Improved Genetic Algorithm

Aman Sisodiya (2022A7PS1263G)

September 7, 2024

Abstract

In this report, I explore methods to enhance the performance of a genetic algorithm by varying critical parameters such as the fitness function, population size, and the number of generations. These parameters significantly influence the quality of solutions and the convergence time of the algorithm. By systematically altering these variables, I analyze how different configurations impact the algorithm's ability to find the best solution and the number of generations required to achieve this.

Furthermore, I discuss how each modification contributes to the overall behavior of the genetic algorithm, offering a deeper understanding of the relationship between parameter adjustments and algorithmic efficiency. The report concludes with recommendations for future optimization and experimentation with other variables to further improve genetic algorithm performance.

1 Introduction

In this report, I explore the use of a genetic algorithm (GA) to find optimal or near-optimal solutions to the Set Covering Problem (SCP). Genetic algorithms are powerful search heuristics based on the principles of natural selection and evolution. They have been successfully applied to a wide range of complex optimization problems due to their ability to explore large solution spaces effectively.

The objective of this study is to implement and improve the performance of a genetic algorithm for SCP by experimenting with key parameters such as the fitness function, population size, and the number of generations. I evaluate the algorithm's ability to find the best possible solution and analyze the effect of various modifications on both solution quality and computational efficiency. In particular, I assess how changes in population size and generations impact the convergence rate and final solution accuracy.

The report includes a detailed analysis of the algorithm's performance across different parameter settings, supported by graphs that visually demonstrate the relationship between parameter choices and outcomes. Through this investigation, I aim to identify the optimal configuration of the genetic algorithm for solving the Set Covering Problem efficiently.

2 Part A: Linear Fitness Function

Initially, I used a simple linear fitness function defined as the sum of two terms: the **coverage score** and the **subset count**. The goal was to minimize the number of subsets while maximizing the coverage of the universe.

The fitness function can be expressed as:

$$F_{\text{linear}} = \text{coverage\_score} + \text{subset\_count}$$

2.1 Coverage Score

The coverage score represents how well the selected subsets cover the elements of the universe. Ideally, a higher coverage score means more elements are covered, and the algorithm should be incentivized to achieve full coverage. However, this simple term alone cannot balance the complexity of selecting minimal subsets, which led to suboptimal results.

2.2 Subset Count

The subset count penalizes the fitness based on the number of subsets chosen. The fewer the subsets, the lower this penalty. This aspect of the fitness function aimed to reduce the number of subsets selected while ensuring that the universe was fully covered.

2.3 Results Over 50 Generations

Although the linear fitness function worked to some extent, the algorithm's performance was not satisfactory over 50 generations. The convergence was slow, and the solutions found were far from optimal, especially for larger population sizes. Below are the results from two runs of the algorithm:

