





Article

Solving and Optimization of Cobb–Douglas Function by Genetic Algorithm: A Step-by-Step Implementation

Ali Dinc ^{1,*}, Faruk Yildiz ¹, Kaushik Nag ², Murat Otkur ² and Ali Mamedov ²¹ Engineering Technology, Sam Houston State University, Huntsville, TX 77340, USA; fxy001@shsu.edu² College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; kaushik.nag@aum.edu.kw (K.N.); murat.otkur@aum.edu.kw (M.O.); ali.mamedov@aum.edu.kw (A.M.)

* Correspondence: ali.dinc@shsu.edu; Tel.: +1-(936)294-1857

Abstract: This study presents an innovative application of genetic algorithms (GAs) for optimizing the Cobb–Douglas production function, a cornerstone of economic modeling that examines the relationship between production output and the inputs of labor and capital. This research integrates traditional optimization methods, such as partial derivatives, with evolutionary computation techniques to address complex economic constraints. The methodology demonstrates how GAs outperform classical techniques in solving constrained optimization problems, offering superior robustness, adaptability, and efficiency. Key results highlight the alignment between GA solutions and traditional Lagrangian methods while underscoring the computational advantages of GAs in navigating non-linear and multi-modal landscapes. This work serves as a valuable resource for both educators and practitioners, offering insights into the potential of GAs to enhance optimization processes in engineering, economics, and interdisciplinary applications. Visual aids and pedagogical recommendations further illustrate the algorithm’s utility, making this study a significant contribution to the computational optimization literature. Additionally, the optimization process using genetic algorithms is presented in a step-by-step manner, with accompanying visual graphs that enhance comprehension and demonstrate the method’s effectiveness in solving mathematical problems, as validated by the study’s results.

Keywords: genetic algorithms (GAs); optimization; mathematical economics; Cobb–Douglas production function; computational methods; evolutionary computation



Academic Editor: Garzia Fabio

Received: 17 December 2024

Revised: 14 January 2025

Accepted: 20 January 2025

Published: 23 January 2025

Citation: Dinc, A.; Yildiz, F.; Nag, K.; Otkur, M.; Mamedov, A. Solving and Optimization of Cobb–Douglas Function by Genetic Algorithm: A Step-by-Step Implementation. *Computation* **2025**, *13*, 23.

<https://doi.org/10.3390/computation13020023>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

This study explores the optimization of the Cobb–Douglas production function in mathematics using partial derivatives and genetic algorithms (GA). This research aims to explore mathematical methods and algorithms applied in economics, specifically focusing on GAs due to their recent emergence as an unconventional method.

The Cobb–Douglas function, formulated by economist Paul Douglas and mathematician Charles Cobb, represents a simple algebraic equation that relates output production to the inputs of capital and labor [1]. Labor refers to the human efforts involved in production, whereas capital includes the worth of non-human inputs such as machinery, buildings, and raw materials. This concept is crucial in economics for examining the relationship between input factors and the resulting output. A common form of the Cobb–Douglas production function is expressed as follows:

$$Z = AX^{\alpha}Y^{\beta} \quad (1)$$

In the above equation, Z symbolizes the output generated, A denotes total factor productivity, X corresponds to the quantity of capital used in production, and Y represents the amount of labor. α and β are the elasticity coefficients of labor and capital, respectively, indicating the proportional contribution of labor and capital to the output.

The motivation behind this study stems from the inherent complexity of optimizing the Cobb–Douglas production function, particularly under constraints like budget or resource allocation. Traditional methods, such as Lagrange multipliers or gradient-based optimization, rely on differentiability and linearity assumptions, which may not always hold in real-world applications involving non-linear constraints and complex solution spaces [2]. These methods can also struggle with high-dimensional or multi-modal problems, leading to suboptimal results.

Alternative optimization methods for solving the Cobb–Douglas function may include GAs, gradient-based optimization, particle swarm optimization, differential evolution, etc. Genetic algorithms (GAs) were selected for this study due to their ability to handle non-linear, multi-modal, and constrained optimization problems effectively. Unlike traditional gradient-based methods, GAs do not require the objective function to be differentiable or continuous, making them highly adaptable for solving complex problems like the Cobb–Douglas function. The primary advantage of GAs lies in their balance between exploration and exploitation, which reduces the risk of getting trapped in local optima [3] and ensures a more thorough search of the solution space. This makes GAs particularly suitable for optimizing economic models with non-linear constraints, as demonstrated in this study. Comparative studies have shown GAs outperforming other optimization techniques in scenarios involving non-linear constraints and high-dimensionality [4].

GAs, conceived at the University of Michigan in the 1960s and 1970s by John Holland [5] and his team, stem from the notion that evolutionary processes could be harnessed as optimization techniques for engineering challenges [6]. These algorithms have evolved thanks to contributions from computer scientists in the 1950s and 1960s [3]. The advancements in the computational capabilities and common accessibility of computers have facilitated the evolution of intelligent computing methods, with GAs emerging prominently among them [7]. These algorithms are highly effective at solving mathematical equality problems and optimization tasks [8], making them powerful tools for addressing complex problems when combined with computer coding.

In recent studies, Thompson et al. [9] examined the foundational principles of GAs, detailing their key components, such as initial population generation, parent selection, crossover and mutation operators, and population updating mechanisms. Machacha et al. [10] introduced an innovative approach that integrates GAs, case-based reasoning, and the k -nearest neighbor classifier, significantly enhancing classification performance. Hasan Kazmi et al. [11] developed optimization techniques utilizing GAs to select process parameters for near-net shape deposition, aiming to minimize defects. Anupama et al. [12] applied GAs to optimize costs in an M/M/3 queuing system with a queue-dependent multi-server setup. Babu et al. [13] proposed a comprehensive framework for improving Network Intrusion Detection Systems (NIDSs) by combining the k -nearest neighbors (KNN) algorithm, Karhunen–Loeve Transform (KLT), and genetic algorithm optimization. Jiao et al. [14] presented a lightweight quantum-inspired genetic algorithm (LQIGA) designed to address workforce scheduling challenges in supply chain and logistics operations, particularly for outsourced workforce management. Wang et al. [15] proposed a reinforcement learning-based ranking teaching-learning-based optimization algorithm for accurately identifying photovoltaic model parameters, demonstrating superior performance in accuracy, convergence speed, and complexity across five different photovoltaic models compared to eleven established algorithms. GAs [16,17] have also

been applied in numerous fields, including science [18–20], engineering [21–23], business, and finance [24–27].

