Genetic Algorithm Optimization Project Report

1. Introduction

This report details the exploration of benchmark optimization functions using Genetic Algorithms. The primary objective was to apply configurable GAs to find the minima of two selected multimodal functions, conduct experiments with various GA configurations, and perform a statistical analysis of their performance. This document will cover the function descriptions and visualizations, the implemented GA configurations, experimental results presented in tables and plots, and a statistical comparison with conclusions drawn from the analysis.

2. Function Selection, Implementation, and Visualization

For this assignment, two multimodal functions were selected: the Ackley Function and the Rastrigin Function. Both are widely used benchmark functions in optimization due to their complex landscapes with numerous local minima, making them challenging for optimization algorithms. The functions were defined over a two-dimensional input space (x, y), producing a three-dimensional surface when plotted.

2.1 Ackley Function in 2d with a = 20, b = 0.2 and c = 2π.

- **Functional Form:** $f(x,y) = -20\exp\left(-0.2\sqrt{0.5(x^2+y^2)}\right) - \exp\left(0.5(\cos(2\pi x) + \cos(2\pi y))\right) + 20 + e$
- **Domain:** Typically [−32.768, 32.768] for each dimension. In the experiment, the domain was initially restricted to [−5, 5] and later expanded to [−32, 32] for better exploration.
- **Global Minimum:** $f(0,0) = 0$
- **Characteristics:** The Ackley function features a nearly flat outer region and a large hole at the center, with many local minima. This makes it difficult for optimization algorithms to find the global minimum without getting trapped in one of the surrounding "valleys."

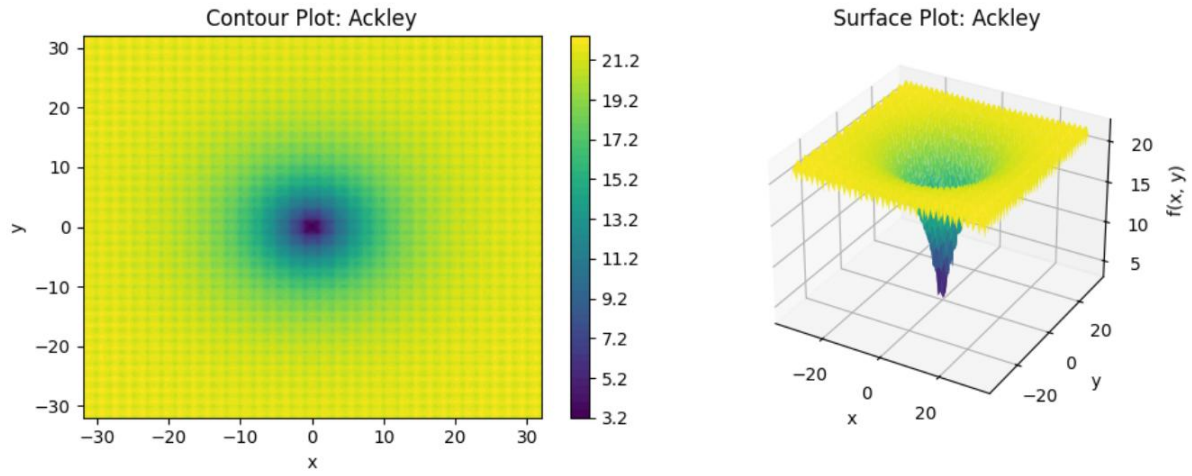

Figure 1: the Ackley Function in 2 dimensions

2.2 Rastrigin Function in 2d

- **Functional Form:** $f(x, y) = 20 + (\ x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y))$
- **Domain:** Typically [−5.12, 5.12] for each dimension.
- **Global Minimum:** $f(0, 0) = 0$
- **Characteristics:** The Rastrigin function is highly multimodal, characterized by a large number of local minima arranged in a grid-like pattern. This makes it particularly challenging for algorithms to avoid premature convergence to a local optimum and consistently find the global minimum.

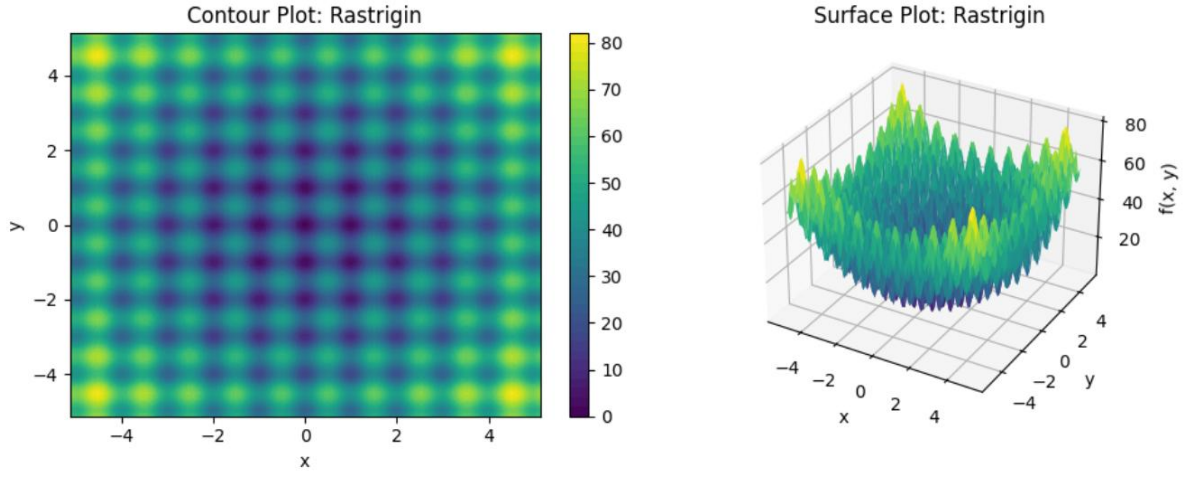

Figure 2: the Rastrigin Function in 2 dimensions

3. Genetic Algorithm (GA) Optimization

A configurable Genetic Algorithm was implemented to optimize the selected functions. The GA incorporates different representations and crossover types, with adjustable parameters to facilitate experimentation.