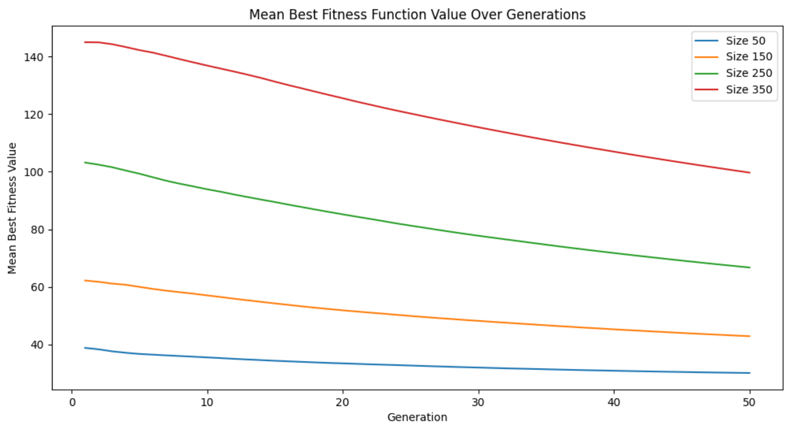

Figure 1: Mean Fitness Values Over 50 Generations Using Linear Fitness Function

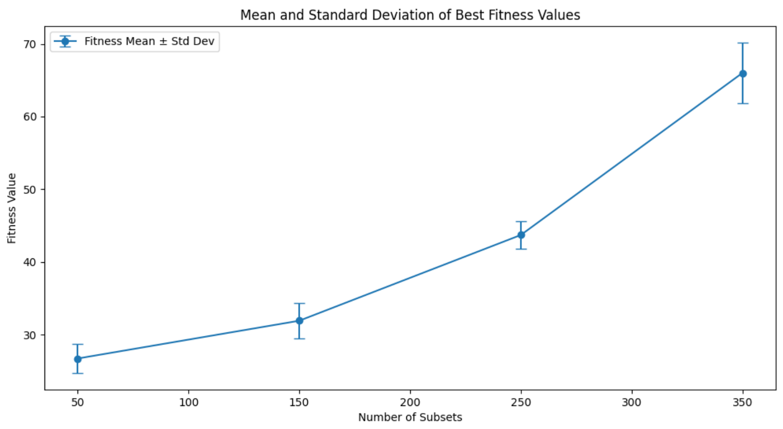

Figure 2: Fitness Value and Standard Deviation for Linear Fitness Function Over 50 Generations

As shown in the graphs above, the linear fitness function failed to produce high-quality solutions. Although some improvements were observed as the generations progressed, the algorithm did not reach a satisfactory level of convergence. These results led me to explore methods to improve the algorithm, which are discussed in the subsequent sections.

3 Part B: Effects of Varying Parameters and Alternative Fitness Functions

To address the shortcomings of the linear fitness function, I experimented with alternative fitness functions and explored the effects of varying population size and the number of generations on the genetic algorithm's performance.

3.1 Effects of Varying Generations and Population Size

It was observed that increasing both the population size and the number of generations improved the results of the genetic algorithm. As shown in the graph below, a larger population size and a higher number of generations generally resulted in better fitness values. However, even with these improvements, the algorithm did not achieve satisfactory convergence, as the solutions remained suboptimal in terms of minimizing the number of subsets while ensuring full coverage of the universe.

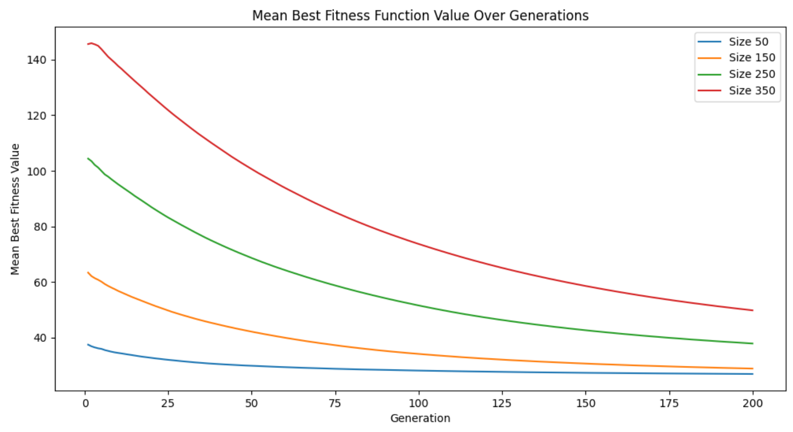

Figure 3: Mean Fitness Values Over 200 Generations for Different Population Sizes using Linear Fitness function

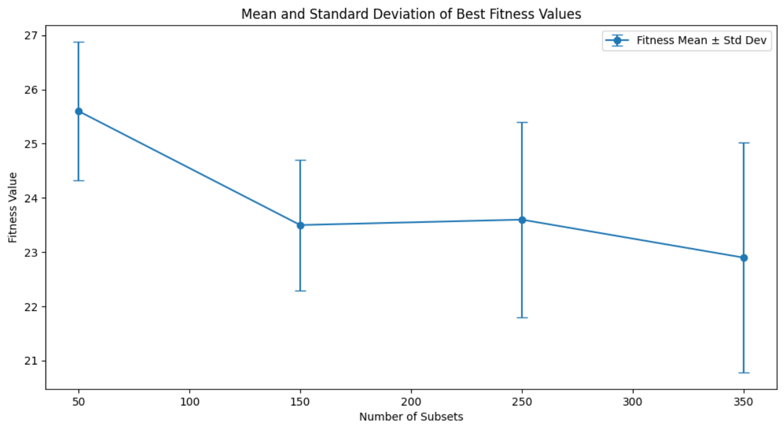

Figure 4: Mean Fitness Values and Standard Deviation Over 200 Generations for using Linear Fitness function

