FEN ve MÜHENDİSLİK BİLİMLERİ DERGİSİ

Vol: 7 No:2 Year: 2025 | Araştırma Makalesi/Research Article | e-ISSN: 2667-7989

https://doi.org/10.47112/neufmbd.2025.88

Fibonacci Kodlaması ve *k*-Zeckendorf Gösterimleri ile Yeni Bir Genetik Algoritma Modeli

Yunus Emre GÖKTEPE [1*] Fikri KÖKEN [1] Halime ERGÜN [1]

[1] Necmettin Erbakan University, Seydisehir Ahmet Cengiz Faculty of Engineering, Department of Computer Engineering, Konya, Türkiye

Makale Bilgisi

Geliş Tarihi: 03.11.2024
Kabul Tarihi: 15.12.2024
Yayın Tarihi: 31.08.2025

Anahtar Kelimeler:
Fibonacci sayıları,
Genetik algoritma,
Optimizasyon,
Zeckendorf gösterimleri.

ÖZET

Bu çalışma, karmaşık optimizasyon problemlerini çözmenin verimliliğini artırmak için Fibonacci kodlamasını ve *k*-Zeckendorf gösterimlerini kullanan yeni bir genetik algoritma (GA) modeli sunmaktadır. Geleneksel GA, kromozomları temsil etmek için ikili veya sayısal kodlamayı kullanır, ancak bu çalışma Fibonacci dizisine ve Zeckendorf teoremine dayanan alternatif bir yaklaşım önermektedir. Bu gösterimleri GA çerçevesine dahil ederek, model arama sürecini iyileştirmeyi ve daha etkili çözümlere yol açmayı amaçlamaktadır. Bu yaklaşımın önemi, GA'ların performansını etkileyen kritik bir faktör olan kromozom gösterimini iyileştirme yeteneğinde yatmaktadır. Yeni kodlama şemaları, algoritmanın keşif ve yararlanma aşamalarını geliştirerek, optimum çözümlere doğru daha verimli bir yakınsama sağlar. Model iki zorlu problemde test edilmiştir: yüksek dereceli bir polinomun optimum değerini bulma ve dikdörtgen prizmanın hacmini optimize etme. Sonuçlar, önerilen yöntemin standart GA'dan daha başarılı olduğunu göstermektedir. Bu araştırma, alternatif kromozom gösterimlerinin GA performansını önemli ölçüde etkileyebileceğini göstermektedir. Ayrıca, özellikle karmaşık polinom denklemleri ve geometrik kısıtlamaları içeren optimizasyon problemleri için yeni bir çerçeve sağlar. GA'ların daha faklı optimizasyon problemlerine uygulanabileceğini gösterir ve hesaplamalı matematik ve mühendislik tasarımındaki potansiyelini vurgular.

A Novel Genetic Algorithm Model with Fibonacci Encoding and *k*-Zeckendorf Representations

Article Info

Received: 03.11.2024
Accepted: 15.12.2024
Published: 31.08.2025

Keywords:
Fibonacci numbers,
Genetic algorithm,
Optimization,
Zeckendorf representations.

ABSTRACT

This study presents a new genetic algorithm (GA) model that uses Fibonacci coding and *k*-Zeckendorf representations to improve the efficiency of solving complex optimization problems. Traditional GA uses binary or numerical coding to represent chromosomes, but this study proposes an alternative approach based on Fibonacci sequence and Zeckendorf theorem. By incorporating these representations into the GA framework, it aims to improve the model search process and lead to more efficient solutions. The importance of this approach lies in its ability to improve the chromosome representation, which is a critical factor affecting the performance of GAs. The new coding schemes improve the exploration and exploitation phases of the algorithm, allowing for more efficient convergence towards optimal solutions. The model is tested on two challenging problems: finding the optimum value of a high-degree polynomial and optimizing the volume of a rectangular prism. The results show that the proposed method outperforms the standard GA. This research shows that alternative chromosome representations can significantly affect GA performance. It also provides a new framework for optimization problems, especially those involving complex polynomial equations and geometric constraints. It shows that GAs can be applied to more diverse optimization problems and highlights their potential in computational mathematics and engineering design.

To cite this article:
Göktepe, Y.E., Köken, F. & Ergün, H. (2025). A novel genetic algorithm model with Fibonacci encoding and *k*-Zeckendorf representations. *Necmettin Erbakan University Journal of Science and Engineering, 7*(2), 228-244. https://doi.org/10.47112/neufmbd.2025.88

***Corresponding Author:** Yunus Emre Göktepe, *ygoktepe@erbakan.edu.tr*

INTRODUCTION

Inspired by the principles of natural selection and genetics, genetic algorithms (GAs) operate by iteratively generating a population of potential solutions represented as chromosomes. These chromosomes undergo selection, crossover, and mutation operations to produce new offspring that inherit beneficial traits from their parents. Through this iterative process, the population evolves over generations, gradually improving the quality of solutions until an optimal or near-optimal solution is found [1].

GA offers a robust approach to optimization problems, offering approximate solutions and addressing challenges that may arise during the search process. Optimization refers to the minimization (or maximization) of a specific objective function of multiple decision variables while satisfying functional constraints. Mathematical optimization involves selecting the best element from a given set of available alternatives based on specific criteria. In its simplest form, an optimization problem entails systematically selecting input values from a given set and calculating the function's value to either maximize or minimize a real function. Optimization involves the use of specific techniques to evaluate the most cost-effective and efficient solution for a problem or process design [2–4]. GAs have been widely implemented in various domains for optimization.

A GA-based optimization model was proposed by Srimathi et al. to address resource constraints and activity conflicts in construction projects. Researchers from different fields use genetic algorithms to solve optimization problems. The model uses the selection, mutation, and crossover processes of GA to create a conflict-free schedule. It is noted that the proposed model can be adapted to optimize construction programs and consider additional factors [5].

Ulkir and Akgün investigated the surface roughness of the samples fabricated from polyethylene terephthalate glycol (PETG). They generated a dataset of 25 samples by varying four factors at three levels using a 3D printer. Subsequently, they trained a bidirectional feedback neural network (CFNN) with this dataset and optimized the parameters of the CFNN using genetic algorithms (GA). This study primarily aims to predict and optimize surface roughness in additive manufacturing (AM) using machine learning (ML) techniques [6].