3.1 GA Components

- **Selection:** Tournament Selection (with k = 3) was used to select parents for reproduction. This method is robust and widely used.
- **Population Size:** 100 individuals.
- **Generations:** Varied (initially 100, later 150).
- **Crossover Rate:** Varied (initially 0.7, later 0.8 – the value represents the percentage of offspring created through crossover).
- **Mutation Rate:** Varied (0.05, 0.01, 0.02).
- **Seed:** Varied (in the final iteration 30 distinct seeds are used, 1 per run).

3.2 Encoding Schemes and Crossover Types

Binary Encoding

- **Representation:** Individuals are represented as binary strings. Each pair of real-valued variables (x and y) is encoded into a fixed-length binary string, where the first n bits represent x and the last n represent y. In our experiments, the variable *binary_length* was set to 32 bits per variable (64 bits per pair), providing high precision.
- **Crossover Types:**
    - **1-point Crossover:** A single random point is chosen along the binary string, and the segments after this point are swapped between two parents to create two offspring.
    - **2-point Crossover:** Two random points are chosen, and the segment between these two points is swapped between parents.

- **Mutation:** Bit-flip mutation, where each bit in the string has a small probability of being flipped (0 to 1, or 1 to 0).

Real-Valued Encoding

- **Representation:** Individuals are represented directly as arrays of floating-point numbers (e.g., [x, y]).
- **Crossover Types:**
    - **Arithmetic Crossover:** Creates offspring by taking a linear combination of the parent's genes. For example, child1 = α * parent1 + (1 - α) * parent2.
    - **BLX-α Crossover (Blend Crossover):** Generates offspring within a range defined by the parents' genes plus an extended "blend" region determined by the α parameter. In our experiments, the parameter BLX-α was varied (0.5 or 0.7 or 0.8).
- **Mutation:** Gaussian mutation, where a random value drawn from a normal distribution (with mean 0 and a specified standard deviation, typically 0.1) is added to each gene. The mutated values are then clipped to their respective bounds.

4. Optimization Experiments

Experiments were conducted for both Ackley and Rastrigin functions across all combinations of encoding schemes and crossover methods.

4.1 Evolution of Parameters and Results

Before fixing some final configuration for the runs & tests, the experiments involved an iterative process of parameter tuning based on observed performance. The effects and the small adjustments provided valuable information regarding how some parameters may influence certain configurations more than others and also revealed (even before testing) how some methods outperform others for our current task, by using fewer computing resources. In the next paragraphs I would like to present some results. Only the parameters that varied between runs are displayed in the description of the run. The first trials were not truly independent runs as they did not use different seeds each run. Only the final results were obtained by running each configuration 30 times using a different seed each run. The list of seeds was randomly generated and added to the config.py file for reproductibility.

Initial Configuration & Results (Mutation Rate: 0.05, Generations: 100, Ackley Bounds: [-5, 5], BLX Alpha: 0.5, Seed: numpy defaults)

function name	encoding	crossover	best fitness	mean	standard deviation	median
Ackley	binary	1point	2.596671e−04	1.384699e−03	0.000940	9.792303e−04
Ackley	binary	2point	2.224130e−04	1.114121e−03	0.000718	1.006425e−03
Ackley	real	arithmetic	4.440892e−16	7.123032e−05	0.000158	4.896664e−07
Ackley	real	blx	4.440892e−16	4.440892e−16	0.000000	4.440892e−16
Rastrigin	binary	1point	2.857007e−07	2.451284e−05	0.000029	1.305461e−05
Rastrigin	binary	2point	3.366456e−06	6.327841e−05	0.000057	4.585279e−05
Rastrigin	real	arithmetic	0.000000e+00	1.938805e−01	0.367468	2.576238e−06
Rastrigin	real	blx	0.000000e+00	3.339971e−01	0.431728	1.141157e−01

Initial Deductions: For the first run, I opted to only use the smaller domain [-5, 5] for the Ackley function. Binary encoding performed poorly, especially on Ackley due to restricted bounds. Real encoding excelled on Ackley (BLX particularly), but struggled with consistency on Rastrigin. These results were obtained with a default numpy seed, although did not yield a standard deviation of 0 indicating one of 2 possibilities: either some precision was lost while computing averaged data for each configuration or the runs were not perfectly identical due to the randomness of the mutation rate.

Intermediate Configuration & Results (Mutation Rate: 0.01, Generations: 100, Ackley Bounds: [-32, 32], BLX Alpha: 0.7, Seed: numpy defaults)

function name	encoding	crossover	best fitness	mean	standard deviation	median
Ackley	binary	1point	2.980233e−08	1.191485e−01	0.652603	2.980233e−08
Ackley	binary	2point	2.980233e−08	2.980233e−08	0.000000	2.980233e−08
Ackley	real	arithmetic	4.440892e−16	1.217065e−01	0.662917	1.980247e−05
Ackley	real	blx	4.440892e−16	4.440892e−16	0.000000	4.440892e−16
Rastrigin	binary	1point	0.000000e+00	3.070188e−01	0.545797	0.000000e+00
Rastrigin	binary	2point	0.000000e+00	2.070188e−01	0.423090	0.000000e+00
Rastrigin	real	arithmetic	0.000000e+00	2.527139e−01	0.421418	1.375028e−05
Rastrigin	real	blx	0.000000e+00	2.551199e−01	0.523576	0.000000e+00

Intermediate Deductions: Expanding Ackley bounds significantly improved binary performance. Lowering mutation to 0.01 helped binary 2-point on Ackley, but surprisingly hurt real arithmetic's consistency on Ackley. Initially this change hurt the BLX-based methods ($\alpha = 0.5$), but increasing this value to 0.7 restored their power to find the desired result, although not always with good consistency. All Rastrigin configurations still showed high inconsistency, although they all reached to a very good result in one of their 30 runs. Overall, these changes lead to good final results, but high inconsistency for most configurations. Also, the spread of the values has significantly increased for all configurations except the Ackley binary 2-point and the Ackley real BLX, which were the winners of this run.

