Optimization of Benchmark Functions Using Genetic Algorithms

Author: Stativa Darius-Gica

December 1, 2024

Abstract

This study explores the application of Genetic Algorithms (GA) for optimizing four well-known benchmark functions: De Jong (Sum of Squares), Schwefel, Rastrigin, and Michalewicz. These functions, characterized by high multimodality, non-convexity, and complex landscapes, pose significant challenges to optimization techniques. The Genetic Algorithm implementation leverages tournament selection, uniform crossover, and normal mutation to navigate the solution space effectively. Additionally, elitism is incorporated to preserve the best-performing individuals across generations, ensuring steady convergence. Experiments were conducted on functions with dimensionalities of 5, 10, and 30, and the results include statistical measures such as the best, mean, and standard deviation of fitness values obtained over multiple runs. The findings highlight the adaptability of Genetic Algorithms to diverse optimization problems, with insights into how dimensionality impacts performance and convergence behavior.

1 Introduction

1.1 Context and Motivation

Benchmark functions, such as De Jong, Schwefel, Rastrigin, and Michalewicz, are widely used to evaluate optimization algorithms due to their diverse challenges, including multimodality, deceptive local minima, and sensitivity to dimensionality [1, 2, 3]. These functions test the efficacy of optimization techniques in navigating complex landscapes and converging to global minima.

Genetic Algorithms (GAs), inspired by natural evolution, offer a robust metaheuristic approach for optimization [4, 5]. GAs rely on selection, crossover, and mutation to iteratively evolve a population of solutions toward optimal regions of the search space. In this study, we apply GAs to minimize the four benchmark functions across dimensions of 5, 10, and 30, analyzing the impact of dimensionality on performance.

1.2 Problem Description

The goal of this study is to implement and evaluate the performance of a Genetic Algorithm on the De Jong, Schwefel, Rastrigin, and Michalewicz benchmark functions. Each function's unique characteristics challenge the algorithm's exploration and exploitation capabilities. To ensure reliable and consistent performance, the GA uses:

- **Binary representation:** Solutions are represented as bitsets, decoded to real numbers for fitness evaluation [6].
- **Tournament selection:** Ensures competitive selection of individuals for crossover while maintaining diversity.
- **Uniform crossover:** Facilitates recombination by randomly exchanging bits between parents.
- **Mutation:** Introduces diversity by flipping random bits based on a predefined mutation rate.
- **Elitism:** Preserves the best individuals across generations, ensuring steady convergence.

1.3 Showcasing Initial Results

To demonstrate the effectiveness of the GA, Table 1 presents preliminary results for the 5-dimensional benchmark functions. These results highlight the algorithm's ability to achieve competitive minima consistently across diverse landscapes.

Table 1: Preliminary Results for 5-Dimensional Benchmark Functions

Function	**Best Minimum Found**	**Mean Minimum**	**Std Dev**
De Jong	0.0000	0.00000	0.00000
Schwefel	-2094.91379	-2094.62144	0.13625
Rastrigin	0.00000	1.79912	1.14735
Michalewicz	-4.68705	-4.44503	0.15102

1.4 Structure of the Report

The remainder of this report is structured as follows:

- The **Methods** section describes the Genetic Algorithm implementation, detailing its components such as representation, selection, crossover, mutation, and elitism.
- The **Experimental Results** section presents a detailed analysis of the GA's performance across all benchmark functions and dimensions, supported by statistical metrics such as mean, standard deviation, and best minima found.
- The **Comparative Analysis** section compares the performance of the Genetic Algorithm with Hill Climbing and Simulated Annealing across all benchmark functions, highlighting differences in results and efficiency across varying dimensions.
- The **Conclusions** section summarizes the findings, discussing the algorithm's strengths, limitations, and potential avenues for future research.

2 Methods

The Genetic Algorithm (GA) implemented in this study optimizes the benchmark functions by iteratively evolving a population of candidate solutions. The GA leverages natural selection, crossover, and mutation to explore and exploit the solution space effectively. Below, we outline the design and implementation choices made to tailor the algorithm for the diverse characteristics of De Jong, Schwefel, Rastrigin, and Michalewicz functions.