Genetic algorithms represent individuals within a population as chromosomes and apply genetic operators such as selection, crossover, and mutation based on their fitness values. In this way, the population gradually approaches the best solution over time. The performance of the genetic algorithms depends on the choice of chromosome representation. Chromosome representation is one of the most important components of genetic algorithms because it reflects the characteristics of the problem domain under investigation and influences the effectiveness of genetic operators. The most commonly used method for chromosome representation uses bit strings or numerical values. Therefore, there is a need to explore alternative chromosome representation methods.

An optimization method using Fibonacci and genetic algorithm has been proposed to develop a method to reduce maintenance costs in wind turbines. In only the first layer of their proposed model, the researchers used the Fibonacci search algorithm to optimize the number of maintenances in a given lifetime. In the second layer, they applied a genetic algorithm to determine preventive maintenance times and which component to preserve in each preventive maintenance activity [7].

Fibonacci numbers have found applications in various fields, including plant biology, animal population dynamics, art, and music. In mathematics and computer science, Fibonacci numbers possess intriguing properties and are used in diverse applications. They have been employed in image processing, data compression, encryption, and coding theory [8–10]. In computer science, Fibonacci numbers play a significant role in the analysis of algorithm complexity and the design of efficient data

structures. Their wide range of applications has made them a popular topic of study in mathematics and computer science [11,12].

Genetic algorithm and Fibonacci Q-matrix were used for the image encryption problem and an algorithm consisting of diffusion-mixing-diffusion-optimization layers was proposed. Experimental results and security analyses demonstrate that the algorithm exhibits not only high security but also a certain level of robustness suitable for practical applications, along with real-time performance [13].

The objective of this study was to examine the impact of representing chromosomes using Fibonacci numbers on the performance of genetic algorithms. To achieve this, we tested the proposed methods on two different problems. The first problem involved finding the roots of a fourth-degree polynomial, whereas the second problem involved maximizing the volume by bending the edges of a flat plate. These problems are commonly used in the genetic algorithm literature. We compared the performance of the proposed methods with that of the standard genetic algorithm. In addition, we analysed the effect of chromosome length and the parameter k on performance. The results of our study demonstrate that representing chromosomes using Fibonacci numbers yields better results than the standard genetic algorithm.

In the literature, no study has examined Fibonacci integration sequences into chromosomal representations used in genetic algorithms. This novel approach remains unexplored and presents a potential area for research within the field of computational genetics. This study fills this research gap by showcasing the performance enhancement achieved using Fibonacci numbers in chromosome representation. The contributions of this study can be summarized as follows: Proposal of a genetic algorithm model that incorporates Fibonacci numbers and Zeckendorf and *k*-Zeckendorf representations. Definition of the components of the proposed model, including chromosome representation, fitness evaluation, chromosome selection, crossover, and mutation. Application of the proposed model to various optimization problems and comparison of the results with those obtained using traditional genetic algorithm methods. The experimental findings demonstrate the superior accuracy of the proposed model compared with alternative approaches.

MATERIALS AND METHODS

In this section, an in-depth explanation of Fibonacci base and Zeckendorf representations is given and how these values are calculated is shown. Also, the structure of the Genetic algorithm is mentioned and the proposed method is introduced in detail.

Fibonacci Representations

It is known [14] that the Fibonacci sequence $\{F_n\}$ is defined recursively as follows:

$$F_n = \begin{cases} 0, & if\ n = 0 \\ 1, & if\ n = 1 \\ F_{n-1} + F_{n-2}, & if\ n > 1 \end{cases} \quad (1)$$

Leaving the first Fibonacci number, the remaining 12 numbers and their corresponding indices are as follows:

Table 1
The n^{th} Fibonacci numbers.

n	2	3	4	5	6	7	8	9	10	11	12	13
F_n	1	2	3	5	8	13	21	34	55	89	144	233

Every positive integer N can be written as

$$N = F_{k_1} + F_{k_2} + \cdots + F_{k_r}, \qquad k_1 > k_2 > \cdots > k_r \geq 2 \quad (2)$$

which is called a Fibonacci representation of N. There are other ways of representing N as the Fibonacci representation. Zeckendorf's theorem is a reasonably well-known result concerning the possibility of writing positive integers as a sum of distinct Fibonacci numbers provided the k_j satisfy the inequalities $k_j - k_{j+1} \geq 2,\ (j = 1,2,..,.,r-1), k_r \geq 2$. This representation is called Zeckendorf's Fibonacci representation [15,16].

The most well-known method for representing non-negative integers using just the digits $\{0,1\}$ is of course by way of the minimal system. However, there are other ways of representing integers using just $\{0, 1\}$ for which the corresponding situation is not quite so obvious. One such system is Zeckendorf's Fibonacci representation.

Zeckendorf's Representation with Fibonacci Bases

Zeckendorf's theorem states that every positive integer can be represented uniquely as the sum of one or more distinct Fibonacci numbers. This representation is achieved in such a way that the sum excludes any two consecutive Fibonacci numbers [15].

The Zeckendorf theorem presents an alternative to the binary number system, which is useful in applications such as data transmission and compression [17].

Zeckendorf's Fibonacci encoding uses Zeckendorf's Fibonacci representation: Any positive integer $N = (\dots d_3 d_2 d_1)_{fib}$ can be represented as follows:

$$N = \sum_{i=1}^{n} d_i F_{k_i} \tag{3}$$

where $F_{k_i},\ k_1 \geq 2$ is the $k_i{}^{th}$ Fibonacci number, $k_i - k_{i-1} \geq 2$, $d_i \in \{0,1\}$, $d_n = 1$. The Zeckendorf's Fibonacci coding is denoted by $N = (d_i)_{fib}$.

It is seen that for $N \leq 34$, an 8-bit $\{0,1\}$ array is generated. For example, the Zeckendorf's Fibonacci representation for $N = 23$ is $21 + 0 + 0 + 0 + 0 + 2 + 0$ and $N = (1000010)_{fib}$.

A number representation system is typically most useful when it has a unique representation for every integer. Zeckendorf's Fibonacci encoding achieves this by avoiding consecutive 1's. However, we saw that although each number does have such a sum, some numbers have more than one sum, and so their representation is not unique.

Let's slightly modify Zeckendorf's Fibonacci encoding to give a new form $\{0,1\}$ with a single representation.

In [18] a Zeckendorf-Wythoff sequence is defined, where each row is a Fibonacci sequence. The Zeckendorf-Wythoff sequence is designed to divide positive integers into columns ordered by their Zeckendorf representation. If the largest (smallest) Fibonacci number used in the Zeckendorf representation of N is F_k, N starts (ends) with F_k. The k column of the Zeckendorf-Wythoff sequence is composed of the integers N in ascending order, whose Zeckendorf representation ends with F_{k+1}.