The novelty of this paper lies in its step-by-step implementation of the genetic algorithm, which not only serves as an effective pedagogical tool for enhancing comprehension among new learners but also demonstrates superior performance in addressing constrained optimization problems, surpassing the limitations of traditional methods.

The organization of the paper is as follows: investigating the concept, defining the problem, solving it using partial derivatives, explaining the solution steps of the genetic algorithm method, applying the genetic algorithm to solve the problem, presenting and discussing the results, and ending with a conclusion.

2. Methodology

2.1. Problem Definition

To identify and construct the problem, the following scenario was considered. An investor aims to establish a workshop for industrial equipment. A study reveals that the number of units of machinery equipment produced can be modeled by Equation (1), where A is 150, and β is equal to 0.7 as given in Equation (2). This relationship indicates that the production output increases with additional labor and capital inputs. However, the investor faces a budget constraint of \$150,000. With labor costing \$180 per unit and capital costing \$220 per unit, the objective is to find the precise number of labor and capital units required to maximize production within the \$150,000 budget in Equation (3).

The problem can be formalized as a Cobb–Douglas production function:

$$z = f(x, y) = 150x^{0.7}y^{0.3} \tag{2}$$

where z symbolizes the output generated, x represents the amount of labor, and y corresponds to the quantity of capital used in production. Given the costs of labor and capital, the budget constraint can be expressed as:

$$180x + 220y = 150,000 \tag{3}$$

2.2. Classical Lagrange Solution

The task is to determine the values of x and y that maximize production, constrained by the budget equation. The optimization conundrum can be tackled by employing the Lagrange multipliers technique [28]. The Lagrangian function can be defined as:

$$f(x, y, \lambda) = 150x^{0.7}y^{0.3} + \lambda(180x + 220y - 150,000) \tag{4}$$

Taking partial derivatives of the Lagrangian and setting them to zero results in:

$$\frac{df}{dx} = 105x^{-0.3}y^{0.3} + 180\lambda = 0 \tag{5}$$

$$\frac{df}{dy} = 45x^{0.7}y^{-0.7} + 220\lambda = 0 \tag{6}$$

$$\frac{df}{d\lambda} = 180x + 220y - 150,000 = 0 \tag{7}$$

Combining Equations (5) and (6) to eliminate λ gives:

$$\lambda = \frac{-105x^{-0.3}y^{0.3}}{180} = \frac{-45x^{0.7}y^{-0.7}}{220} \tag{8}$$

$$x = 2.85y \quad (9)$$

Substituting $x = 2.85y$ into Equation (7) results in:

$$180(2.85y) + 220y - 150,000 = 0 \Rightarrow y = 204.55 \quad (10)$$

Substituting $y = 204.55$ into Equation (9) gives:

$$x = 2.85y \Rightarrow x = 583.33 \quad (11)$$

Therefore, the results of Equations (10) and (11) show that the Cobb–Douglas production function z reaches its maximum when $x = 583.33$ and $y = 204.55$. The maximum production output is found by placing the values found for x and y into Equation (2):

$$z = f(583.33, 204.55) = 150(583.33)^{0.7}(204.55)^{0.3} = 63895 \quad (12)$$

This analysis demonstrates the optimal allocation of labor and capital resources to achieve the maximum production output within a specified budget using the given function and the method of Lagrange multipliers.

2.3. Solution by Genetic Algorithm Implementation

Within the scope of this study, the genetic algorithm method was selected for the solution to this optimization problem. While gradient-based methods and simulated annealing have their merits, GAs offer a unique combination of flexibility, adaptability, and robustness that makes them the preferred choice for optimizing the Cobb–Douglas function, especially in challenging scenarios with non-linear constraints and complex solution spaces. These points have been elaborated in the revised manuscript to strengthen the discussion and provide a comprehensive perspective. GAs offer several advantages that make them particularly suitable for optimizing the Cobb–Douglas function in this context:

Flexibility: Unlike gradient-based methods, GAs do not require differentiability or continuity in the objective function, making them robust for handling non-linear, multi-modal, and constrained optimization problems.

Global optimization: GAs excel at exploring the solution space thoroughly, reducing the risk of being trapped in local optima.

Adaptability: The evolutionary nature of GAs allows them to adapt to diverse problem structures, including highly complex and irregular optimization landscapes.

Parameter tuning: While GAs require parameter tuning (e.g., population size, mutation rate), their performance is less sensitive to initial conditions compared to gradient-based methods.

Scalability: GAs can efficiently handle high-dimensional problems and complex constraints, which are common in real-world economic and engineering applications of the Cobb–Douglas function.

The principles, procedures, and terminology employed within the genetic algorithm (GA) draw inspiration from the theory of evolution, yet they primarily entail mathematical operations. Concepts like chromosomes, generations, mutation, and crossover denote mathematical manipulations conducted on binary numerical representations. A series of steps derived from Riechmann [29] and Kramer [4] is followed to address a well-defined problem for the application of the genetic algorithm.

Step 1: Parameter Initialization

For the solution to the problem, the following parameters were selected as suggested by Goldberg [30]:

- Initial population (number of chromosomes): 300
- Rate of crossover: 0.80 (80%)
- Rate of mutation: 0.005 (0.5%)
- Iterations (number of generations): 100

Step 2: Generation of Initial Population

Initial chromosomes (initial population) were generated using random values. The parameters were assumed as follows:

- The range for labor (x) under investigation: 0 to 1000
- The range for capital (y) under investigation: 0 to 1000
- Bit count for labor (x): 7
- Bit count for capital (y): 7
- Precision for labor (x): $1000/2^7 = 7.81$
- Precision for capital (y): $1000/2^7 = 7.81$

Initial random values for labor and capital in 7-bit intervals (0–128) were assumed to start the calculations. These values were then converted to real intervals (0–1000) using linear scaling:

$$Number_{1000} = \frac{1000 \text{ Number}_{128}}{(128 - 1)} \tag{13}$$

For instance:

$$17 \text{ in } (0 - 128) \text{ range} = \frac{1000 \times 17}{(128-1)} = 134 \text{ in } (0 - 1000) \text{ range}$$

$$43 \text{ in } (0 - 128) \text{ range} = \frac{1000 \times 43}{(128-1)} = 339 \text{ in } (0 - 1000) \text{ range}$$

Therefore, as illustrated in Table 1, ‘17’ becomes ‘134’, and ‘43’ becomes ‘339’ in the new intervals (ranges) in Table 1:

Table 1. Conversion of numbers from 7-bit (0–128) interval to real (0–1000) interval.