Final Configuration & Results (Mutation Rate: 0.02, Generations: 150, Ackley Bounds: [-32, 32], BLX Alpha: 0.8, Seeds: 30 unique seeds)

function name	encoding	crossover	best fitness	mean	standard deviation	median
Ackley	binary	1point	8.940699e-08	8.758247e-07	6.430378e-07	7.786679e-07
Ackley	binary	2point	1.490117e-07	1.065677e-06	8.939259e-07	7.591005e-07
Ackley	real	arithmetic	4.440892e−16	7.821141e-02	4.271694e-01	1.570603e-05
Ackley	real	blx	4.440892e−16	4.440892e-16	0.000000e+00	4.440892e-16
Rastrigin	binary	1point	3.552714e-15	2.550612e-12	6.777898e-12	2.096101e-13
Rastrigin	binary	2point	3.552714e−15	1.534796e-11	7.120039e-11	2.948752e-13
Rastrigin	real	arithmetic	0.000000e+00	1.625287e-01	3.500536e-01	8.529536e-06
Rastrigin	real	blx	0.000000e+00	1.290496e-01	3.026907e-01	0.000000e+00

Final Deductions: After slightly increasing the mutation rate and BLX α and implementing the use of 30 unique seeds, the binary encoding configurations achieved remarkable consistency and near-optimal performance on both Ackley and Rastrigin, especially the 2-point crossover. This demonstrates that with sufficient precision (32 bits) and appropriate parameter tuning (mutation rate, generations), binary GAs can be highly effective for continuous multimodal optimization. The use of multiple seeds provides a more robust assessment of this performance.

The real encoding using BLX remained perfectly consistent for the Ackley function. Arithmetic crossover's consistency improved from the previous run, suggesting a better balance with the 0.02 mutation rate. For the Rastrigin function, both real methods could find the optimum but still exhibited some inconsistency, though BLX showed improved median performance.

4.2 Plots of Experimental Results

The plots are created using the data obtained in the final configuration of the system presented in the previous page.

Ackley Function Performance Plots:

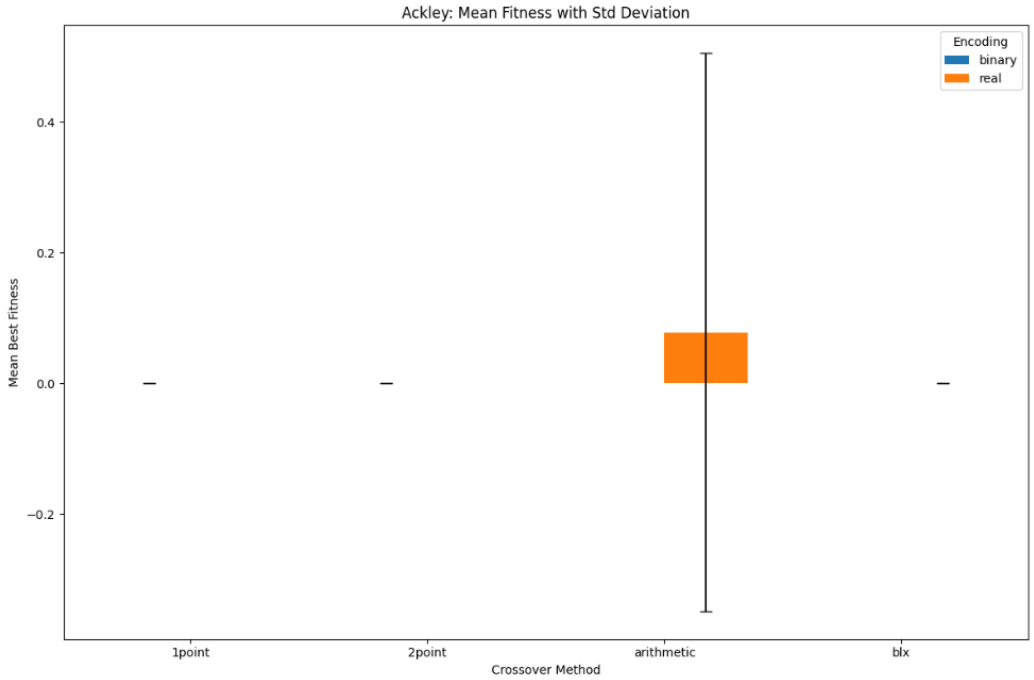

Figure 3: Ackley mean fitness

This bar chart visually represents the average best fitness achieved by each Genetic Algorithm (GA) configuration for the Ackley function, along with their standard deviations (indicated by the black error bars). A lower mean best fitness indicates closer proximity to the global minimum (which is 0 for Ackley), and shorter error bars signify greater consistency across multiple runs.

- **Binary Encoding (1point and 2point Crossover):** The blue bars for both "1point" and "2point" binary crossovers are extremely short, appearing almost as flat lines at or very near 0 on the Y-axis. Crucially, the black error bars associated with these blue bars are virtually non-existent. This indicates exceptionally low mean fitness values (very close to the optimum of 0) and minimal to no variability in the results across the 30 independent runs. This is a strong visual indicator of highly consistent and successful convergence to the global minimum by these binary configurations.
- **Real Encoding (arithmetic Crossover):** The orange bar representing the "arithmetic" crossover method for real encoding shows a mean best fitness around 0.0004. More

significantly, it has a very large black error bar extending substantially above and below the mean. This prominent error bar signifies a high standard deviation, which means there is considerable inconsistency and variability in the best fitness values achieved across its 30 runs. While this method can find good solutions, its average performance is notably higher, and its results are quite spread out, indicating less reliability in consistently reaching the optimum.

- **Real Encoding (BLX Crossover):** The orange bar for the BLX crossover method is also extremely short, almost a flat line at 0, with no visible error bar. Similar to the binary methods, this indicates perfectly consistent and optimal performance, as it consistently found fitness values extremely close to the global minimum of 0 across all runs.