Each row of the sequence is composed of N integers with Zeckendorf representations of the same form, i.e., any two consecutive numbers in the row have the indices of the Fibonacci numbers F_k corresponding to the digits used in the Zeckendorf representations with a difference of one [15,16].

The Zeckendorf-Wythoff sequence arranged in 12 columns is shown in Table 2 because we used 8-, 12-, and 14-bit arrays. The 14-bit representation is omitted for brevity.

Table 2
Zeckendorf-Wythoff sequence

$w_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$w_{1,j}$	1	2	3	5	8	13	21	34	55	89	144	233
$w_{2,j}$	4	7	11	18	29	47	76	123	199	322	521	843
$w_{3,j}$	6	10	16	26	42	68	110	178	288	466	754	1220
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$w_{13,j}$	33	54	87	141	228	369	597	966	1563	2529	4092	6621
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$w_{56,j}$	145	235	380	615	995	1610	2605	4215	6820	11035	17855	28890
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Some general properties of the sequence are revealed from the Zeckendorf representations. Let $N = w_{ij}$ be the element in the i −th row and j −th column of the Zeckendorf-Wythoff sequence. The first element in column j is considered $w_{1,j} = F_{j+1}$ and the Zeckendorf representation of each integer listed in column j results in F_{j+1}. It can be exemplified:

$$w_{2,j} = F_{j+1} + F_{j+3}$$
$$w_{13,j} = F_{j+1} + F_{j+3} + F_{j+5} + F_{j+7} \quad (4)$$
$$w_{56,j} = F_{j+1} + F_{j+11}$$

Moreover, each of the indices of the Fibonacci numbers used in the Zeckendorf representation of $w_{i,j+1}$ is the sum of the Fibonacci numbers with one more index than the corresponding indices in the Zeckendorf representation of $w_{i,j}$.

Each row can be seen as a generalized Fibonacci sequence $w_{3,j} = 2F_{j+3}$ or $w_{13,j} = 3F_{j+2} + 9F_{j+3}$. In $w_{i,j}$, each positive integer occurs exactly once, and each consecutive element in the first column is the smallest unused integer [19].

Let $R(N)$ is the number of representations of the non-negative integer N as the sum of different Fibonacci numbers. It is given that for integers N in odd columns of a row, successive values of $R(N)$ form an arithmetic progression and $R(N-1)$ is a constant. Furthermore, if there is an N element of the form $F_n < N < F_{n+1} - 1$ given in a column of the Zeckendorf-Wythoff sequence, then $N + F_{n+k}, k \geq 2$ is an element in the same column. Then properties of the Zeckendorf representations and Wythoff pairs are then applied to count $R(N)$ [15,16].

Because each row contains the generalized Fibonacci sequence of positive integers, $R(N) = R(w_{i,j} - 1)$ will be a constant for the integers N in any given row for sufficiently large j.

The number of representations of N as sums of different Fibonacci numbers can be written from the Zeckendorf representation of N such that; if the Zeckendorf representation of N ends with $F_k, k \geq 2$, then a constant q.

$$R(N) = R(N-1)R(F_k) - q, 0 \leq q \leq R(N-1) \quad (5)$$

where $R(F_k) = \left[\!\left[\frac{k}{2}\right]\!\right]$ and $R(F_k - 1) = 1, k \geq 1$, and $[\![x]\!]$ is the greatest integer less than a real number x.

Table 3 lists the number of different Fibonacci representations of numbers N according to columns in Table 2.

Table 3
$R(N)$ for N in Table 2

j	1	2	3	4	5	6	7	8	9	10	11	12
$R(F_{j+1})$	1	1	2	2	3	3	4	4	5	5	6	6
$R(w_{2,j})$	1	1	3	3	5	5	7	7	9	9	11	11
$R(w_{3,j})$	2	2	4	4	6	6	8	8	10	10	12	12

An alternative canonical representation can be derived to express any given number as a summation of Fibonacci numbers, while also allowing for the use of consecutive Fibonacci numbers in the same sum. This requirement can be satisfied by employing a base system with Fibonacci numbers as placeholders, allowing for the occurrence of adjacent numbers. This condition is justified by the fact that the sum of any two consecutive Fibonacci numbers corresponds to the subsequent Fibonacci number. As a result, the representation $(..100..)_{fib}$ can always be substituted with $(..011..)_{fib}$. In this investigation, each number $(..100..)_{fib}$ was transformed into $(..011..)_{fib}$. This representation not only serves as a canonical form but also offers a more compact alternative to Zeckendorf's Fibonacci encoding.

To convince yourself that every number can be represented in this system, write down the *k*-Zeckendorf representations of some numbers for 10 bits. It starts as follows:

Table 4
Representations of k-Zeckendorf using Zeckendorf's Fibonacci coding

N	Zeckendorf's code	*1*-Zeckendorf's code	*2*-Zeckendorf's code	*3*-Zeckendorf's code	*4*-Zeckendorf's code
3	100	011			
5	1000	0110			
8	10000	01100	01011		
13	100000	011000	010110		
21	1000000	0110000	0101100	0101011	
34	10000000	01100000	01011000	01010110	
55	100000000	011000000	010110000	010101100	010101011
89	1000000000	0110000000	0101100000	0101011000	0101010110

It is seen that the least number of Fibonacci numbers is the number of 1s in the Zeckendorf's Fibonacci coding, since the Zeckendorf's theorem guarantees the least number of Fibonacci's and is also called the minimal Fibonacci representation.

Genetic Algorithms in Optimization

Genetic algorithms represent a computational methodology that emulates natural evolutionary processes and are widely employed to address search and optimization challenges. The genetic algorithm technique was first proposed by Professor John Holland [20]. Because of its broad perspective and applicability, it has been applied to many optimization and search problems.

The genetic algorithm technique is an iterative optimization method. It works by simultaneously considering a certain number of possible solutions for a given problem in each iteration. It starts working with randomly generated solutions called chromosomes. It continues to search for the best by using the genetic operators of reproduction, crossover, and mutation in each iteration.

The structure of a genetic algorithm mainly consists of chromosome encoding, fitness calculation, chromosome selection, and recombination [21].

