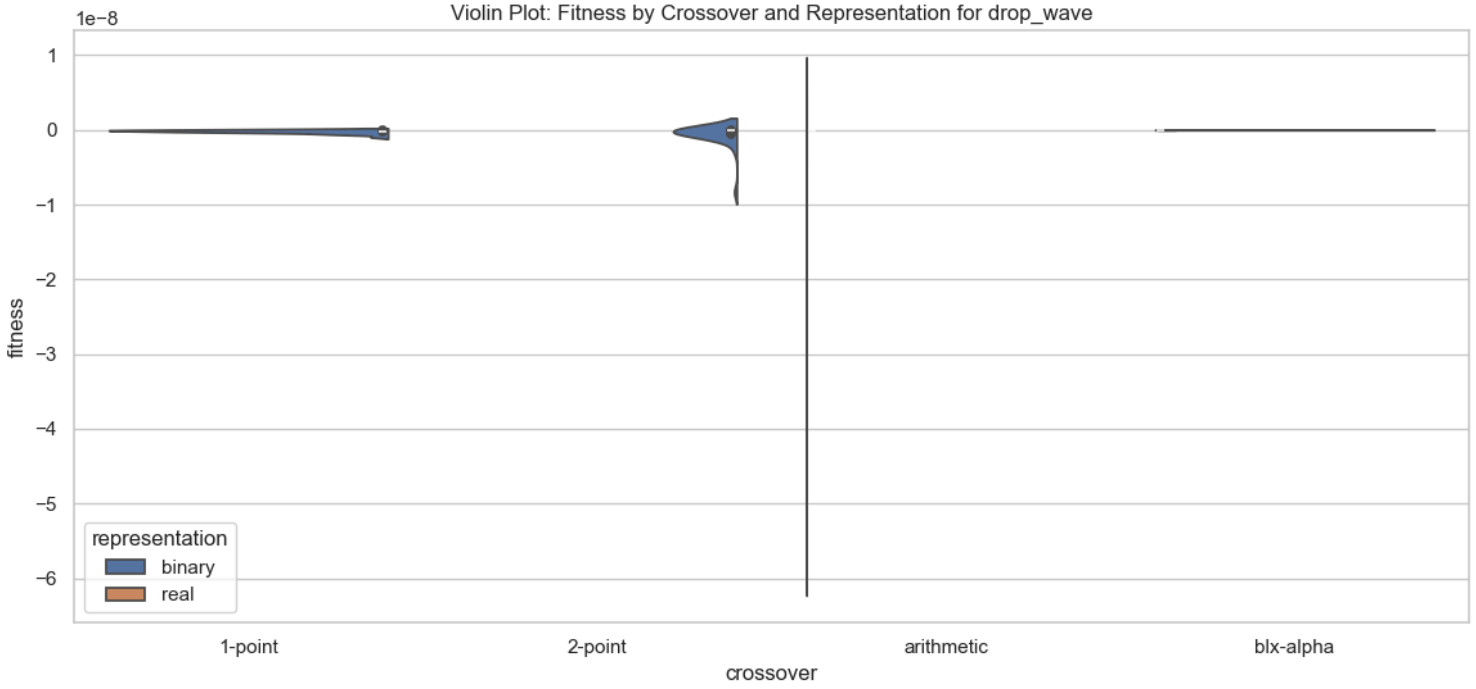
- Using **binary representation** with *1-point* and *2-point* crossover, the fitness values were generally very close to zero but slightly negative (indicating minimization close to the global optimum). The minimum fitness values ranged between approximately −10^{-10} and −10^{-9}.
- Using **real-valued representation** with *arithmetic* and *BLX-alpha* crossover, the GA consistently reached fitness values near zero, often reported as −0.0, indicating excellent convergence close to the global minimum.