As can be seen in the graph, increasing the population size from 50 to 350 led to progressively better solutions, but the rate of improvement slowed down over time. Similarly, increasing the number of generations improved the algorithm's ability to explore the solution space, but the final results still left room for improvement. Therefore, I explored alternative fitness functions to further enhance the algorithm's performance.

3.2 Alternative Fitness Functions

In the initial stages of this study, a basic linear function was used as the fitness function. This linear function was designed to minimize the number of subsets selected while ensuring maximum coverage of the universe. However, the results obtained using this fitness function were suboptimal. The algorithm struggled to achieve convergence, and the solutions found were far from the best possible ones. This indicated that the linear fitness function might not be adequately guiding the genetic algorithm toward more effective solutions.

Given these unsatisfactory results, alternative fitness functions were explored to enhance the algorithm's performance. Among the various fitness functions tested, the entropic function proved to be the most effective. Below, I discuss the different fitness functions considered, along with a detailed analysis of why the entropic function emerged as the best choice.

3.2.1 Linear Function

The first fitness function applied was a simple sum, aiming to minimize the total number of subsets selected. The fitness function can be mathematically represented as:

$$F_{\text{sum}} = \sum_{i=1}^{n} c_i \cdot x_i$$

where c_i is the cost or weight of subset S_i, and x_i is a binary variable indicating whether subset S_i is selected (1 if selected, 0 otherwise).

This function assigned a fitness score based on the sum of all selected subsets, thus encouraging the algorithm to select as few subsets as possible. While this approach seemed straightforward, it often led to incomplete coverage, as the algorithm tended to neglect some elements of the universe to minimize the number of subsets.

3.2.2 Other Fitness Functions

Several other fitness functions were explored in an attempt to improve the performance of the genetic algorithm. These functions include the weighted sum of squares, the Euclidean norm, and a logarithmic function. However, despite the promising theoretical properties of these functions, none performed as well as the entropic fitness function.

Weighted Sum of Squares The weighted sum of squares fitness function penalizes the selection of larger subsets more heavily than smaller ones by squaring the size of the subsets. This approach was intended to encourage the selection of minimal subsets while ensuring coverage. The fitness function is expressed as:

$$F_{\text{wss}} = \sum_{i=1}^{n} c_i \cdot x_i^2$$

where c_i is the cost or weight of subset S_i, and x_i is a binary variable indicating whether subset S_i is selected. While this function penalized large subsets more than smaller ones, it did not consistently lead to better solutions compared to the entropic function.

Euclidean Norm The Euclidean norm-based fitness function aims to minimize the number of subsets selected by treating the selected subsets as vectors in a high-dimensional space. This function uses the Euclidean distance to measure the "length" of the selected solution:

$$F_{\text{euclid}} = \sqrt{\sum_{i=1}^{n} (c_i \cdot x_i)^2}$$

The intention was to reduce the size of the selected subsets by minimizing the overall norm of the vector representing the solution. However, the Euclidean norm fitness function was less effective in guiding the genetic algorithm towards an optimal solution, and convergence was slower compared to the entropic function.

Logarithmic Function A logarithmic fitness function was also tested, where the number of subsets selected was penalized logarithmically. This function can be expressed as:

$$F_{\text{log}} = \sum_{i=1}^{n} \log(1 + c_i \cdot x_i)$$

This fitness function was intended to reduce the impact of selecting large subsets while promoting the selection of smaller subsets. Unfortunately, the logarithmic function did not perform as well as expected, particularly in larger problem instances where the entropy-based fitness function outperformed it in both convergence speed and solution quality.