No. of Chrom.	Labor in 7-Bit (0–128) Interval	Labor Values in Real (0–1000) Interval	Capital in 7-Bit (0–128) Interval	Capital Values in Real (0–1000) Interval
1	17	→ 134	43	→ 339
2	86	677	125	984
3	54	425	77	606
...				
15	75	591	88	693

Step 3: Evaluation of Fitness

The fitness of each chromosome was assessed by calculating the objective function to identify the maximum value. Furthermore, the constraint equation was computed to ensure that the values did not surpass 150,000:

Objective function = Cobb–Douglas production function given in Equation (2)

Constraint Equation = Budget constraint given in Equation (3)

For example:

$$z_1 = f(134, 339) = 150(134)^{0.7}(339)^{0.3} = 26,553$$

$$z_2 = f(677, 989) = 150(677)^{0.7}(989)^{0.3} = 113,779$$

$$z_7 = f(73, 91) = 150(173)^{0.7}(291)^{0.3} = 30,331$$

Next, the constraint checks need to be done for these chromosomes number 1, 2, 7 and so on in Table 2:

$$\text{Constraint Equation}_1: 180x + 220y = 180(134) + 220(339) = 98,700 \leq 150,000$$

Constraint Equation₂: $180x + 220y = 180(677) + 220(984) = 338,340 > 150,000$

Constraint Equation₇: $180x + 220y = 180(173) + 220(291) = 77,860 \leq 150,000$

In Table 2, the objective function (z) needs to be maximized. Simultaneously, the constraint equation must be monitored. If the value of the constraint equation exceeds 150,000, the objective function value is set to zero to eliminate the chromosome.

Table 2. Calculation of objective function and application of constraint.

No of Chrom.	Labor Values (x) in (0–128) Interval	Labor Values (x) in Real (0–1000)	Capital Values (y) in (0–128) Interval	Capital Values (y) in Real (0–1000)	Objective Function (z)	Constraint Equation (If $\leq 150,000$)	Objective Function After Constraint (z)
1	17	134	43	339	26,553	98,700	26,553
2	86	677	125	984	113,779	338,340	0
7	22	173	37	291	30,331	77,860	30,331
10	69	543	92	724	88,792	202,720	0
11	15	118	35	276	22,839	70,160	22,839
15	75	591	88	693	92,987	199,740	0

Step 4: Selection and Duplication

The process of choosing the most optimal chromosomes relied on the preceding phase, employing fitness criteria derived from the objective function. In the scenario, a higher objective function value indicates a better chromosome for finding the maximum value. After applying the constraint, chromosomes with an objective function value set to zero were eliminated. The remaining chromosomes after this selection process are listed below. To replenish the removed chromosomes, the chosen ones were duplicated, ensuring the total count returned to 15. As shown in Table 3, chromosome 1 has the highest duplication number of 9, due to its objective function value being the greatest. This implies that chromosome 1, identified as the most optimal individual, will be replicated and employed nine times. In contrast, chromosome 11, ranked the lowest in the objective function, will be duplicated twice, the lowest frequency compared to the other chromosomes.

Table 3. Selection and duplication of best chromosomes.

No. of Chromosomes	Labor Values (x) in (0–128)	Capital Values (y) in (0–128)	Objective Function	Constraint (If $\leq 150,000$)	Duplication of Best Chromosomes
1	17	43	26,553	98,700	9
7	22	37	30,331	77,860	4
11	15	35	22,839	70,160	2
total					15

Following the selection and replication process, the complete list of chromosomes is presented in Table 4. Chromosomes numbered 1, 7, and 11 need to be converted to binary format. The first chromosome consists of a labor value of 17 and a capital value of 43. The next step involves converting these labor and capital values into binary format. Converting between decimal (base 10) and binary (base 2) numbers involves a standard algorithm.

Table 4. Binary format conversion of best chromosomes.

No of Chromosomes	Labor Value Chromosomes	Capital Value Chromosomes
Number 1 chromosome (9 times duplication)	0 0 1 0 0 0 1	0 1 0 1 0 1 1
Number 7 chromosome (4 times duplication)	0 0 1 0 1 1 0 0	1 0 0 1 0 1
Number 11 chromosome (2 times duplication)	0 0 0 1 1 1 1 0	1 0 0 0 1 1
	17 in base 10	43 in base 10

Step 5: Crossover

The crossover process involves selecting pairs of chromosomes and exchanging segments to create new chromosomes. With a crossover rate of 0.80, 12 out of 15 chromosomes underwent crossover. The randomly selected pairs for crossover were (1 and 7), (11 and 10), (5 and 14), (13 and 15), (12 and 3), and (4 and 8), totaling 12 chromosomes or 6 pairs. The crossover sites for each pair were chosen randomly as follows: 3, 7, 3, 11, 12, and 6, corresponding to each respective pair.

Below, Figure 1 illustrates the crossover at the 3rd digit (crossover site) between the 1st and 7th chromosomes. As an example, the 1st and 7th chromosomes were selected, and the crossover occurred at the 3rd digit of a 14-digit chromosome. As shown in Figure 1, the crossover involved splitting each 14-digit (bit) chromosome at the 3rd digit and swapping the resulting segments to create two new chromosomes.

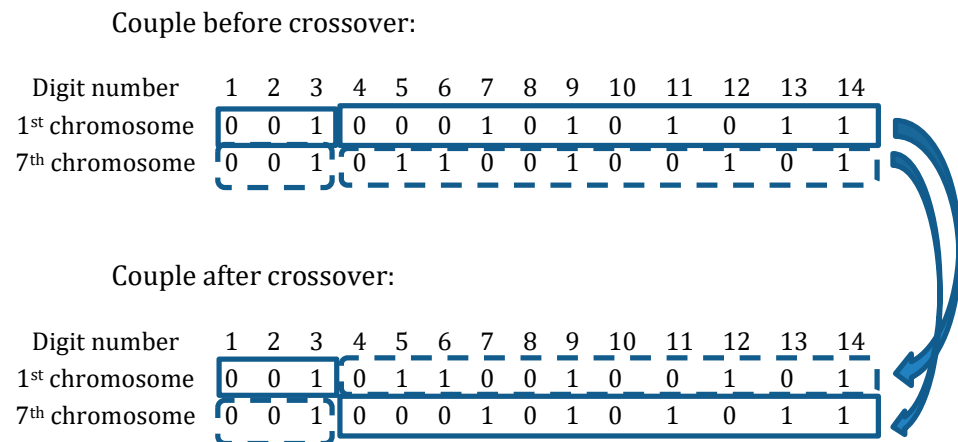


Figure 1. Crossover process of two chromosomes.