Representation	Crossover	Standard deviation	Best Fitness	Mean Fitness
binary	1-point	2.32e-10	-9.57e-10	-2.10e-10
binary	2-point	1.57e-0.9	-8.33e-09	-6.30e-10
real	arithmetic	9.60e-09	-5.25e-08	-1.75e-09
real	blx-alpha	6.27e-13	-2.48e-12	-2.14e-13



05/06/2025

Bărnuț Cristiana Florina Diana
Group: 921



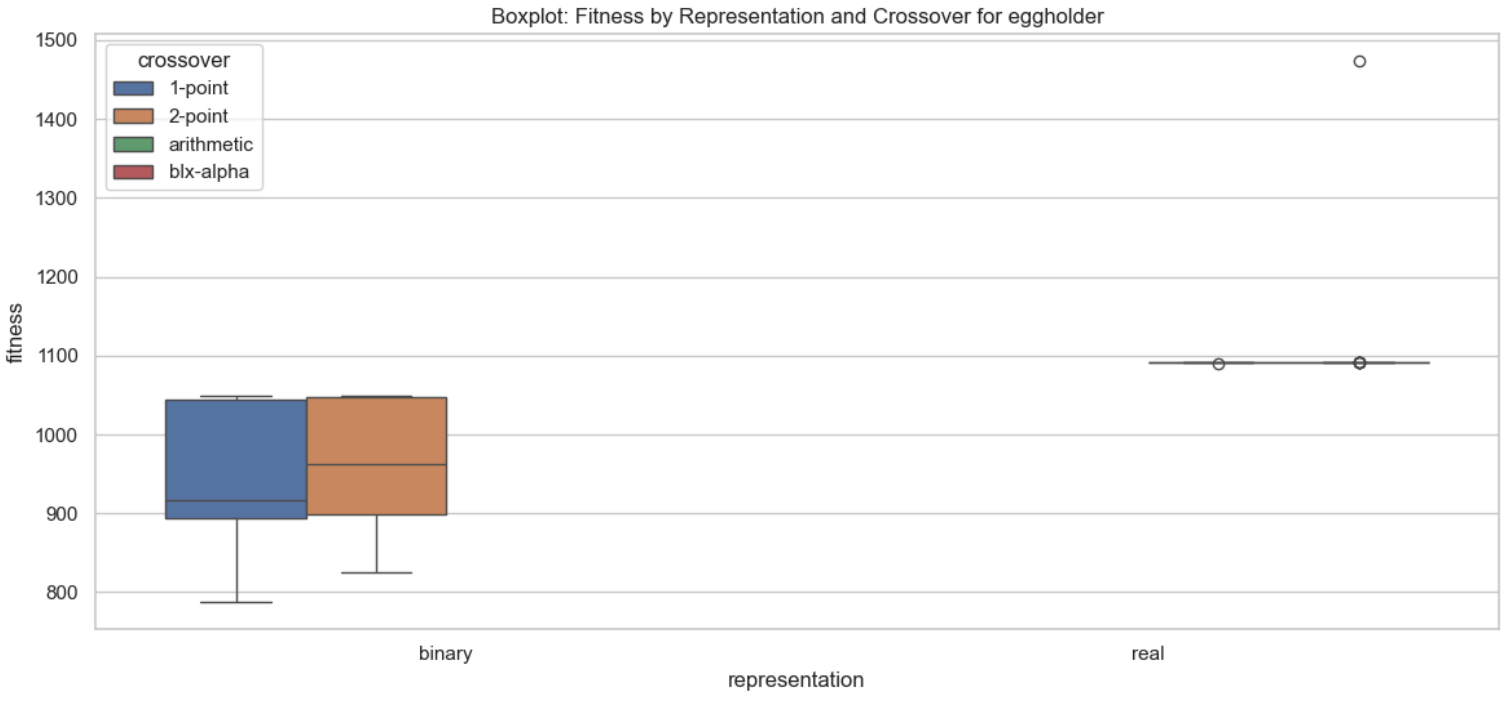
05/06/2025

Bărnuț Cristiana Florina Diana
Group: 921

2. Eggholder Function:

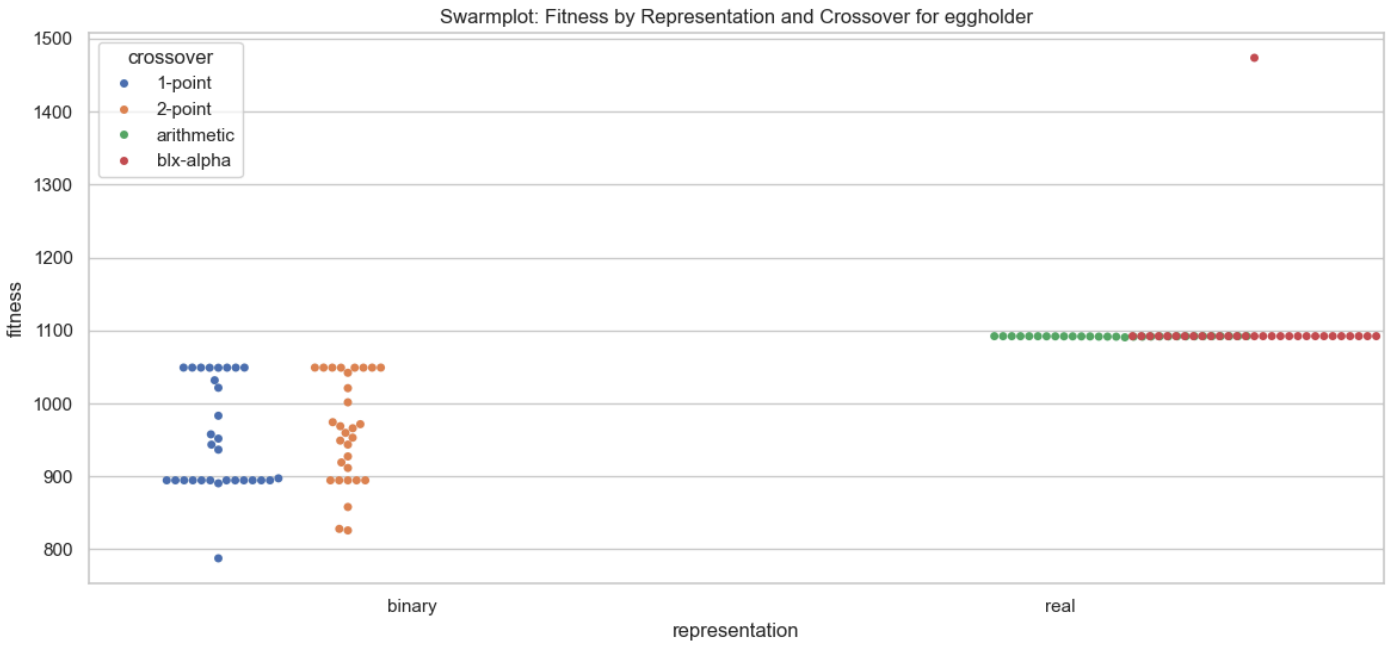
- For **binary encoding** with *1-point* and *2-point* crossover, the fitness values ranged widely, typically around 800 to 1050, indicating the algorithm reached suboptimal solutions due to the complexity of the function.
- With **real-valued representation** using *arithmetic* crossover, the fitness values clustered around 1091-1092, suggesting better optimization performance compared to binary encoding.
- The **BLX-alpha crossover** for real-valued representation showed stable fitness values near 1092, with occasional outliers (e.g., 1473), possibly due to run variability.