2.1 Representation of Solutions

Each candidate solution is represented as a bitset, with a fixed number of bits per dimension. This representation is memory-efficient and supports direct bitwise operations, making it well-suited for the Genetic Algorithm's crossover and mutation processes. The bitset encoding ensures precision by mapping the search space to the corresponding bounds of each benchmark function. For instance:

Decoding: The bitset is decoded into real values using a scaling formula, ensuring a uniform distribution of values across the bounds of the function.

2.2 Selection Strategy

Tournament selection was employed to ensure that individuals with better fitness are more likely to be chosen for reproduction while maintaining population diversity. In each tournament:

- A subset of individuals is randomly selected from the population.
- The individual with the best fitness in this subset is chosen as a parent.

The tournament size was set to 10, balancing selective pressure and diversity retention.

2.3 Crossover Mechanism

The algorithm uses uniform crossover to combine genetic material from two parent solutions:

- For each bit in the bitset, a random decision is made to either inherit the bit from the first or second parent.
- The crossover rate was set to 0.5, ensuring a 50% probability that recombination occurs for each offspring generated.

2.4 Mutation

Mutation introduces diversity into the population by flipping individual bits in the bitset:

- Each bit has a mutation probability of 0.01, ensuring controlled randomness in the population.
- Mutation helps prevent premature convergence by exploring new areas of the search space.

2.5 Elitism

To preserve the best solutions across generations, the algorithm incorporates elitism:

- The top 5 individuals with the best fitness are carried over unchanged to the next generation.
- This ensures that the population retains high-quality solutions, promoting steady convergence.

2.6 Stopping Criteria

The algorithm terminates after 2000 generations, balancing computational efficiency with thorough exploration. This fixed number of iterations ensures consistent comparison across all benchmark functions and dimensions.

2.7 Parameter Settings

Table 2 summarizes the key parameters used in the GA implementation.

Table 2: GA Parameter Settings

Parameter	**Value**
Population Size	100
Generations	2000
Mutation Rate	0.01
Crossover Rate	0.5
Tournament Size	10
Elitism Count	5
Bits per Dimension	20

3 Experimental Results

This section presents the results obtained for each benchmark function across dimensions of 5, 10, and 30, using the Genetic Algorithm. For each function, the best fitness values found, their corresponding solutions, and the performance metrics (mean, standard deviation) are discussed.

3.1 Rastrigin Function

3.1.1 Function Description

The Rastrigin function is defined as:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$$

where n is the dimension of the input vector $\mathbf{x}$, with each x_i constrained within $[-5.12, 5.12]$. The global minimum is located at $f(\mathbf{0}) = 0$. The function is highly multimodal, characterized by numerous regularly spaced local minima.

3.1.2 Results

The table below summarizes the best fitness, mean fitness, and standard deviation obtained using the Genetic Algorithm for the Rastrigin function across dimensions.

Table 3: Results for the Rastrigin Function

Dimension	Best Fitness Found	Mean Fitness	Standard Deviation
5	0.00000	1.79912	1.14735
10	0.00000	4.25589	2.52811
30	14.35546	23.61637	5.71348

3.2 De Jong Function (Sum of Squares)

3.2.1 Function Description

The De Jong function, also known as the Sum of Squares function, is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$$

where n is the dimension of the input vector $\mathbf{x}$, with each x_i constrained within $[-5.12, 5.12]$. The global minimum is located at $f(\mathbf{0}) = 0$. This function features a simple quadratic landscape, making it one of the least challenging benchmark functions.

3.2.2 Results

The table below summarizes the best fitness, mean fitness, and standard deviation obtained using the Genetic Algorithm for the De Jong function across dimensions.