After conducting the crossover on each randomly selected chromosome, the new population of chromosomes is displayed in Table 5:

Table 5. New chromosomes after crossover.

No of Chromosomes	Labor Value Chromosomes	Capital Value Chromosomes
1	0 0 1 0 1 1 0 0	1 0 0 1 0 1
2	0 0 1 0 0 0 1 0	1 0 1 0 1 1
...		
15	0 0 1 1 0 0 0 0	1 0 0 1 1 1

Step 6: Mutation

Mutation involves altering bits of the chromosomes from 1 to 0 or vice versa. In this case, the calculation requires multiplying the mutation rate, the number of chromosomes, and the total length of chromosomes to find the number of digits to be replaced. Therefore, $0.005 \times 15 \times 14 = 1.05$, resulting in approximately one digit being replaced. The location of this digit was randomly assumed as the 7th chromosome and the 4th bit, as seen in Table 6.

Table 6. Mutation process.

No. of Chromosomes	Labor Value Chromosomes							Capital Value Chromosomes								
7th old	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1
7th new	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1

Step 7: Generation of New Offspring

New chromosomes (offspring) are determined after the crossover and mutation processes. The outcomes of steps 5 and 6 are combined to find the new generation of chromosomes in this step, as shown in Table 7.

Table 7. New chromosomes after crossover and mutation.

No. of Chromo.	New Labor Value Chromosomes							New Capital Value Chromosomes							New Labor Values (Real)	New Capital Values (Real)
1	0	0	1	0	1	1	0	0	1	0	0	1	0	1	22	37
2	0	0	1	0	0	0	1	0	1	0	1	0	1	1	17	43
3	0	0	0	1	1	0	1	0	0	1	1	1	0	1	13	29
...																
15	0	0	1	1	0	0	0	0	1	0	0	1	1	1	24	39

Step 8: Repetitive Procedure

The above steps (4–7) are reiterated for many generations until convergence is achieved, resulting in the best chromosomes that maximize the objective function.

3. Results and Discussion

In this section, computer-generated plots are presented initially to illustrate the Cobb–Douglas function within a specified interval, enabling visual observation of maximum values. Subsequently, leveraging the Lagrangian solution and visual representations previously introduced, an alternative solution achieved through a Genetic Algorithm (GA) is presented. A specialized computer code tailored specifically to executing the GA solution, integrating the procedural steps previously delineated, was meticulously developed. The discussion elaborates on the outcomes derived from this implementation.

In economics, the Cobb–Douglas function is widely used to model production processes, such as determining the optimal allocation of resources like labor and capital to maximize output. Our work demonstrates how GAs can efficiently solve such optimization problems, even under complex constraints like budget limits or production quotas, which traditional gradient-based methods may struggle to handle. For example, resource allocation in manufacturing or agricultural sectors often involves multiple non-linear constraints, where our method offers a robust and flexible alternative.

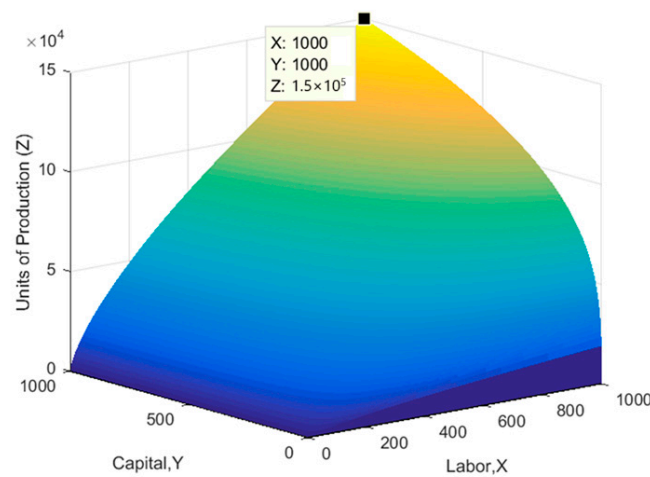
In engineering, the Cobb–Douglas function is applicable in systems optimization, such as balancing resource usage and output in energy generation or industrial production. Our

approach addresses scenarios where traditional methods may fail to converge or require significant computational resources due to the non-linearity and multi-modal nature of the problem space.

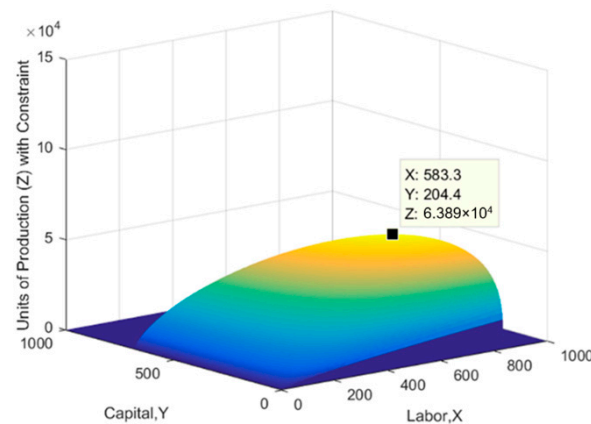
Moreover, we highlight gaps in existing methodologies. Traditional optimization techniques, such as Lagrange multipliers or gradient-based methods, are often limited by their reliance on linearity and smoothness assumptions. They may also require precise analytical formulations, which are impractical in real-world scenarios with noisy or incomplete data. Our work bridges these gaps by employing a genetic algorithm, which is inherently flexible and can handle non-linear, multi-modal, and constrained optimization problems effectively.