A genetic algorithm considers a population of individuals representing possible solutions to a problem. These individuals, called chromosomes, are formed by the combination of a certain number of genes. The structure created by expressing these genes with letters, numbers, or other representations is

called chromosome encoding.

Determining how suitable the chromosomes in the population are for the problem being examined is done by fitness calculation. The calculated value provides information about the quality of the chromosome and determines the probability of its transmission to the next generation during the selection stages. For example, the One-Max problem, which is a frequently used test problem, tries to maximize the number of 1 bit in a chromosome of a certain length. The fitness value for this problem is simply the number of 1s in the bit string of the chromosome.

In the chromosome selection part of the genetic algorithm, chromosomes whose quality is determined through fitness functions are selected for recombination. Chromosomes with high fitness values are more likely to be selected for recombination. The most commonly used methods are roulette wheel selection, tournament selection, and random stochastic selection [22].

Recombination involves improving the number of chromosomes coming from the selection phase using genetic algorithms. The aim of this study was to preserve the genetic information available and transmit it by providing a certain amount of mixture while creating a new generation. Thus, chromosomes with higher fitness values can be obtained. The two most important operators used at this stage are genetic crossover and mutation. Mixing of selected parent chromosomes based on one or more determined points is called crossover. Mutation acts on individual chromosomes and changes a specified gene.

The GA code will continue iteratively until a specified stopping criterion is met. The population obtained after the recombination steps was evaluated using the fitness function. This is because the new population obtained at this stage is accepted as the input population for the next iteration. The pseudocode of the basic genetic algorithm consisting of the aforementioned stages can be given as shown in Algorithm 1.

Algorithm 1. Pseudocode for the genetic algorithm

```
Begin
    Initialize the population
    Evaluate the population
    While not (stop condition reached)
        Chromosome selection
        Perform the crossover operator
        Perform the mutation operator
        Evaluate the population
    Return best chromosomes
End
```

Proposed Method

The genetic algorithm attempts to obtain the result by searching for the most appropriate chromosomes through candidate solutions. For a given problem to be solved using a GA approach, possible solution candidates must be represented in accordance with the problem.

In this study, our proposed genetic algorithm approach facilitates the representation of chromosomes using Fibonacci numbers. Initially, we generated the initial population from randomly chosen values. There are two different versions of the proposed genetic algorithm model, based on the differentiation of the selected chromosome representation method. To express the chromosomes within this population using Fibonacci numbers, we derived both Zeckendorf and *k*-Zeckendorf representations. Subsequently, we computed the fitness values of the chromosomes. Iteratively, we executed steps involving chromosome selection, operation of genetic operators, and generation of a new offspring until the stipulated termination criterion was met. After fulfilling the termination criterion, we presented the best-performing chromosome alongside its corresponding optimal fitness values as the

model output. The flowchart of the proposed model is shown in Figure 1.

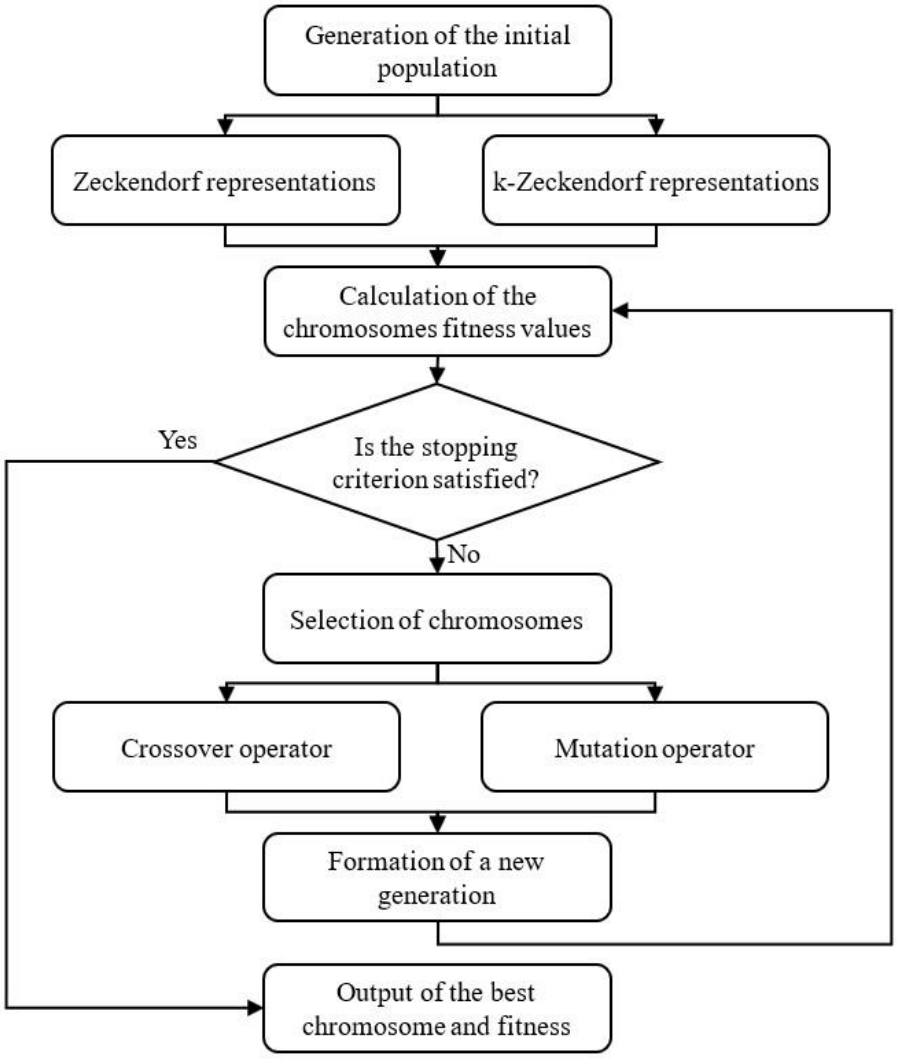

Figure 1
Flowchart of the proposed model.

The proposed model performs specific calculations for whichever of the two different representations (Zeckendorf / *k*-Zeckendorf) is chosen and continues to work on this representation throughout all iterations. As shown in Figure 1, the Zeckendorf/*k*-Zeckendorf representations are calculated for each individual in the initial randomly generated population. The pseudocode of the model is given in Algorithm 2. The for loop, which starts in the third step of Algorithm 2, involves calculating the representations for all chromosomes in the population.