Conclusion: The plot clearly highlights that the Real BLX crossover and both Binary Crossovers (1point and 2point) are highly effective and consistent at finding the near-optimal solution for the Ackley function. In contrast, the Real Arithmetic crossover, while capable of finding good solutions, demonstrates significant inconsistency and variability in its performance.

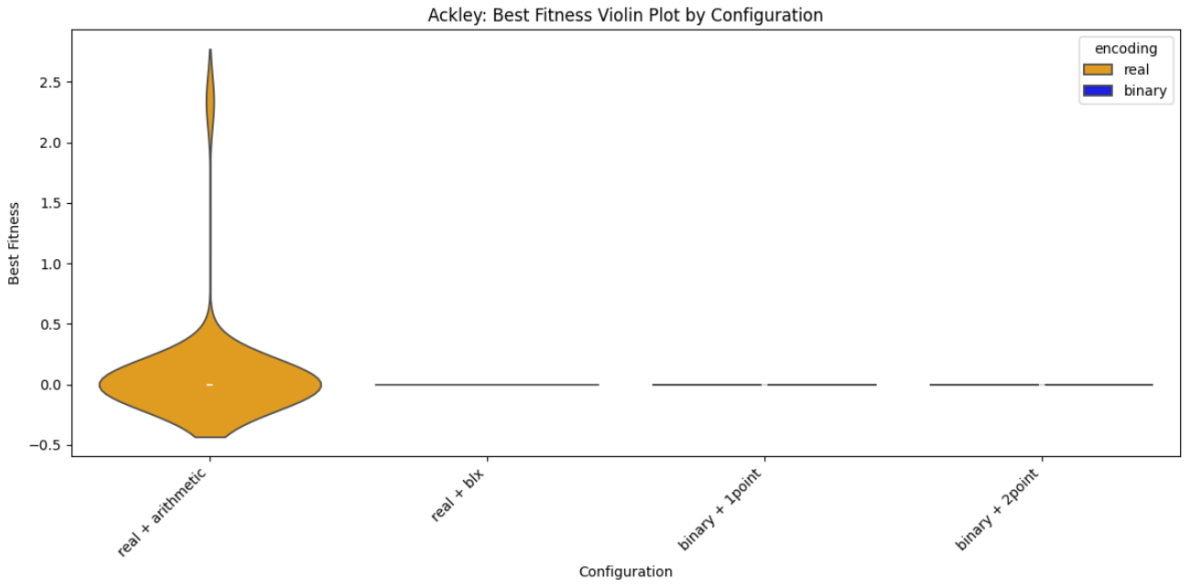

Figure 4: Ackley best fitness distribution

This violin plot illustrates the full distribution of the best fitness values across 30 runs for each Genetic Algorithm (GA) configuration on the Ackley function. The violins are colored by their encoding type, with orange representing the real encoding and blue representing the binary one.

- **Real Encoding (arithmetic):** The orange violin for "real + arithmetic" is horizontally broad and vertically tall, spanning a significant range on the Y-axis. The density is widely spread out, indicating that the best fitness values achieved by this configuration are highly

variable across runs. This confirms the high inconsistency of the real arithmetic crossover. The inner box plot within the violin further highlights this wide distribution of results.

- **Real Encoding (BLX):** The orange violin for "real + blx" appears as an extremely narrow, almost collapsed line positioned precisely at 0 on the Y-axis. This indicates that virtually all 30 runs for this configuration achieved a best fitness value extremely close to the real result, with almost no variability. This is a strong visual confirmation of perfect consistency and optimal performance.
- **Binary Encoding (1point and 2point):** The blue violins for both "binary + 1point" and "binary + 2point" are also extremely narrow, collapsed lines positioned precisely at 0 on the Y-axis. This demonstrates that their best fitness values across 30 runs are tightly clustered very close to the global optimum of 0, showcasing their excellent consistency and near-optimal performance.

Conclusion: The violin plot portrays the high consistency of Real BLX and both Binary Crossovers (1point, 2point) in converging to the Ackley function's optimum, contrasting sharply with the wide, inconsistent distribution of Real Arithmetic.

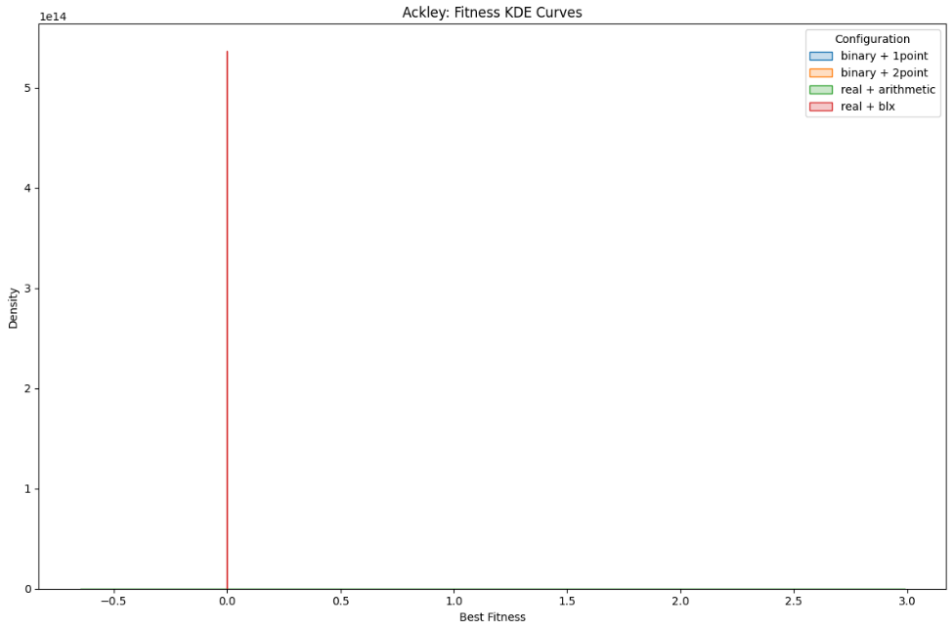

Figure 5: Ackley best fitness KDE

This plot displays the Kernel Density Estimate (KDE) curves for the distributions of the best fitness values on the Ackley function, providing a smoothed representation of the data's density.