3.1. Graphical Representation

To visualize the Cobb–Douglas function described by Equation (2), a 3D plot was created over the interval 0–1000 for both x and y , as shown in Figure 2a. The value of the function rises as the values of x and y increase, reaching a maximum of 1.5×10^5 without any constraints. However, after applying the constraint specified in Equation (3), the maximum value of the function decreases to 63,895 as depicted in Figure 2b. This result aligns with the previously determined Lagrangian solution of 63,895. Therefore, the maximum value of the Cobb–Douglas function can be inferred from the plots in Figure 2a,b. This method provides a visual confirmation of the Lagrangian solution and also serves as a reference for the genetic algorithm solution discussed in the next section.



(a)



(b)

Figure 2. Plot of Cobb–Douglas function equation (2) in 0–1000 interval (a) without constraint (b) and with constraint Equation (3).

3.2. Solution by Genetic Algorithm (GA)

Following the Lagrangian solution and visual verifications presented in previous sections, this section introduces the solution obtained through a GA. A dedicated computer code was developed for the GA solution, incorporating steps 1–8 as detailed previously.

The computer code calculates the objective function Z (Cobb–Douglas production function) as defined in Equation (2), along with the costs of labor and capital and the budget constraint specified in Equation (3). At each iteration, if the budget constraint is exceeded, the value of the objective function Z is set to zero, prompting the algorithm to search for alternative values that satisfy the budget limit or constraint.

Figure 3 displays the convergence of the labor variable (x) towards the optimal value needed to maximize production under the specified constraints. The labor value converged to 583.33, consistent with the Lagrangian solution, within 20 iterations.

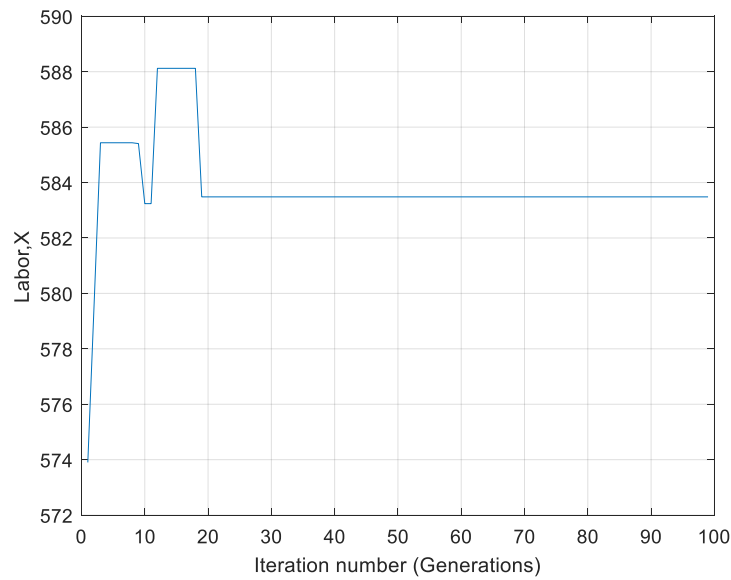


Figure 3. Convergence of labor (x).

In Figure 4, the convergence of the capital variable (y) is shown, with the value settling at 204.55, which matches the Lagrangian solution and is achieved within 20 iterations.

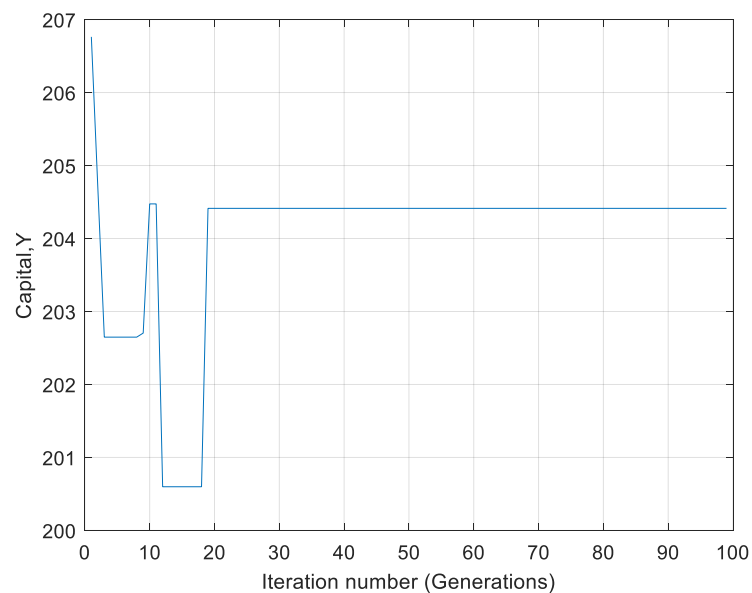


Figure 4. Convergence of capital (y).

Figure 5 below presents the relationship between labor (x) and production units (z), highlighting the distribution of the genetic population or chromosomes. The population predominantly clusters around the labor value of 583.33, corresponding to the maximum production value identified by the GA. Zero values of Z , particularly for X -axis labor values exceeding 600, correspond to populations that failed the test under the constraint specified in Equation (3).

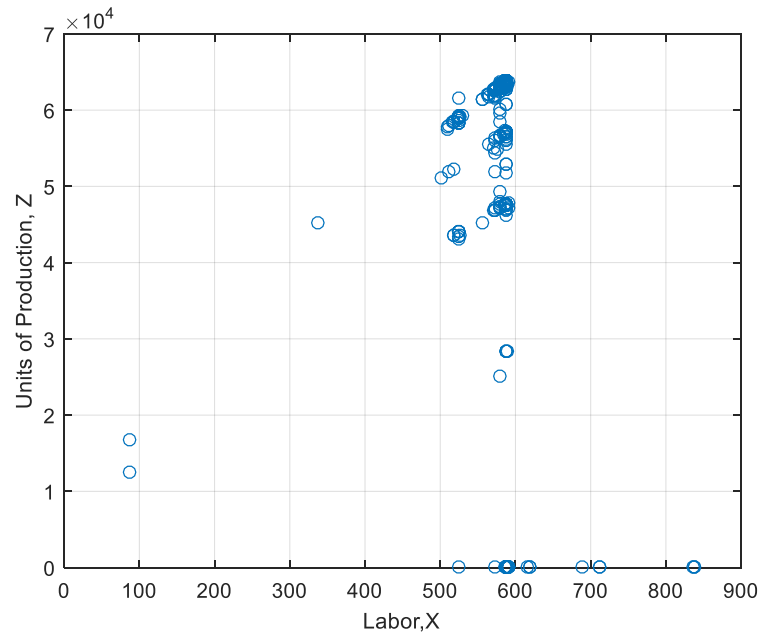


Figure 5. Plot of Cobb–Douglas function vs. labor (x).