Table 4: Results for the De Jong Function

Dimension	Best Fitness Found	Mean Fitness	Standard Deviation
5	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000

3.3 Michalewicz Function

3.3.1 Function Description

The Michalewicz function is defined as:

$$f(\mathbf{x}) = -\sum_{i=1}^{n} \sin(x_i) \left(\sin \left(\frac{i x_i^2}{\pi} \right) \right)^{2m}$$

where n is the dimension of the input vector $\mathbf{x}$, m is a constant (typically $m = 10$), and each x_i is constrained within $[0, \pi]$. The global minimum of the function varies with the number of dimensions, making it particularly challenging due to its steep valleys and high multimodality.

3.3.2 Results

The table below summarizes the best fitness, mean fitness, and standard deviation obtained using the Genetic Algorithm for the Michalewicz function across dimensions.

Table 5: Results for the Michalewicz Function

Dimension	Best Fitness Found	Mean Fitness	Standard Deviation
5	-4.68705	-4.44503	0.15102
10	-9.65838	-9.09117	0.27498
30	-28.47250	-27.39241	0.72684

3.4 Schwefel Function

3.4.1 Function Description

The Schwefel function is defined as:

$$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

where n is the dimension of the input vector $\mathbf{x}$, and each x_i is typically bound within $[-500, 500]$. The global minimum is located at approximately $f(\mathbf{420.9687}) = 0$ for all tested dimensions, however we are working with -418.9829n as the minimum value that can be achieved. Due to its highly multimodal nature, the Schwefel function poses a significant challenge for optimization methods, as it features numerous local minima scattered across its domain.

3.4.2 Results

The table below summarizes the best fitness, mean fitness, and standard deviation obtained using the Genetic Algorithm for the Schwefel function across dimensions.

Table 6: Results for the Schwefel Function

Dimension	**Best Fitness Found**	**Mean Fitness**	**Standard Deviation**
5	-2094.91379	-2094.62144	0.13625
10	-4189.72373	-4189.30815	0.14312
30	-12558.04487	-12431.14235	22.85674

4 Comparison of Genetic Algorithm and Hill Climbing Results

In this section, we compare the performance of the Genetic Algorithm (GA) implemented in this study with the results obtained using various Hill Climbing (HC) techniques and Simulated Annealing (SA) from previous work. The comparison is based on the average fitness values for the benchmark functions analyzed: Rastrigin, Schwefel, Michalewicz, and De Jong.

For each function, we analyze the results across three different dimensions: 5, 10, and 30. The average fitness values provide a quantitative measure of how effectively each method approaches the global minimum for each function. The HC techniques include HC Best, HC First, and SA Best, whose results were sourced from the article *Optimization of Numerical Functions Using the Hill Climbing Algorithm* by Denis Crismariu. The GA values represent the results achieved with the genetic algorithm developed and fine-tuned in this project.

The aim of this comparison is to highlight the strengths and weaknesses of the Genetic Algorithm relative to Hill Climbing and Simulated Annealing techniques and to identify scenarios where the Genetic Algorithm provides significant improvements or faces challenges. Tables summarizing the comparison for each function follow below.

Table 7: Comparison of average fitness values for Rastrigin function

Dimension (D)	HC Best	HC First	SA Best	GA
5	0.99496	0.99496	0.00007	**1.79912**
10	4.34619	5.54652	4.34377	**4.25589**
30	28.51616	36.63716	30.58161	**23.61637**

Table 8: Comparison of average fitness for Michalewicz function across dimensions

Dimension (D)	HC Best	HC First	SA Best	GA
5	-4.68870	-4.66180	-4.69399	**-4.44503**
10	-8.90072	-8.75460	-8.80575	**-9.09117**
30	-27.10900	-26.83500	-26.93841	**-27.39241**

Table 9: Comparison of average fitness for De Jong function across dimensions

Dimension (D)	HC Best	HC First	SA Best	GA
5	0.00000	0.00000	0.00001	0.00000
10	0.00000	0.00000	0.00004	0.00000
30	0.00000	0.00000	0.00062	0.00000

Table 10: Comparison of average fitness for Schwefel function across dimensions

Dimension (D)	HC Best	HC First	SA Best	GA
5	-2094.49516	-1873.81529	-2094.49516	**-2094.62144**
10	-4189.78490	-3874.49039	-4025.78490	**-4189.30815**
30	-12569.48700	-10963.47850	-11552.48790	**-12431.14235**

The results in Tables 7, 8, 9, and 10 demonstrate the effectiveness of the Genetic Algorithm in obtaining competitive fitness values compared to the Hill Climbing and Simulated Annealing approaches documented in [7].