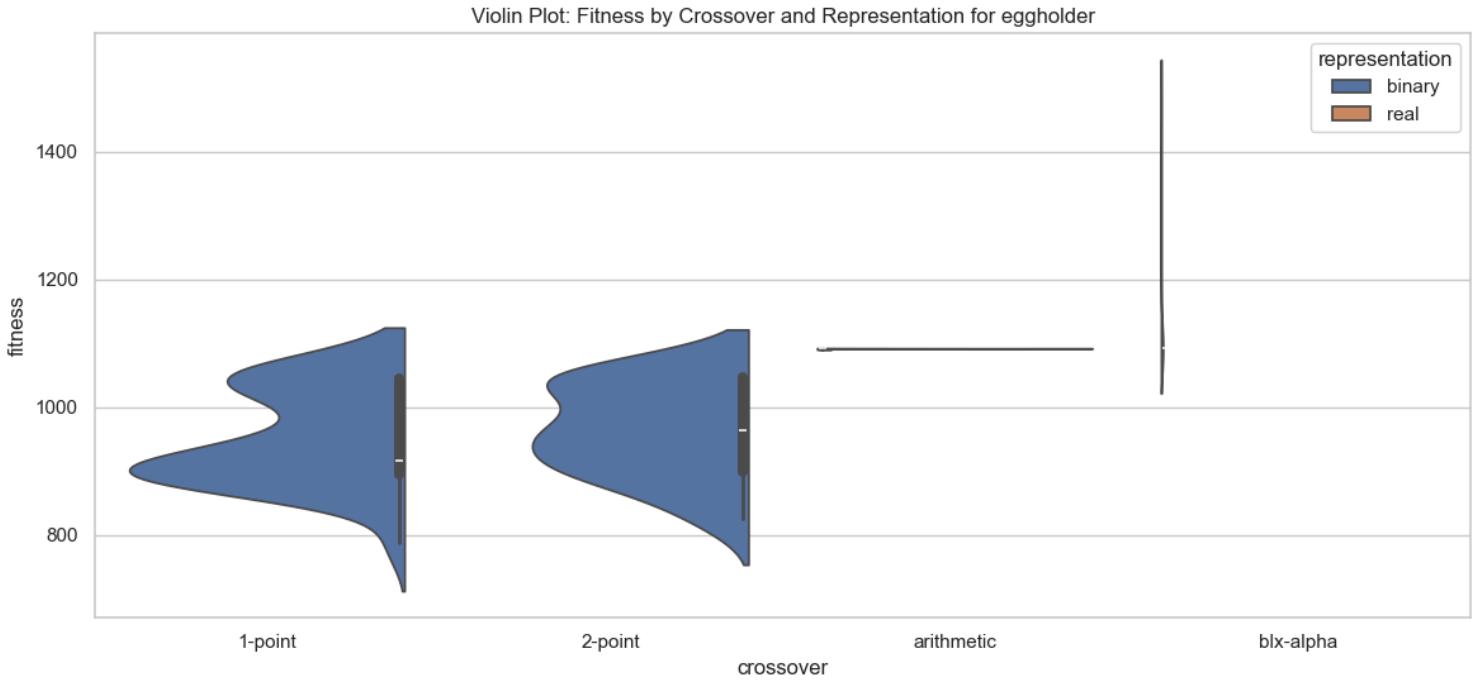
Representation	Crossover	Min Fitness	Max Fitness	Mean Fitness
binary	1-point	787.63	1049.13	948.37
binary	2-point	825.89	1049.13	944.55
real	arithmetic	1090.66	1092.21	1091.92
real	blx-alpha	1092.19	1473.80	1102.91



05/06/2025

Bărnuț Cristiana Florina Diana
Group: 921





05/06/2025

Bărnuț Cristiana Florina Diana
Group: 921

5. Statistical Analysis

We performed pairwise statistical comparisons of fitness values between all combinations of algorithm configurations, defined by representation and crossover method. Both **independent samples t-tests** and **Wilcoxon rank-sum tests** were used to assess differences in fitness distributions.

- **Independent Samples T-Test**

  This parametric test compares the means of two independent groups, assuming normality and equal variances. It is sensitive to outliers and may be less reliable if assumptions are violated.

- **Wilcoxon Rank-Sum Test**

  A non-parametric alternative that compares the distributions of two independent groups without assuming normality. It is more robust to outliers and skewed data and focuses on median differences.