Similarly, Figure 6 shows the distribution of the genetic population concerning capital (y) and production units (z). The population converges around the capital value of 204.55, aligning with the optimal production value determined by the GA. Zero values of Z , particularly for X -axis capital values exceeding 205, correspond to populations that failed the test under the constraint specified in Equation (3).

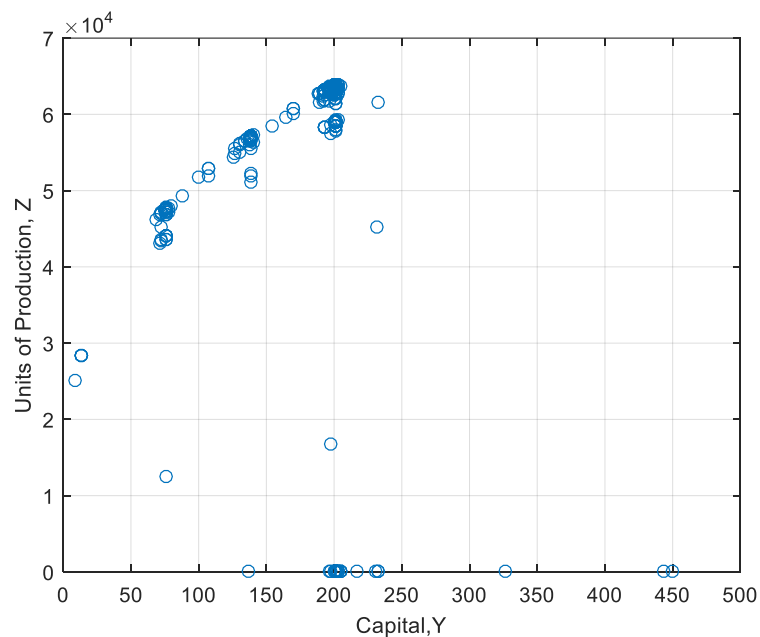


Figure 6. Plot of Cobb–Douglas function vs. capital (y).

Figure 7 shows the convergence of the Cobb–Douglas function (z), representing the maximum units of production under the given constraints, reached a value of 63,895. This result matches the value obtained via the Lagrangian solution (traditional method), with convergence achieved in fewer than 20 iterations.

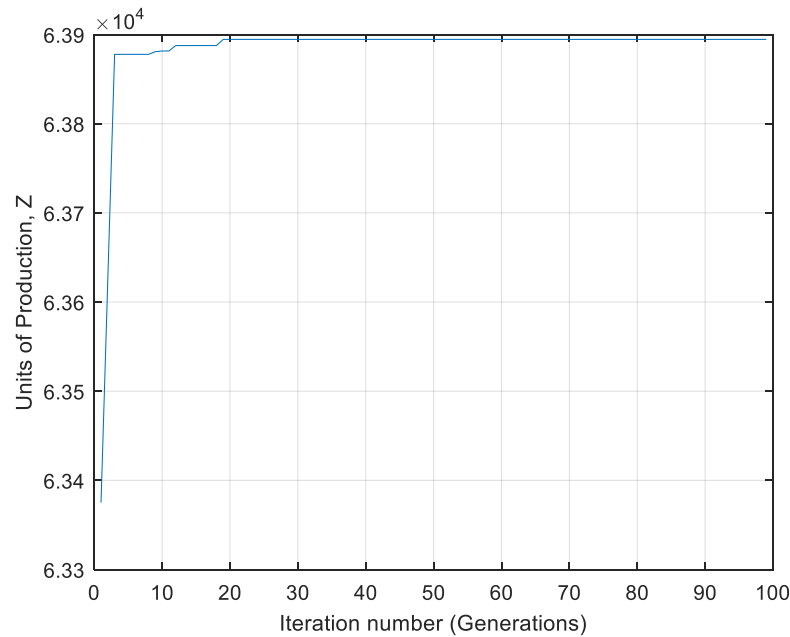


Figure 7. Convergence of Cobb–Douglas function (z).

Figure 8 provides a three-dimensional representation of the genetic population with respect to labor (x), capital (y), and production units (z). The population predominantly accumulates around the maximum production value of 63,895, demonstrating the effectiveness of the GA in identifying the optimal solution.

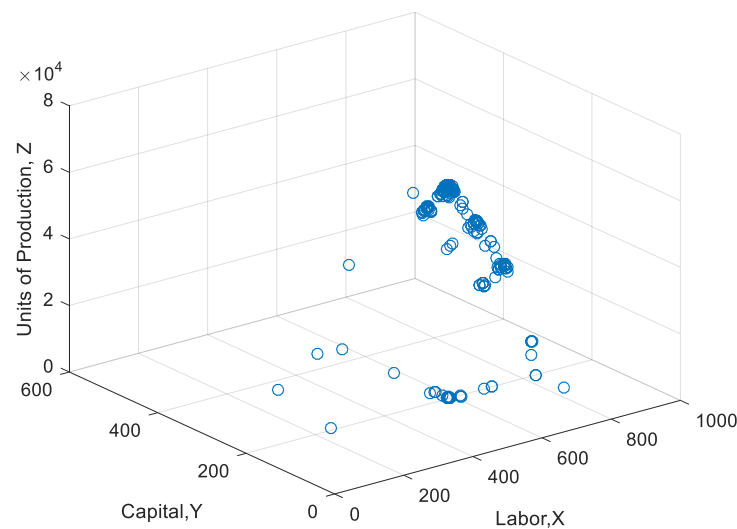


Figure 8. Plot of Cobb–Douglas function vs. capital (y) and labor (x).

The genetic algorithm successfully matched the results of the Lagrangian method, verifying its robustness and efficiency in solving the optimization problem. The swift convergence within 20 iterations highlights the GA’s potential for rapid solution discovery in complex optimization scenarios. The genetic population distributions further validate the accuracy of the GA, showing consistent clustering around optimal values for labor and capital. These findings underscore the GA’s applicability not only in economic optimization

but also in broader contexts requiring complex problem solving and mathematical modeling. The visualizations and step-by-step implementation provided in this study serve as a valuable resource for both students and practitioners, enhancing their understanding of GAs and their practical applications.