The code in the 4th step of Algorithm 2 indicates the Zeckendorf representation of a chromosome. The pseudocode that corresponds to this step and calculates the Zeckendorf representations of a chromosome is given in algorithm 3. In this code, the Zeckendorf representation of the chromosome is kept in a variable that is initialized to null. The chromosome value was compared with all Fibonacci numbers from largest to smallest within the cycle. In each cycle, the Fibonacci number with a maximum value less than or equal to the chromosome value is searched. At the end of the for loop, the unique Zeckendorf representation for the chromosome is calculated. Because this code is in the for loop of Algorithm 2, it performs these calculations for the entire population.

Algorithm 2. Pseudocode for the proposed algorithm

```
1   Begin
2     Generate a random initial population
3     For each chromosome in the population
4       Calculate Zeckendorf representations of chromosomes
5       If model = k-Zeckendorf, then
6         Find k-Zeckendorf representations
7       Calculate the fitness function of all chromosomes
8     While not (stop condition reached)
9       Execute the chromosome selection step
10      Perform the crossover operator
11      Define mutation rate using Fibonacci numbers
12      Perform the mutation operator
13      Evaluate the population
14    Obtain chromosome values from the Zeckendorf / k-Zeckendorf representations
15    Return best chromosome/chromosomes
16  End
```

Algorithm 3. Pseudocode for calculating Zeckendorf representations

```
1   Begin
2     Initialize var to maintain Zeckendorf representation of the given chromosome
3     For each item in the Fibonacci array
4       If chromosome > item, then
5         Insert "1" into var
6         Decreased value of the chromosome by item
7       Else
8         Insert "1" into var
9     Return var
10  End
```

If the proposed model works on *k*-Zeckendorf representations, the function of finding *k*-Zeckendorf representations works in the sixth step of algorithm 2. The pseudocode of the program running in this function is given in algorithm 4.

Algorithm 4. Pseudocode to find *k*-Zeckendorf representations

```
1   Begin
2     While (any changes made)
3       For i = 0 to i < length(chromosome)
4         If gene_i = "1" and gene_i+1 = "0" and gene_i+2 = "0" then
5           gene_i = "0" and gene_i+1 = "1" gene_i+2 = "1"
6           break
7     Return gene
8   End
```

The basic logic of the *k*-Zeckendorf representation is that the bit with the largest value in the chromosome is distributed to the two bits before it. This process is performed by assigning the value of the bit with the largest value to the two largest bits that are smaller than itself. In this way, in most cases, the chromosome can be expressed with a smaller number of bits without losing its value.

By distributing one bit within the chromosome to two smaller bits, a different representation of the same chromosome is obtained. After such a change, repeated distribution of another gene within the same chromosome may be possible. This means that a new and different representation can be obtained for the same chromosome. At this point, our approach for obtaining the *k*-Zeckendorf representation is based on all possible distributions. As a result of this process, the last representation with no possibility of a new distribution is accepted as the expression of the chromosome.

After this point, the fitness values of the chromosomes are calculated. Chromosome selection was

performed according to the roulette wheel method. The crossover genetic operator is run on the selected chromosomes.

The golden ratio was used to determine the mutation rate. The formula for the golden ratio is expressed as $\alpha = (1+\sqrt{5})/2$, which corresponds to the value $\cong 1.618$. Because the mutation rate should be less than 1, we used this value as $1/\alpha$. At each iteration, we calculate the mutation rate by multiplying it by the reciprocal of the variable α, denoted as $1/\alpha$. As the iterations progress, it is expected that the chromosomes will approach the ideal point. In this way, as we get closer to the target, fewer mutations will be made compared with the first iterations.

When the termination criterion is met, iterations are concluded, and the output includes the best chromosome (closest to the target) and the corresponding highest fitness value.

The genetic algorithm model with Fibonacci numbers and Zeckendorf representations used in this study was implemented in Python.

EXPERIMENTAL RESULTS

In the course of this investigation, efforts were made to enhance the genetic algorithm optimization method. Chromosomal representations were generated using Fibonacci numbers. The outcomes derived from this approach were assessed through a comparative analysis with those obtained using the standard genetic algorithm.

For this purpose, polynomials of high degree [23] and maximum volume problems [24] are considered.

Polynomials of the High Degree

Exploration of maximum and minimum points in polynomial functions is an extensively studied topic in the field of mathematics. Notably, the identification of extrema becomes progressively more challenging and resource-intensive when dealing with higher-degree polynomials [23].

The general expression for polynomials is given by Equation 6. By selecting coefficients (denoted as 'a') and values for the variable 'x' from the set of real numbers, a polynomial of the desired degree can be obtained.

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \, , a_n, x \in R \, , a_n \neq 0 \tag{6}$$

In this study, two polynomials of the fourth and fifth degrees were evaluated. The coefficients of the polynomials were chosen randomly, resulting in the formulation of the polynomials as shown in Equations 7 and 8. These polynomials were then used as input for the model.

$$P_4(x) = 5x^4 - 7x^3 + 3x^2 - 8x + 11 \tag{7}$$

$$P_5(x) = 6x^5 - 3x^4 + 4x^3 - 2x^2 + 9x - 7 \tag{8}$$

Maximum Volume Problem

The problem at hand is the optimization of a rectangular prism's dimensions in order to achieve the maximum volume, given a limited amount of material for its construction. This situation commonly arises when faced with the task of creating a rectangular container, such as a box, using a fixed area of material, as depicted in Figure 2.

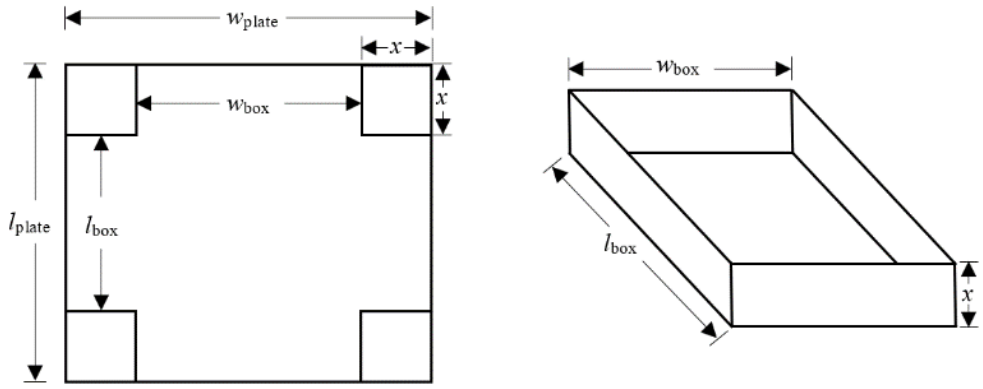

Figure 2
Shema Maximum volume problem (a) equal squares of side length x from each corner of the cardboard, (b) folding these edges.