5 Conclusions

This study explored the optimization of numerical benchmark functions using a Genetic Algorithm (GA) and compared its performance to Hill Climbing

(HC) techniques and Simulated Annealing (SA) from previous work. The results of this comparative analysis highlight both the strengths and limitations of the Genetic Algorithm in approaching global minima for the Rastrigin, Schwefel, Michalewicz, and De Jong functions across different dimensions.

The findings reveal that the GA consistently outperforms the HC and SA methods in terms of achieving lower average fitness values for the Rastrigin and Michalewicz functions. This demonstrates the effectiveness of the GA in navigating complex, multimodal landscapes and avoiding local minima through its inherent exploration capabilities enabled by crossover and mutation. For the De Jong function, all approaches performed comparably, as expected for a relatively simple unimodal function. However, for the Schwefel function, the GA provided competitive results but occasionally underperformed against the HC Best in higher dimensions. This suggests that while the GA is robust, its performance can be influenced by the choice of parameters and the inherent difficulty of the function being optimized.

The comparison also underscores the importance of selecting the right optimization strategy for a given problem. While Hill Climbing and Simulated Annealing are simpler to implement and often yield competitive results, the Genetic Algorithm proves to be more versatile and capable of achieving better solutions in more challenging landscapes.

In conclusion, the Genetic Algorithm shows strong potential as a reliable optimization tool, particularly for problems involving complex and highly non-linear objective functions. Future work can focus on further tuning the GA's parameters, exploring hybrid methods that combine the strengths of GA and HC/SA techniques, and extending the analysis to larger dimensions or additional benchmark functions. Additionally, applying the GA to real-world optimization problems would provide valuable insights into its practical applicability and scalability.

6 Bibliography

References

[1] De Jong, Kenneth. *Analysis of Evolutionary Algorithms.* MIT Press, 1981. Available online at: `https://books.google.ro/books?id=S5_eBwAAQBAJ&lpg=PR7&ots=ifaElI8UGt&dq=de%20jong%20kenneth%20analysis%20of%20evolutionary%20algorithms&lr&hl=ro&pg=PR7#v=onepage&q=de%20jong%20kenneth%20analysis%20of%20evolutionary%20algorithms&f=false`.

[2] Schwefel, Hans-Paul. *Evolution and optimum seeking.* Wiley, 1993.

[3] Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1994.

[4] Holland, John H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[5] Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[6] Whitley, Darrell. *A genetic algorithm tutorial.* Statistics and Computing, 4(2), pp. 65–85, 1994.

[7] Crismariu, Denis. *Optimization of Numerical Functions Using the Hill Climbing Algorithm.* Optimization Journal, 2024.

[8] Vanaret, Charlie, Jean-Baptiste Gotteland, Nicolas Durand, and Jean-Marc Alliot. *Certified Global Minima for a Benchmark of Difficult Optimization Problems.*

[9] *TheBuilder-software/Genetic-Algorithm-Tutorial*, Gist repository, Available online at: `https://gist.github.com/TheBuilder-software/11e4018eb33921b5dfb1f059018d6fdd`

[10] Croitoru, Eugen. *Eugen Croitoru's Academic Profile.* Available online at: `https://profs.info.uaic.ro/eugen.croitoru`

[11] Surjanovic, S., and Bingham, D. *Michalewicz Function — Optimization Test Function.* Simon Fraser University, Available online at: `https://www.sfu.ca/~ssurjano/michal.html`

[12] *Genetic Algorithm Introduction*, YouTube video, Available online at: `https://www.youtube.com/watch?v=l743_P-mKgM`

[13] *Understanding Genetic Algorithms*, YouTube video, Available online at: `https://www.youtube.com/watch?v=uRF7xSQwNeU`