Below is a summary of the benchmarking methodology and the comparison of results obtained using partial derivatives against those achieved with the genetic algorithm:

Consistency of Problem Instances

- The same problem instances were used for both methods to ensure a fair comparison. Specifically, the Cobb–Douglas production function and its associated constraints (e.g., budget limits and cost coefficients) were identical in both approaches. This ensured that the solutions from the partial derivatives method and the genetic algorithm could be directly compared.

Error Metrics

- The accuracy of the genetic algorithm was assessed by comparing its results with the optimal solutions obtained analytically using the partial derivatives method, with the deviation found to be less than 0.01%.

Computation Time

- The computation time was not compared due to the differing nature of the two methods. The partial derivative solution was performed manually, whereas the genetic algorithm solution was implemented using a computer.

Overall, this study reaffirms the transformative impact of GAs in optimizing the Cobb–Douglas production function and paves the way for advanced approaches in economic and engineering optimization.

4. Conclusions

This study demonstrates the significant potential of GAs in optimizing the Cobb–Douglas production function. By delving into the mathematical and computational methods utilized in engineering and mathematics, this study underscores the significance of GAs, a cutting-edge method inspired by evolutionary principles, in addressing complex optimization problems. This research highlights the historical evolution and theoretical underpinnings of GAs and showcases their practical application in optimizing economic models such as the Cobb–Douglas function. Through a comparative analysis, this study illustrates the superior robustness and efficiency of GAs in economic optimization. Additionally, the step-by-step implementation, accompanied by visual graphs, enhances comprehension and provides valuable insights into their broader applicability in mathematical modeling and problem solving within the realms of engineering and economics. This work underscores the potential of GAs as a potent tool for optimizing intricate functions and solving multifaceted problems across various domains.

Pedagogical Recommendations:

- **Step-by-Step Implementation:** Incorporating a step-by-step framework for teaching GAs can significantly enhance students' comprehension of the process. Visual aids, detailed procedural descriptions, and real-world examples, like the Cobb–Douglas production function, provide a concrete foundation for understanding the abstract concepts behind GAs.
- **Comparative Analysis:** Teaching GAs alongside traditional methods, such as the Lagrange multiplier technique, allows students to appreciate the strengths and limitations of both approaches. This comparative analysis fosters critical thinking and a deeper understanding of problem-solving methodologies.

- **Interactive Learning:** Practical exercises where students implement GAs to solve optimization problems can reinforce their learning. Tools like C, C++, MATLAB, Python, or specialized genetic algorithm software can be used to provide hands-on experience.
- **Real-World Applications:** Highlighting the interdisciplinary applications of GAs in fields such as engineering, economics, and finance helps students grasp the versatility and relevance of this optimization technique.

By reframing the conclusions as pedagogical recommendations, this study underscores the value of GAs not only as a computational tool but also as a teaching methodology. Future research could explore additional examples and case studies to further enhance the pedagogical utility of GAs in academic and professional settings.

Key findings and implications are summarized as follows:

- **Effectiveness in Economic Optimization:** Genetic algorithms have proven to be robust and efficient tools for addressing complex optimization problems in economic and engineering contexts, with notable improvements in computational accuracy and convergence speed.
- **Quantitative Insights:** The genetic algorithm outperformed traditional methods, achieving the same optimal results with fewer iterations and improved computational efficiency. Metrics such as optimization accuracy, convergence rate, and sensitivity to parameter changes highlight its robustness.
- **Educational Value:** The step-by-step implementation, accompanied by pseudocode and detailed graphs, provides an effective pedagogical tool for teaching genetic algorithms. This approach enhances learners' understanding of evolutionary computation and its applications.
- **Real-World Applicability:** The findings demonstrate the utility of genetic algorithms in practical scenarios, such as resource allocation, production optimization, and financial modeling, where traditional methods may encounter limitations. This approach is particularly advantageous in multi-modal or non-linear problem spaces.
- **Comparison with Traditional Methods:** The genetic algorithm's ability to handle complex constraints and achieve rapid convergence underscores its superiority. The benchmarking against partial derivatives highlights its computational advantages and broader applicability.
- **Limitations and Future Directions:** While effective, the performance of genetic algorithms depends on parameter tuning and initial population selection. Future research should explore adaptive parameterization techniques, real-time optimization applications, and extending the approach to other production functions or multi-objective optimization scenarios.
- **Practical Recommendations:** Researchers and practitioners adopting genetic algorithms are advised to carefully select parameters and benchmark results against traditional methods to validate their efficiency. The detailed implementation provided in this paper serves as a practical guide for leveraging genetic algorithms in diverse optimization problems.

In conclusion, this study reaffirms the transformative impact of genetic algorithms on economic and engineering optimization. By bridging theoretical foundations with practical applications, the research contributes to advancing computational problem-solving methodologies and offers valuable insights for academics, practitioners, and educators. Genetic algorithms are poised to remain an indispensable tool for addressing intricate optimization challenges, driving innovation across multiple disciplines.