1. Drop-wave function

Comparison	T-test p-value	Wilcoxon p-value	Interpretation
binary 1-point vs binary 2-point	0.1576	0.6100	No significant difference between binary crossover types.
binary 1-point vs real arithmetic	0.3857	1.7739e-09	Means are similar, but distributions differ significantly — **real arithmetic performs more consistently or has fewer outliers**.
binary 1-point vs real blx-alpha	3.0185e-05	9.4449e-11	Both tests show **real BLX-alpha significantly outperforms** binary 1-point.
binary 2-point vs real arithmetic	0.5318	1.7739e-09	Again, the mean difference isn't clear, but **real-coded arithmetic is likely more robust or has better median performance**.
binary 2-point vs real blx-alpha	0.0363	5.2283e-11	BLX-alpha shows a **clear performance advantage** over binary 2-point.
real arithmetic vs real blx-alpha	0.3252	0.2972	No statistically significant difference between the two real-coded crossover methods.

Bărnuț Cristiana Florina Diana
Group: 921

2. Eggholder function

Comparison	T-test p-value	Wilcoxon p-value	Interpretation
binary 1-point vs binary 2-point	0.5307	0.5493	No significant difference between binary crossover types.
binary 1-point vs real arithmetic	2.9612e-11	2.8719e-11	Real-coded arithmetic **significantly outperforms** binary 1-point.
binary 1-point vs real blx-alpha	2.2641e-11	2.8719e-11	BLX-alpha crossover is **clearly superior** to binary 1-point.
binary 2-point vs real arithmetic	8.7193e-11	2.8719e-11	Real arithmetic **outperforms** binary 2-point crossover.
binary 2-point vs real blx-alpha	1.3697e-10	2.8719e-11	BLX-alpha **significantly better** than binary 2-point.
real arithmetic vs real blx-alpha	0.13140	3.1752e-11	Means may not differ, but **distribution/medians do** — BLX-alpha likely has better robustness or consistency.

6. Conclusion

This study investigated the impact of different genetic algorithm configurations — focusing on **representation types** (binary vs. real-coded) and **crossover methods** — across two benchmark optimization functions: **Eggholder** and **Drop-Wave**.

Key Takeaways:

- **Real-coded representations consistently outperformed binary representations** in both functions, especially when paired with the **BLX-alpha** or **arithmetic crossover** methods.

- **Statistical analysis** using both **parametric (T-test)** and **non-parametric (Wilcoxon rank-sum)** tests revealed:

    - No significant performance differences between **1-point and 2-point crossover** within binary representations.

    - **Significant performance gains** with real-coded methods compared to binary ones — particularly notable in **Eggholder**, where all comparisons with binary were statistically significant ($p < 0.05$).

    - The **Wilcoxon test often detected differences not caught by the T-test**, suggesting that **fitness distributions were not always normally**

distributed, and emphasizing the importance of using both types of statistical tests.

- No significant difference was observed **between real-coded crossover methods** (arithmetic vs. BLX-alpha), indicating both are viable and competitive.

Final Remarks:

Overall, the results strongly support the use of **real-coded representations with either arithmetic or BLX-alpha crossover** for continuous optimization problems like Eggholder and Drop-Wave. The statistical analysis underscores the importance of selecting appropriate testing methods and reveals subtle performance dynamics not captured by mean comparisons alone.

7. References

★ Drop-Wave function: https://www.sfu.ca/~ssurjano/drop.html [01/06/2025]
★ Eggholder function: https://www.sfu.ca/~ssurjano/egg.html [01/06/2025]
★ Pandas: https://pandas.pydata.org/ [02/06/2025]
★ Seaborn: https://seaborn.pydata.org/ [02/06/2025]
★ Matplotlib: https://matplotlib.org/ [01/06/2025]
★ Scipy: https://scipy.org/ [02/06/2025]
★ Numpy: https://numpy.org/ [01/06/2025]