The goal is to maximize the volume of the box by cutting equal squares with side length x from each corner of a rectangular piece of cardboard with dimensions a inches and b inches, and then folding the resulting sides, as illustrated in Figure 2.

Consider a rectangular box characterized by the dimension length *l*, width *w*, and height *x*, yielding a volume $V = l*w*x$. Assuming a predetermined amount of material, the construction of the box entails cutting squares of side length *x* from each corner of a rectangular sheet. Subsequently, the residual material is folded to form the box. Expressing the volume in terms of *x* yields the formulation given in Equation 9.

$$V(x) = x * (l - 2 * x) * (w - 2 * x) \tag{9}$$

The determination of the optimal value of *x* that maximizes the volume of the box can be calculated using the principals of calculus.

Expressing the volume $V(x)$ of the box as a function of x, we obtain $V(x) = 4x^3 - 2(l + w)x^2 + lwx$ which can be considered as the problem of finding the value that x can take to have the maximum volume.

If $l = F_n$ and $w = F_{n+1}$ are selected, a golden rectangle can be constructed, and we have:

$$V(x) = 4x^3 - 2F_{n+2}x^2 + F_nF_{n+1}x \tag{10}$$

then it is worked on a golden rectangle, where F_nF_{n+1} is golden rectangle numbers A001654.

A similar rectangle can be constructed when $l = F_n$ and $w = F_{n+2}$ are taken as:

$$V(x) = 4x^3 - 2L_{n+1}x^2 + F_nF_{n+2}x \tag{11}$$

where L_n is the Lucas number.

Comparison of Experimental Results

We initially assessed the efficacy of the proposed approach using a sample problem featuring a fourth-order polynomial. We tested the genetic algorithm and the proposed approaches separately for polynomial solutions using 8-bit, 12-bit, and 14-bit long chromosomes.

The results obtained by solving the fourth-order sample polynomial problem created with randomly selected coefficients and given in Equation 7 are shown in Table 5.

Table 5
Comparison of the results obtained using the proposed method for a fourth-degree polynomial problem with those obtained from the genetic algorithm.

Solution method	Coding base	Value of the best chromosome	Maximum fitness value	Increase in fitness according to the genetic algorithm
Genetic algorithm (standard solution)	8-bit	250	$1,94*10^{10}$	-
	12-bit	4063	$1,36*10^{15}$	-
	14-bit	16244	$3,48*10^{17}$	-
Genetic algorithm with Zeckendorf representations	8-bit	255	$2,1*10^{10}$	8,3%
	12-bit	4084	$1,39*10^{15}$	2,08%
	14-bit	16374	$3,59*10^{17}$	3,24%
Genetic algorithm with *k*-Zeckendorf representations	8-bit	255	$2,1*10^{10}$	8,3%
	12-bit	4095	$1,41*10^{15}$	3,19%
	14-bit	16382	$3,6*10^{17}$	3,44%

As shown in Table 5, when 8-bit chromosomes are used, the best chromosome values obtained for the genetic algorithm, Zeckendorf, and *k*-Zeckendorf approaches are 250, 255, and 255, respectively. The maximum fitness values are 1.941010, 2.11010, and 2.11010, respectively. Considering the maximum fitness values obtained, the Zeckendorf and *k*-Zeckendorf approaches gave 8.3% better results than the genetic algorithm. When 12-bit chromosomes were employed, the Zeckendorf and *k*-Zeckendorf approaches yielded maximum fitness values that were 2.08% and 3.19% better than those obtained with the genetic algorithm, respectively. Similarly, with the use of 14-bit chromosomes, both the Zeckendorf and *k*-Zeckendorf approaches exhibited superior performance compared with the genetic algorithm, achieving 3.24% and 3.44% better maximum fitness values, respectively.

The results obtained by solving the fifth-degree sample polynomial problem, which was developed with randomly selected coefficients as presented in Equation 8, are depicted in Table 6.

Table 6
Comparison of the results obtained using the proposed method for a fifth-degree polynomial problem with those obtained from the genetic algorithm.

Solution method	Coding base	Value of the best chromosome	Maximum fitness value	Increase in fitness according to the genetic algorithm
Genetic algorithm (standard solution)	8-bit	250	$5,85*10^{12}$	
	12-bit	4053	$6,56*10^{18}$	-
	14-bit	16157	$6,61*10^{21}$	-
Genetic algorithm with Zeckendorf representations	8-bit	254	$6,33*10^{12}$	8,3%
	12-bit	4081	$6,79*10^{18}$	3,5%
	14-bit	16377	$7,07*10^{21}$	7%
Genetic algorithm with *k*-Zeckendorf representations	8-bit	255	$6,46*10^{12}$	10,4%
	12-bit	4087	$6,84*10^{18}$	4,27%
	14-bit	16383	$7,08*10^{21}$	7,2%

As indicated in Table 6, the Zeckendorf and *k*-Zeckendorf approaches demonstrated superior performance, exhibiting improvements ranging from 3.5% to 10.4% for problems that can be represented by a fifth-degree polynomial.

In the maximum volume problem, the objective is to bend the edges of a flat plate to maximize the volume relative to its dimensions. In this context, the model was executed for three sample sheets characterized by side length ratios of 1/2, 1/3, and 1/8. The dimensions of these sheets are specified as 4096x8192, 1024x3072, and 2048x16384, respectively. The outcomes derived for these sample sheets are illustrated in Table 7.

Table 7
Comparison of the results obtained by the proposed method for the maximum volume problem with those obtained from the genetic algorithm.

		Genetic algorithm (standard solution)		Genetic algorithm with Zeckendorf representations		Genetic algorithm with *k*-Zeckendorf representations	
Sheet size	**Optimum bending length**	**Obtained fitness value**	**Error rate (deviation from optimum)**	**Obtained fitness value**	**Error rate (deviation from optimum)**	**Obtained fitness value**	**Error rate (deviation from optimum)**
4096x8192	865,6	851	1,69%	878	1,43%	868	0,28%
1024x3072	231,1	235	1,68%	234	1,24%	231	0,05%
2048x16384	495,0	486	1,82%	490	1,01%	496	0,20%

As indicated in Table 7, the anticipated optimal bending points for plates of sizes 4096x8192, 1024x3072, and 2048x16384 are 866, 231, and 495, respectively. However, the genetic algorithm method yielded a bending point of 851 for the 4096x8192 plate. On the other hand, our Zeckendorf approach identified it as 878, and the *k*-Zeckendorf approach determined it to be 868.