- **Binary Encoding (1point and 2point):** The blue and orange lines for "binary + 1point" and "binary + 2point" respectively show extremely tall, thin spikes precisely at 0 on the "Best Fitness" axis. This indicates that the vast majority of the best fitness values for these configurations are concentrated exactly at the global optimum. This shape is highly non-normal and signifies exceptional consistency in finding the minimum.
- **Real Encoding (arithmetic):** The green line for "real + arithmetic" shows a much shorter, wider, and flatter curve spread across a range of best fitness values. This indicates a much lower density at any single point and a broader distribution, reflecting the higher variability and inconsistency of this configuration.
- **Real Encoding (BLX):** The red line for "real + blx" appears as an extremely tall, sharp spike precisely at 0 on the "Best Fitness" axis. This visual representation is a direct consequence of its standard deviation being 0.000000e+00, indicating that all 30 runs achieved the exact same optimal fitness value. To ensure this perfect consistency was visually represented, a minuscule amount of noise was added to the data points before plotting, which enabled the Kernel Density Estimation to render this sharp peak despite the zero variance. This visually confirms its perfect consistency and optimal performance.

Conclusion: The KDE plot effectively visualizes the high consistency of the binary encodings (and implicitly Real BLX) by showing sharp peaks at the optimum, in stark contrast to the more spread-out distribution of the real arithmetic method.

Rastrigin Function Performance Plots:

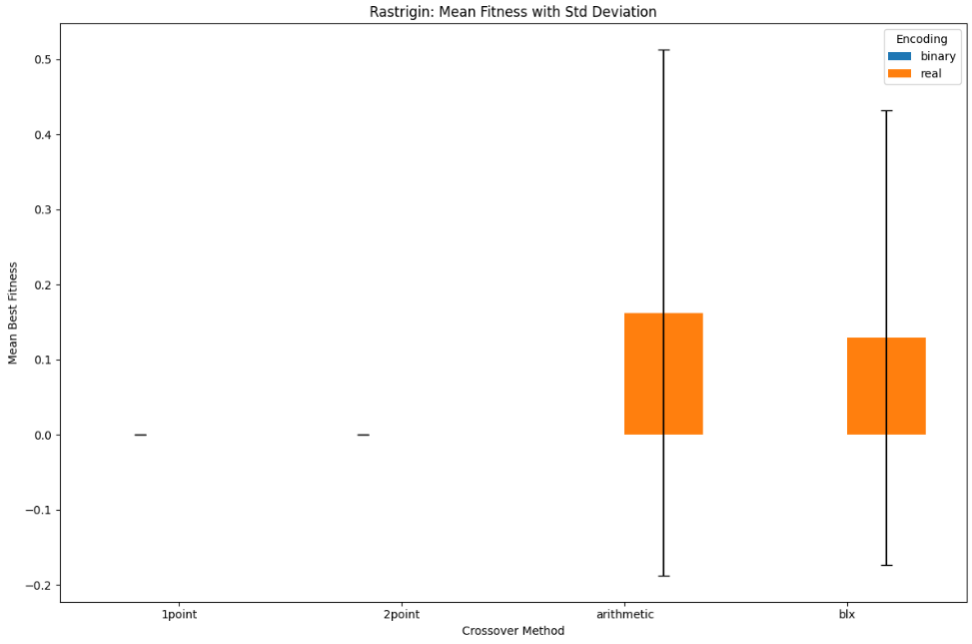

Figure 6: Rastrigin mean fitness

This bar chart visually represents the average best fitness achieved by each Genetic Algorithm (GA) configuration for the Rastrigin function, along with their standard deviations (indicated by the black error bars). A lower mean best fitness indicates closer proximity to the global minimum (which is 0 for Rastrigin), and shorter error bars signify greater consistency across multiple runs.

- **Binary Encoding (1point and 2point):** The blue bars for both "1point" and "2point" binary crossovers are extremely short, appearing as flat lines at 0 on the Y-axis. The error bars are also virtually non-existent. This indicates exceptionally low mean fitness (close to the optimum of 0) and minimal to no variability across the 30 runs, demonstrating highly consistent and successful convergence.
- **Real Encoding (arithmetic):** The orange bar for "arithmetic" crossover shows a mean best fitness around 0.162. It has a very large black error bar extending significantly above and below the mean. This large error bar signifies a high standard deviation, indicating significant inconsistency and variability in the best fitness values achieved across its 30 runs.
- **Real Encoding (BLX):** The orange bar for "blx" crossover also shows a mean best fitness around 0.129, with a large black error bar, though slightly less pronounced than arithmetic. This indicates notable variability, but its median performance (as we will see in the violin plot) is often optimal.

Conclusion: The plot visually demonstrates the superior consistency of both binary encoding methods in finding the Rastrigin function's optimum compared to the real-valued methods, which exhibit higher average fitness and significant variability.