Author Contributions: Conceptualization, A.D.; methodology, A.D.; software, A.D.; validation, A.D.; formal analysis, A.D.; investigation, A.D.; data curation, A.D.; writing—original draft preparation, A.D., F.Y., K.N., M.O. and A.M.; writing—review and editing, A.D., F.Y., K.N., M.O. and A.M.; visualization, A.D.; supervision, A.D.; project administration, A.D., F.Y., K.N., M.O. and A.M.; funding acquisition, K.N., M.O. and A.M. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by American University of the Middle East.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alan, J.; Auerbach, L.J.K. *Macroeconomics: An Integrated Approach*; MIT Press: Cambridge, MA, USA, 1998; ISBN 9780262511032.
2. Du, X.; Zhang, L.; Shang, Y.; Li, M. Exact Augmented Lagrangian Function for Nonlinear Programming Problems with Inequality Constraints. *Appl. Math. Mech.* **2005**, *26*, 1649–1656. [[CrossRef](#)]
3. Mitchell, M. *An Introduction to Genetic Algorithms*; The MIT Press: Cambridge, MA, USA, 1998; ISBN 9780262280013.
4. Kramer, O. *Genetic Algorithm Essentials*; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2017; Volume 679, ISBN 978-3-319-52155-8.
5. Coley, D.A. *An Introduction to Genetic Algorithms for Scientists and Engineers*; World Scientific: Singapore, 1999; ISBN 978-981-02-3602-1.
6. Whitley, D. A Genetic Algorithm Tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [[CrossRef](#)]
7. Drake, A.E.; Marks, R.E. Genetic Algorithms In Economics and Finance: Forecasting Stock Market Prices And Foreign Exchange—A Review. In *Genetic Algorithms and Genetic Programming in Computational Finance*; Springer: Boston, MA, USA, 2002; pp. 29–54.
8. Hermawanto, D. Genetic Algorithm for Solving Simple Mathematical Equality Problem. *arXiv* **2013**. [[CrossRef](#)]
9. Thompson, J. Genetic Algorithms and Applications. In *Handbook of Formal Optimization*; Springer Nature: Singapore, 2024; pp. 981–1006.
10. Machacha, L.; Bhattacharya, P. Cancer Classification From DNA Microarray Using Genetic Algorithms and Case-Based Reasoning. In *Research Anthology on Bioinformatics, Genomics, and Computational Biology*; IGI Global: Hershey, PA, USA, 2023; pp. 378–399.
11. Hasan Kazmi, K.; Kumar Bara, A.; Sharma, S.K. Multi-Objective Optimisation of Wire Arc Additive Manufacturing Deposition Using Genetic Algorithm. In *Thermal Claddings for Engineering Applications*; CRC Press: Boca Raton, FL, USA, 2024; pp. 77–91.
12. Anupama; Kumar, C. Analysis of a Multiserver System of Queue-Dependent Channel Using Genetic Algorithm. In *Mathematics and Computer Science Volume 2*; Wiley: Hoboken, NJ, USA, 2023; pp. 337–348.
13. Babu, C.V.S.; Suruthi, G.; Indhumathi, C.; Sushruth, S. Integrated Approach for Network Intrusion Detection. In *Metaheuristic and Machine Learning Optimization Strategies for Complex Systems*; IGI Global: Hershey, PA, USA, 2024; pp. 19–39.
14. Jiao, R.; Zou, F. Quantum-Inspired Genetic Algorithm for Workforce Scheduling in Supply Chain and Logistics Operations. In *Quantum Computing and Supply Chain Management: A New Era of Optimization*; IGI Global: Hershey, PA, USA, 2024; pp. 376–394.
15. Wang, H.; Yu, X.; Lu, Y. A Reinforcement Learning-Based Ranking Teaching-Learning-Based Optimization Algorithm for Parameters Estimation of Photovoltaic Models. *Swarm Evol. Comput.* **2025**, *93*, 101844. [[CrossRef](#)]
16. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Reading, MA, USA, 1989; ISBN 9780201157673.
17. Jamali, A.; Nariman-zadeh, N.; Atashkari, K. Multi-Objective Uniform-Diversity Genetic Algorithm (MUGA). In *Advances in Evolutionary Algorithms*; InTech: Houston, TX, USA, 2008; ISBN 9789537619114.
18. Dinc, A. Optimization of Turboprop ESFC and NOx Emissions for UAV Sizing. *Aircr. Eng. Aerosp. Technol.* **2017**, *89*, 375–383. [[CrossRef](#)]
19. Khajavirad, A.; Michalek, J.J.; Simpson, T.W. A Decomposed Genetic Algorithm for Solving the Joint Product Family Optimization Problem. In Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, HI, USA, 23–26 April 2007; Volume 2, pp. 2111–2124.
20. Dinc, A.; Mamedov, A. Optimization of Surface Quality and Machining Time in Micro-Milling of Glass. *Aircr. Eng. Aerosp. Technol.* **2022**, *94*, 676–686. [[CrossRef](#)]
21. Dinc, A.; Otkur, M. Optimization of Electric Vehicle Battery Size and Reduction Ratio Using Genetic Algorithm. In Proceedings of the 2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE), Athens, Greece, 14–17 July 2020; pp. 281–285.

22. Dinc, A.; Gharbia, Y. Global Warming Potential Estimations of a Gas Turbine Engine and Effect of Selected Design Parameters. In Proceedings of the Volume 8: Energy American Society of Mechanical Engineers, Atlanta, GA, USA, 16 November 2020; Volume 8, pp. 1–7. [[CrossRef](#)]
23. Hedayati-Dezfooli, M.; Moayyedian, M.; Dinc, A.; Abdrabboh, M.; Saber, A.; Amer, A.M. Optimizing Injection Molding for Propellers with Soft Computing, Fuzzy Evaluation, and Taguchi Method. *Emerg. Sci. J.* **2024**, *8*, 2101–2119. [[CrossRef](#)]
24. Pacheco, M.A.C.; Vellasco, M.M.R.; de Noronha, M.F.; Lopes, C.H.P. Intelligent Cash Flow: Planning and Optimization Using Genetic Algorithms. In *Genetic Algorithms and Genetic Programming in Computational Finance*; Springer: Boston, MA, USA, 2002; pp. 239–247.
25. Dawid, H. *Adaptive Learning by Genetic Algorithms*; Lecture Notes in Economics and Mathematical Systems; Springer: Berlin/Heidelberg, Germany, 1996; Volume 441, ISBN 978-3-540-61513-2.
26. Kingdon, J.; Feldman, K. Genetic Algorithms and Applications to Finance. *Appl. Math. Financ.* **1995**, *2*, 89–116. [[CrossRef](#)]
27. Fransisca, D.C.; Sukono; Chaerani, D.; Halim, N.A. Robust Portfolio Mean-Variance Optimization for Capital Allocation in Stock Investment Using the Genetic Algorithm: A Systematic Literature Review. *Computation* **2024**, *12*, 166. [[CrossRef](#)]
28. Harshbarger, R.J.; Reynolds, J.J. *Mathematical Applications for the Management, Life, and Social Sciences*; Houghton Mifflin Company: Boston, MA, USA, 2007; ISBN 9781337296977.
29. Riechmann, T. *Learning in Economics*; Contributions to Economics; Physica-Verlag HD: Heidelberg, Germany, 2001; ISBN 978-3-7908-1384-5.
30. Goldberg, D.E. Sizing Populations for Serial and Parallel Genetic Algorithms. In Proceedings of the 3rd International Conference on Genetic Algorithms, Fairfax, VA, USA, 4–7 June 1989; pp. 70–79.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.