Evaluating these outcomes in terms of deviation (error) rates from the optimal point of 851, we observe error rates of 1.69%, 1.43%, and 0.28% for the genetic algorithm, Zeckendorf, and *k*-Zeckendorf approaches, respectively, for the 4096x8192 plate. For the 1024x3072 plate, the error rates were 1.68%, 1.24%, and 0.05%, respectively. Meanwhile, for the 2048x16384 plate, the error rates were 1.82%, 1.01%, and 0.20%, respectively.

In addition, the maximum volume problem was tested on a sample golden rectangle formed by selecting the coefficients from the Fibonacci numbers, as given in Equation 10, and on rectangular plates formed by selecting the coefficients from the Fibonacci and Lukas numbers, as given in Equation 11, and the results were evaluated. When the Fibonacci numbers f_{19}, f_{20}, and f_{21} are selected for the coefficients according to Equation 10, the equation in Equation 12 is obtained.

$$V(x) = 4x^3 - 2F_{21}x^2 + F_{19}F_{20}x \tag{12}$$

When Equation 11 is used, the f_{19} and f_{21} fibonacci numbers and the lukas number l_{20} are determined as coefficients, and the equation in Equation 13 is obtained.

$$V(x) = 4x^3 - 2L_{20}x^2 + F_{19}F_{21}x \tag{13}$$

The outcomes derived from these special rectangle sheets are presented in Table 8.

Table 8
Comparison of the results obtained using the proposed method for the maximum volume problem of special rectangles with those obtained from the genetic algorithm.

		Genetic algorithm (standard solution)		Genetic algorithm with Zeckendorf representations		Genetic algorithm with *k*-Zeckendorf representations	
Special rectangles	**Optimum bending length**	**Obtained fitness value**	**Error rate (deviation from optimum)**	**Obtained fitness value**	**Error rate (deviation from optimum)**	**Obtained fitness value**	**Error rate (deviation from optimum)**
Equation 12	838,9	831	0,94%	836	0,35%	840	0,13%
Equation 13	926,6	933	0,69%	924	0,28%	927	0,04%

As indicated in Table 8, the expected optimal bending points for the golden rectangle determined by Equation 12 and the special rectangle determined by Equation 13 are 838.9 and 926.6, respectively.

The 0.94% deviation rate obtained with the standard genetic algorithm method for the golden rectangle was reduced to 0.35% with the Zeckendorf approach and 0.13% with the *k*-Zeckendorf approach. Similarly, for the special rectangle specified in Equation 13, the 0.69% deviation rate obtained by the genetic algorithm method was reduced to 0.28% with the Zeckendorf approach and 0.04% with the *k*-Zeckendorf approach.

Similarly, for the special rectangle specified in Equation 13, the 0.69% deviation rate obtained by the genetic algorithm method was reduced to 0.28% with the Zeckendorf approach and 0.04% with the *k*-Zeckendorf approach.

These findings indicate that the proposed Zeckendorf and *k*-Zeckendorf methods outperform the genetic algorithm approach because they exhibit lower error rates across the evaluated plate sizes.

We initially assessed the efficacy of the proposed approach using a sample problem featuring a fourth-order polynomial. We tested the genetic algorithm and the proposed approaches separately for polynomial solutions using 8-bit, 12-bit, and 14-bit long chromosomes. The results obtained by solving the fourth-order sample polynomial problem created with randomly selected coefficients and given in Equation 7 are shown in Table 9.

Table 9
Comparison of the results obtained using the proposed method for a fourth-degree polynomial problem with those obtained from the genetic algorithm.

Solution method	Coefficients	Value of the best chromosome	Maximum fitness value	Increase in fitness according to the genetic algorithm
Genetic algorithm (standard solution)	8-bit	250	$1{,}94*10^{10}$	
	12-bit	4063	$1{,}36*10^{15}$	-
	14-bit	16244	$3{,}48*10^{17}$	-
Genetic algorithm with Zeckendorf representations	8-bit	255	$2{,}1*10^{10}$	8,3%
	12-bit	4084	$1{,}39*10^{15}$	2,08%
	14-bit	16374	$3{,}59*10^{17}$	3,24%
Genetic algorithm with *k*-Zeckendorf representations	8-bit	255	$2{,}1*10^{10}$	8,3%
	12-bit	4095	$1{,}41*10^{15}$	3,19%
	14-bit	16382	$3{,}6*10^{17}$	3,44%

The obtained results demonstrate that the proposed genetic algorithm models exhibit better performance than the standard genetic algorithm. These models facilitate the representation of chromosomes using Fibonacci numbers.

There are certain limitations and challenges in this study. First, the proposed genetic algorithm models were tested on only two distinct problems: the fourth-degree polynomial problem and the maximum volume problem. These problems represent only a few among the wide range of problems for which genetic algorithms can be applied. Therefore, it is necessary to test the effectiveness of the proposed models on other problems. Second, note that the proposed genetic algorithm models are dependent on the parameters of the Zeckendorf and *k*-Zeckendorf representations. These parameters involve Fibonacci selection numbers and determination of the value of k. Further research is needed to investigate how these parameters affect the performance of the proposed models and how they can be optimized.

CONCLUSION

In this study, we investigated the use of genetic algorithms (GAs) for optimization by leveraging their capacity to simulate the evolutionary processes found in nature. GAs have been widely applied to diverse search and optimization challenges because of their adaptability and efficacy. The fundamental structure of GA encompasses several essential elements, including chromosome encoding, fitness

evaluation, chromosome selection, and recombination. These elements iteratively collaborate to explore and identify optimal solutions for a given problem.

Our study introduced a novel approach that employs Fibonacci numbers for chromosome representation within GAs. This approach offers two distinct representations: Zeckendorf and *k*-Zeckendorf, each offering unique advantages. Through experimental analysis, we demonstrated the efficacy of our approach by comparing it with standard GA solutions for polynomial optimization and maximum volume problems.

The results of our experiments reveal that the Zeckendorf and *k*-Zeckendorf methodologies surpass the standard GA in terms of fitness values and error rates across the evaluated problems. The adoption of Fibonacci numbers as the foundation for chromosome representation presents a promising avenue for enhancing the performance of GAs in optimization tasks.