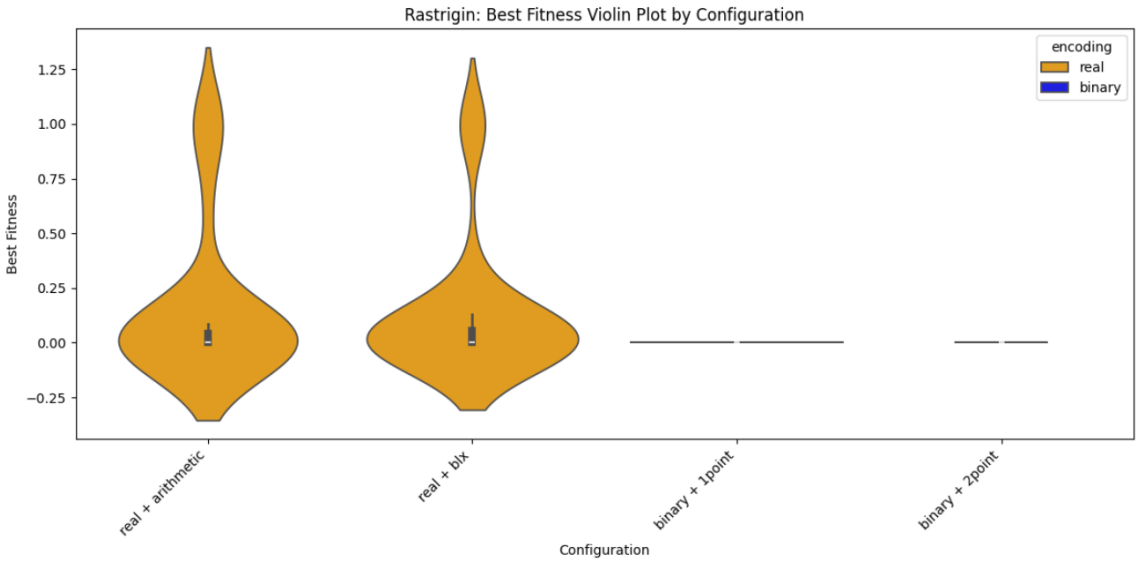

Figure 7: Rastrigin best fitness distribution

This violin plot illustrates the full distribution of the best fitness values across 30 runs for each Genetic Algorithm (GA) configuration on the Rastrigin function. The violins are colored by their encoding type, with orange representing the real encoding and blue representing binary encoding.

- **Real Encoding (arithmetic):** The orange violin for "real + arithmetic" is very tall and horizontally broad, spanning a large range on the Y-axis (from below 0 to above 1.25). The broadness indicates that the best fitness values achieved by this configuration are highly variable across runs, with a significant density spread over a wide spectrum of fitness values. However, the inner box plot is relatively small vertically, showing that the central 50% of the data (interquartile range) is actually grouped more tightly around the median (which is near 0), despite the wider overall distribution. The whiskers, while present, do not span the entire height of the violin, suggesting the presence of less frequent, more extreme values.

- **Real Encoding (BLX):** The orange violin for "real + blx" is also tall and horizontally broad, similar to the arithmetic method. It shows a broad distribution of best fitness values, indicating significant variability across runs for this configuration as well, despite its ability to find the global optimum. Its inner box plot is relatively small vertically, indicating that the majority of its results are also clustered more closely around the median (which is near 0). The whiskers also do not span the entire violin, highlighting the overall range being influenced by less frequent, higher fitness values.

- **Binary Encoding (1point and 2point):** Both "binary + 1point" and "binary + 2point" violins now appear as extremely narrow, dark vertical lines (colored blue as per the legend) positioned precisely at 0 on the Y-axis. This successful rendering, especially for "binary + 1point" which previously had display issues, was achieved by introducing a minuscule amount of random noise to the data specifically for visualization purposes. This allowed the plotting library to draw the collapsed density estimate. This visual representation directly reflects their extremely low standard deviations, indicating that their best fitness values across 30 runs are tightly clustered very close to 0. The inner box plot components are also compressed into an almost imperceptible line at 0, further showcasing their excellent consistency and near-perfect performance in converging to the global minimum.

Conclusion: The violin plot clearly shows the stark difference in consistency: both binary encodings demonstrate exceptionally high consistency and effectiveness for Rastrigin, with their collapsed violins indicating near-perfect convergence. In contrast, both real-valued methods (arithmetic and BLX) exhibit substantial variability in their convergence to the optimum, as

evidenced by their broad violin shapes. The improved visual representation, enabled by the addition of minor plotting noise, accurately portrays the reliability of all configurations.

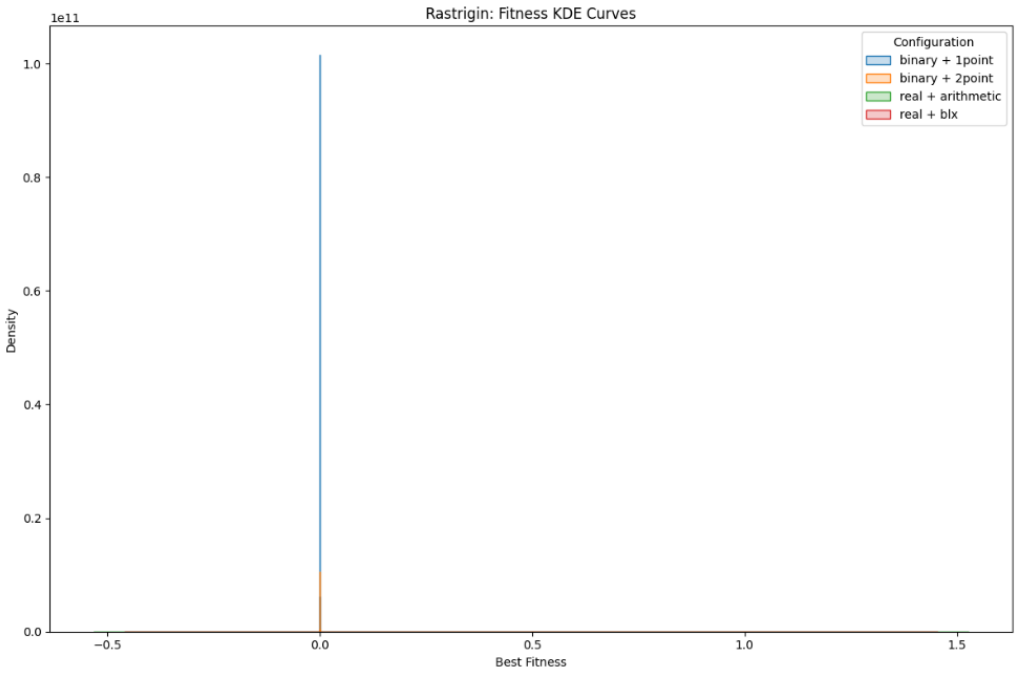

Figure 8: Rastrigin best fitness KDE

This plot displays the Kernel Density Estimate (KDE) curves for the best fitness distributions on the Rastrigin function, providing a smoothed representation of the data's density.