Despite these alternatives, the entropic function consistently produced better results, both in terms of convergence speed and overall solution quality, making it the best choice for this study.

3.2.3 Entropic Function

In order to address the shortcomings of the linear function, an entropic fitness function was proposed. This function incorporates an additional term that measures the entropy of the coverage distribution across subsets. By doing so, it encourages the selection of diverse subsets while still promoting full coverage.

The entropic fitness function can be represented as:

$$F_{\text{entropic}} = \alpha x \log(1 + x) + \beta y \log(1 + y)$$

where ,α and β are weighting factors that balances the contribution of the entropy term.

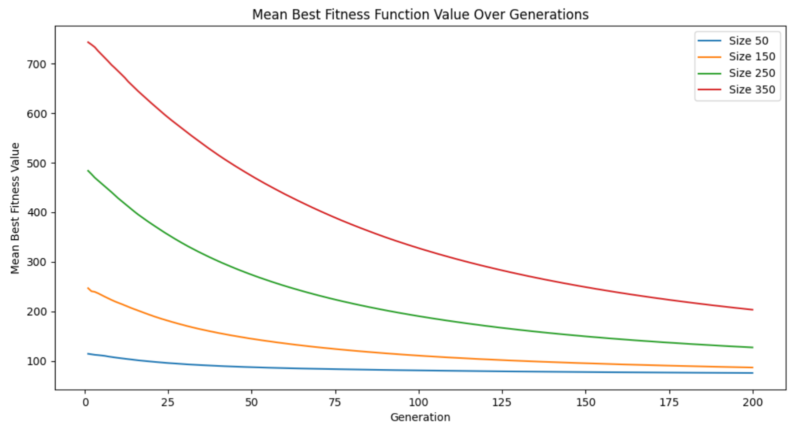

Figure 5: Mean Fitness Values Over 200 Generations Using Entropic Fitness Function

This fitness function performed significantly better in terms of both solution quality and convergence speed. The inclusion of the entropy term helped the genetic algorithm explore a wider range of potential solutions, ultimately leading to more optimal results.

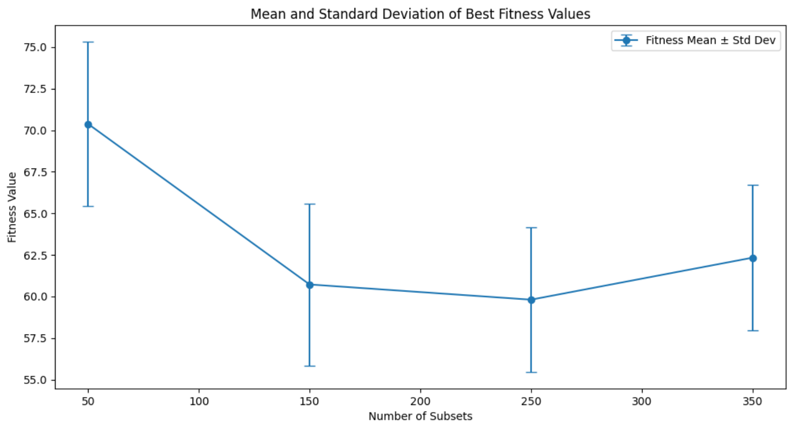

Figure 6: Mean Fitness Values and Standard Deviation Over 200 Generations Using Entropic Fitness Function

4 Analysis of Results

The genetic algorithm's performance was evaluated using different combinations of fitness functions, population sizes, and numbers of generations. The results show that the entropic fitness function consistently outperformed the linear function, especially for larger problem instances.

5 Conclusion

In this study,I solved the Set Covering Problem i.e. making the universal set using minimum number of subsets using genetic algorithm.Later, I demonstrated the importance of selecting an appropriate fitness function for a genetic algorithm. While the linear fitness function provided a basic approach to solving the SCP, it failed to guide the algorithm effectively toward optimal solutions. The entropic fitness function, by contrast, achieved superior results by encouraging diversity in the selected subsets and promoting full coverage of the universe.