In summary, the adoption of Fibonacci numbers for chromosome representation in GA exhibits considerable potential for advancing optimization outcomes. Continued research in this area may yield further advancements in optimization algorithms' efficiency and efficacy.

Ethical Statement
This study is an original research article designed and developed by the authors.

Ethics Committee Approval
This study does not require any ethics committee approval.

Author Contributions
Research Design (CRediT 1) Y.E.G.: (%20) – F.K. (%40) – H.E. (%40)
Data Collection (CRediT 2) Y.E.G.: (%30) – F.K. (%40) – H.E. (%30)
Research - Data Analysis – Validation (CRediT 3-4-6-11) Y.E.G.: (%70) – H.E. (%30)
Writing the Article (CRediT 12-13) Y.E.G.: (%40) – F.K. (%20) – H.E. (%40)
Revision and Improvement of the Text (CRediT 14) Y.E.G.: (%60) – F.K. (%20) – H.E. (%20)

Financing
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest
The authors declare no conflict of interest.

Sustainable Development Goals (SDG)
Sustainable Development Goals: 8 Decent work and economic growth.

REFERENCES

[1] M. Mitchell, An Introduction to Genetic Algorithms, *MIT Press*, 1998.

[2] A.O. Faouri, P. Kasap, Maximum likelihood estimation for the Nakagami distribution using particle swarm optimization algorithm with applications, *Necmettin Erbakan University Journal of Science and Engineering*. 5(2) (2023), 209-218. doi:10.47112/neufmbd.2023.19.

[3] A. Pektaş, İ. Onur, Ağaç tohum algoritmasının kümeleme problemlerine uygulanması, *Necmettin Erbakan University Journal of Science and Engineering*. 4 (2022), 1-10.

[4] A. Ünlü, İ. İlhan, A novel hybrid gray wolf optimization algorithm with harmony search to solve multi-level image thresholding problem, *Necmettin Erbakan University Journal of Science and Engineering*. 5(2) (2023), 230-245. doi:10.47112/neufmbd.2023.21.

[5] K.R. Srimathi, A. Padmarekha, K.S. Anandh, Automated construction schedule optimisation using genetic algorithm, *Asian Journal of Civil Engineering*. 24 (2023), 3521-3528. doi:10.1007/s42107-023-00729-8.

[6] O. Ulkir, G. Akgun, Predicting and optimising the surface roughness of additive manufactured parts using an artificial neural network model and genetic algorithm, *Science and Technology of Welding and Joining*. 28 (2023), 548-557. doi:10.1080/13621718.2023.2200572.

[7] E. Singh, S.S. Afshari, X. Liang, Wind turbine optimal preventive maintenance scheduling using Fibonacci search and genetic algorithm, *Journal of Dynamics, Monitoring and Diagnostics*. 2 (2023), 157-169.

[8] M. Basu, M. Das, Uses of second order variant Fibonacci universal code in cryptography, *Control and Cybernetics*. 45 (2016), 239-251.

[9] A. Rehman, T. Saba, T. Mahmood, Z. Mehmood, M. Shah, A. Anjum, Data hiding technique in steganography for information security using number theory, *Journal of Information Science*. 45 (2019), 767-778. doi:10.1177/0165551518816303.

[10] L. Wu, H. Cai, Novel stream ciphering algorithm for big data images using Zeckendorf representation, *Wireless Communications and Mobile Computing*. 2021 (2021), 1-19. doi:10.1155/2021/4637876.

[11] M.S. Taha, M.S.M. Rahem, M.M. Hashim, H.N. Khalid, High payload image steganography scheme with minimum distortion based on distinction grade value method, *Multimedia Tools and Applications*. 81 (2022), 25913-25946. doi:10.1007/s11042-022-12691-9.

[12] Y. Wu, L. Wu, H. Cai, Cloud-edge data encryption in the internet of vehicles using Zeckendorf representation, *Journal of Cloud Computing*. 12 (2023), 39. doi:10.1186/s13677-023-00417-7.

[13] Z. Liang, Q. Qin, C. Zhou, An image encryption algorithm based on Fibonacci Q-matrix and genetic algorithm, *Neural Computing and Applications*. 34 (2022), 19313-19341.

[14] T. Koshy, Fibonacci and Lucas Numbers with Applications, Volume 2, *John Wiley & Sons*, 2019.

[15] C. Kimberling, The Zeckendorf array equals the Wythoff array, *Fibonacci Quarterly*. 33 (1995), 3-8.

[16] C. Kimberling, Edouard Zeckendorf, *Fibonacci Quarterly*. 36 (1998) 416-418.

[17] P. Pooksombat, P. Udomkavanich, W. Kositwattanarerk, Multidimensional Fibonacci Coding, (2017). http://arxiv.org/abs/1706.06655 (erişim 09 Mart 2024).

[18] M. Bicknell-Johnson, The Zeckendorf-Wythoff Array Applied to Counting the Number of Representations of N as Sums of Distinct Fibonacci Numbers, in: F.T. Howard (Ed.), *Applications of Fibonacci Numbers*, *Springer Netherlands*, Dordrecht, 1999: ss. 53-60.

[19] W. Lang, ed., The Wythoff and the Zeckendorf Representations of Numbers Are Equivalent,

Kluwer Academic Publishers, Dordrecht, 1996. doi:10.1007/978-94-009-0223-7.

[20] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, *The MIT Press*, 1992. doi:10.7551/mitpress/1090.001.0001.

[21] J. McCall, Genetic algorithms for modelling and optimisation, *Journal of Computational and Applied Mathematics*. 184 (2005), 205-222. doi:10.1016/j.cam.2004.07.034.

[22] D. Dumitrescu, B. Lazzerini, L.C. Jain, A. Dumitrescu, Evolutionary Computation, *CRC Press*, 2000.

[23] T. DeAlwis, Maximizing or minimizing polynomials using algebraic inequalities, in: *Proceedings of the 9th Asian Technological Conference on Mathematics*, 2004: ss. 88-97. https://atcm.mathandtech.org/EP/2004/2004I328/fullpaper.pdf (erişim 21 Mart 2024).

[24] A. Mikhalev, I.V. Oseledets, Rectangular maximum-volume submatrices and their applications, *Linear Algebra and its Applications*. 538 (2018), 187-211. doi:10.1016/j.laa.2017.10.014.