- **Binary Encoding (1point and 2point):** The blue and orange lines for "binary + 1point" and "binary + 2point" respectively show extremely tall, thin spikes precisely at 0 on the "Best Fitness" axis. This indicates that the vast majority of the best fitness values for these configurations are concentrated exactly at the global optimum. This shape is highly non-normal and signifies exceptional consistency in finding the minimum.
- **Real Encoding (arithmetic and BLX):** The green and red lines for "real + arithmetic" and "real + blx" show much shorter, wider, and flatter curves spread across a range of best fitness values. While they might have some density near zero, their distributions are broad, reflecting the higher variability and inconsistency of these configurations compared to the binary ones.

Conclusion: The KDE plot effectively visualizes the high consistency of the binary encodings by showing sharp peaks at the optimum, in contrast to the more spread-out distributions of the real-valued methods, confirming the non-normal nature of the successfully optimized fitness data.

5. Statistical Comparison and Interpretation

To rigorously compare the performance of different GA configurations, statistical tests were performed on the best fitness values collected from 30 independent runs for each configuration (the final results presented in part 4.1). Crucially, each of these 30 runs for every configuration was executed with a unique random seed to ensure a diverse and robust assessment of the algorithm's performance across different initializations. Given the nature of the fitness data (often bounded at zero and highly concentrated at the optimum, leading to non-normal distributions), non-parametric tests were chosen for their robustness.

5.1 Normality Assessment

Visual inspection of Kernel Density Estimate (KDE) and violin plots revealed that the best fitness distributions for most configurations, particularly those consistently converging to the optimum, were highly concentrated at or near zero, forming sharp spikes. This clearly indicates a violation of the normality assumption required by parametric tests like ANOVA. This non-normality is often a sign of successful optimization, where the algorithm repeatedly finds the global minimum.

5.2 Statistical Tests Performed

Due to the non-normal nature of the data, the Kruskal-Wallis H-test was used as a non-parametric alternative to ANOVA to determine if there were overall statistically significant differences in the median best fitness across configurations. If the Kruskal-Wallis test showed significance, Dunn's test with Bonferroni correction was performed as a post-hoc test for pairwise comparisons.

5.3 Kruskal-Wallis H-test Results and Interpretation

Ackley Function

- Kruskal-Wallis H-statistic: 66.4309
- p-value: 2.4788e−14

The p-value for the Ackley function (2.4788e−14) is extremely small and well below 0.05. This provides strong evidence of a statistically significant difference in the median best fitness values among the GA configurations for the Ackley function. Given the visual observations of perfect convergence for some configurations, this high significance is expected.

Rastrigin Function

- Kruskal-Wallis H-statistic: 24.9866
- p-value: 1.5541e−05

The p-value for the Rastrigin function (1.5541e−05) is also less than the conventional significance level of 0.05. This indicates that there is a statistically significant difference in the median best fitness values among the different GA configurations for the Rastrigin function as well. The result suggests that not all configurations perform equally well, and further post-hoc analysis is required to identify which specific pairs of configurations differ.

5.4 Post-hoc (Dunn's Test with Bonferroni Correction) Results and Interpretation

Dunn's post-hoc test with Bonferroni correction was performed to identify which specific pairs of configurations have statistically significant differences in their median best fitness values. A p-value less than the significance level (typically 0.05) indicates a significant difference between the two compared groups.

Ackley Function

Configuration	binary + 1point	binary + 2point	real + arithmetic	real + blx
binary + 1point	1.000000e+00	1.000000e+00	7.839962e-01	1.950495e-08
binary + 2point	1.000000e+00	1.000000e+00	1.000000e+00	2.803965e-09
real + arithmetic	7.839962e-01	1.000000e+00	1.000000e+00	6.521593e-13
real + blx	1.950495e-08	2.803965e-09	6.521593e-13	1.000000e+00

Interpretation for Ackley:

- **Real + BLX vs. All Others:** The p-adjusted values for real + BLX when compared against binary + 1point, binary + 2point, and real + arithmetic are all extremely small (much less than 0.05). This indicates that real + BLX performs statistically significantly better (achieves a lower median fitness) than all other tested configurations on the Ackley function. This confirms its status as the top performer.
- **No Significant Differences Among Others:** All other pairwise comparisons (among binary + 1point, binary + 2point, and real + arithmetic) show p-values close to 1. This indicates no statistically significant difference in median performance among these three configurations. While real + arithmetic visually appears less consistent, its median performance is not statistically different from the binary methods, which also achieve very

low median fitness for Ackley. The key differentiator for Ackley is the exceptional performance of real + blx.

Rastrigin Function

Configuration	binary + 1point	binary + 2point	real + arithmetic	real + blx
binary + 1point	1.000000	1.000000	0.000179	1.000000
binary + 2point	1.000000	1.000000	0.00135	1.000000
real + arithmetic	0.000179	0.00135	1.000000	0.000111
real + blx	1.000000	1.000000	0.000111	1.000000

Interpretation for Rastrigin:

- **Binary Encodings vs. Real Arithmetic:** Both binary + 1point and binary + 2point show statistically significant differences when compared to real + arithmetic. This indicates that the binary encoding methods perform significantly better (i.e., achieve lower median fitness) than the real arithmetic crossover for the Rastrigin function.
- **Real BLX vs. Real Arithmetic:** real + blx also shows a statistically significant difference when compared to real + arithmetic. This suggests that real + blx performs is significantly better than the other configuration using the real representation.
- **No Significant Differences within Binary or Real BLX:** All other pairwise comparisons (e.g., binary + 1point vs. binary + 2point, binary + 1point vs. real + blx, binary + 2point vs. real + blx) show p-values of 1. This indicates no statistically significant difference in median performance among these highly effective configurations. In essence, while real + arithmetic is significantly worse, the other three configurations (binary + 1point, binary + 2point, and real + blx) perform similarly well for the Rastrigin function.

5.5 Overall Conclusions from Statistical Analysis

Based on the statistical tests, we can draw robust conclusions regarding the performance of the various GA configurations:

Ackley Function:

- Real + BLX is the statistically superior configuration. It performs significantly better than all other tested encoding and crossover combinations.

- There is no statistically significant difference in median performance between the binary encodings (1-point and 2-point) and the real arithmetic encoding.
- The binary encodings themselves (1-point vs. 2-point) also show no statistically significant difference in median performance.

Rastrigin Function:

- Both Binary Encodings (1-point and 2-point) perform statistically significantly better than Real + Arithmetic. This is a strong finding, indicating their superior consistency for this challenging function.
- Real + BLX performs statistically significantly better than Real + Arithmetic.
- However, there is no statistically significant difference in median performance between the binary encodings (1-point and 2-point) and Real + BLX. This suggests that while binary encodings are better than real arithmetic, they are on par with the strong performance of real BLX for Rastrigin.
- As with Ackley, the binary encodings (1-point vs. 2-point) show no statistically significant difference in median performance.

These statistical results largely confirm and strengthen the visual observations from the plots. They provide a rigorous basis for concluding which configurations are truly superior or equivalent in performance.

6. Python Codebase Structure

The Python codebase for this Genetic Algorithm optimization project is designed with a strong emphasis on modularity, clarity, and maintainability. The project follows a well-organized directory structure, to logically separate different functionalities.

The core components are organized into the following main directories:

- *functions/*: This directory encapsulates the definitions of the benchmark optimization functions
    - *ackley_function.py*: Implements the Ackley function
    - *rastrigin_function.py*: Implements the Rastrigin function
    - *base_function.py*: contains a base class or interface for benchmark functions, promoting extensibility

- *genetic_algorithm/:* This is the heart of the GA implementation
    - *encodings/*: A sub-directory dedicated to different genetic representations
        - *binary_encoding.py*: Handles the binary representation and its specific operations
        - *real_encoding.py*: Manages the real-valued representation and its operations
        - *factory.py*: implements a factory pattern for creating encoding instances
    - *ga_core.py*: Contains the main Genetic Algorithm optimization loop and its core logic
    - *selection.py*: Implements the selection mechanism (Tournament Selection)
- *services/*: This directory provides utility functions for running experiments and analysis
    - *run_all_configurations.py*: Orchestrates the execution of all defined GA configurations across benchmark functions
    - *run_single_configuration.py*: Executes a single GA experiment with a given configuration
    - *statistical_analysis_service.py*: Handles the statistical tests and calculations
    - *visualisation_service.py*: Responsible for generating the various plots and visualizations
- *utils/*: Contains general utility scripts.
    - *config.py*: Manages project configuration parameters
    - *main.py*: The primary entry point for running the overall project.

Each module and function within this structure is documented, ensuring clarity, ease of understanding, and maintainability for future development or debugging. The modular design facilitates independent development and testing of components, contributing to the robustness of the overall system.

To implement a new function, one needs to simply create a new class inheriting the abstract base function class. The process of adding a new representation is similar having to create a new class for that representation and include functions for the same steps as in the others: initialization, encoding/decoding, mutation and crossover. For new crossover types, creating a new crossover method and mapping it to a name in the constructor of the representation is enough to use this

representation in the code. For this new configuration to be also tested in the experiments, the services could also need some extensions (e.g. adding the configuration in the run_all_configurations.py) to support it depending on the user's needs.

7. Conclusions

This project successfully explored the application of Genetic Algorithms to optimize two multimodal benchmark functions, Ackley and Rastrigin, comparing binary and real-valued encoding schemes with various crossover operators. Through an iterative process of experimentation and parameter tuning, significant insights into GA performance and the impact of configuration choices were gained.

Initially, binary encodings struggled, particularly on the Ackley function when restricted to a small domain. However, by expanding the search space to the standard domain and carefully adjusting the mutation rate, binary encoding (especially with 2-point crossover) demonstrated remarkable effectiveness and consistency for both Ackley and Rastrigin.

Real-valued encoding, particularly with the BLX-α crossover, proved to be exceptionally robust for the Ackley function, consistently converging to the global optimum with minimal variance. For the Rastrigin function, Real + BLX also performed strongly, consistently finding the optimum, and statistically outperforming Real + Arithmetic.

The mutation rate emerged as a critical parameter, demonstrating a delicate balance between exploration and exploitation. A too-low rate could lead to premature convergence or getting trapped in local optima, while a balanced rate was crucial for achieving high consistency across most configurations. Increasing the number of generations from 100 to 150 also significantly contributed to improved consistency, particularly for the Rastrigin function.

The statistical analysis using Kruskal-Wallis and Dunn's tests provided confirmation of the visual observations. For Ackley, Real + BLX was statistically superior to all other configurations. For Rastrigin, both binary encodings and Real + BLX were statistically significantly better than Real + Arithmetic, and there was no statistically significant difference between the binary methods and Real + BLX. This underscores the competitive performance of the well-tuned binary GAs against real-valued methods for these specific problems.

In conclusion, this project successfully demonstrated the power and adaptability of Genetic Algorithms for function optimization. It highlighted the critical importance of appropriate problem representation (function domains), the choice of encoding and crossover operators, and meticulous parameter tuning to achieve optimal and consistent performance. The findings

provide a strong foundation for further research into GA applications and comparisons with other approaches.

8. Bibliography

The following Python libraries were used in this project:

- numpy: https://numpy.org/
- pandas: https://pandas.pydata.org/
- tqdm: https://tqdm.github.io/
- scipy: https://scipy.org/
- statsmodels: https://www.statsmodels.org/stable/index.html
- scikit_posthocs: https://pypi.org/project/scikit-posthocs/
- seaborn: https://seaborn.pydata.org/
- matplotlib: https://matplotlib.org/

The optimized functions were taken from: https://www.sfu.ca/~ssurjano/optimization.html