Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 3, December 2025, pp. 1669~1688
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i3.pp1669-1688

Proposition of a new fitness function: Hadj-said fitness function

Hana Ali Pacha[1], Abderrahmene Hadj Brahim[2], Pascal Lorenz[3]
[1]Quartz Laboratory ECAM-EPMI, 13 Bd de l'Hautil, Cergy Pontoise, France
[2]LACOSI Laboratory University of Sciences Technology MB Oran, Oran, Algeria
[3]IRIMAS Institute University of Haute Alsace, IUT 34 rue du Grillenbreit 68008, Colmar, France

Article Info

Article history:

Received Mar 15, 2025
Revised Oct 16, 2025
Accepted Nov 23, 2025

Keywords:

Chaos
Cryptography
Fitness function
Genetic algorithm
Randomness tests

ABSTRACT

In the dynamic field of artificial intelligence, genetic algorithms (GAs) offer a powerful approach to solving complex problems by mimicking biological mechanisms such as mutation, crossover, and natural selection. Their efficiency relies primarily on the fitness function, which evaluates the quality of candidate solutions and guides the evolutionary process toward an optimal outcome. A well-designed fitness function not only enhances convergence speed but also reduces the risk of stagnation and improves algorithmic accuracy. This paper explores the fundamental role of fitness functions in optimization, machine learning, multi-objective optimization, and cryptography, highlighting their impact on the performance of GAs. We propose a novel fitness function that incorporates the influence of crossover, mutation, and inversion rates on solution quality. This approach, which diverges from conventional models, demonstrates improved convergence behavior and adaptability across different problem domains. The proposed method enhances GA performance not only in secure data encryption but also in general optimization and learning tasks, making it a valuable contribution for both researchers and practitioners, which can open new avenues for research in the development of more robust evolutionary strategies that can adapt effectively to the specific characteristics and challenges of each problem domain.

Corresponding Author:

Hana Ali Pacha
Quartz Laboratory ECAM-EPMI, 13 Bd de l'Hautil
95092 Cergy Pontoise, France
Email: h.ali-pacha@ecam-epmi.com

1. INTRODUCTION

In the ever-evolving field of artificial intelligence, genetic algorithms (GAs) are indispensable tools for efficiently solving complex problems [1], [2]. Their performance largely depends on the accuracy and relevance of their fitness function, a crucial criterion guiding the evolutionary process [3]. This function plays a fundamental role in recent research on optimization and artificial intelligence, particularly in evolutionary algorithms, machine learning, and multi-objective optimization.

a. Optimization in evolutionary algorithms:
   - In GAs and evolutionary strategies (ES), the fitness function evaluates the quality of candidate solutions [4].
   - It guides the selection of the fittest individuals for survival and reproduction, thereby influencing convergence toward an optimal solution [5].

b. Machine learning and neural networks:
   - In deep learning, fitness functions, often in the form of loss functions (cross-entropy and MSE), enable weight adjustments in networks through backpropagation [6].

- In evolutionary neural networks (NEAT and CoDeepNEAT) [7], [8], the fitness function evaluates network architectures and directs their evolution.

c. Multi-objective optimization and heuristics
   - Methods like NSGA-II and MOEA/D use fitness functions [9], [10] to balance multiple conflicting criteria.
   - Normalization and aggregation of criteria pose major challenges, with recent approaches integrating Pareto dominance-based methods and reinforcement learning [11], [12].

d. Cryptography and cybersecurity
   - In cryptanalysis, certain attacks leveraging evolutionary algorithms exploit a fitness function to recover a key or optimize a brute-force attack [13], [14].
   - In steganalysis, it assesses the effectiveness of an algorithm for concealing or detecting hidden data [15], [16].

e. Bioinformatics and molecular design
   - In protein design, evolutionary approaches use the fitness function to identify optimal structures based on biochemical criteria [17].
   - In personalized medicine [18], it is employed to optimize treatment protocols based on patient data.

f. Blockchain and explainable artificial intelligence (AI)
   - In proof-of-work (PoW) systems, a fitness function can be likened to mining difficulty [19].
   - In AI explainability, some research proposes fitness functions to optimize interpretability without compromising performance [20].

Defining and adapting fitness functions [21] is crucial for enhancing the efficiency and relevance of algorithms across various scientific and technological fields. Research in this domain is progressing along several promising directions:

a. Adaptation and personalization of fitness functions
   - Auto-adaptation: dynamically adjusting the weighting of criteria during optimization [22].
   - Fitness shaping [23]: transforming the fitness function to improve convergence and avoid local minima.

b. Machine learning and fitness function generation
   - Using neural networks or Bayesian methods to optimize fitness function definitions [24].
   - Meta-learning approaches where an algorithm evolves to identify the most suitable fitness function for a given task [25].

c. Multi-objective and hybrid functions
   - Developing advanced aggregation methods, such as nonlinear combinations of criteria [26].
   - Adjusting fitness functions to better capture solution diversity based on Pareto dominance [27].

d. Specific applications and custom design
   - In cryptography [28], defining fitness functions tailored to evolutionary algorithm-based attacks.
   - In bioinformatics [29]: designing functions inspired by computational biology for molecular optimization.

e. Information theory and complexity
   - Leveraging entropy measures and algorithmic complexity to enhance the robustness of fitness functions [30].
   - Analyzing optimization landscapes to better understand the relationship between fitness functions and algorithmic efficiency [31].

In this context, we have designed a novel fitness function aimed at improving the performance of GAs. By integrating more advanced evaluation criteria and refining selection mechanisms, this approach enables:

- Faster convergence toward optimal solutions.
- Reduced risk of stagnation in local optima.
- Enhanced accuracy and speed of evolutionary algorithms.

This paper is particularly relevant to researchers and practitioners searching for more adaptive and efficient evolutionary algorithms. By improving convergence and robustness in different domains such as AI, cryptography, and bioinformatics, the proposed method addresses limitations in fitness design and provides practical solutions for specific problems with greater reliability.

2. PRIMITIVE TOOLS

2.1. Genetic algorithm

GAs are a class of global optimization methods [2]. They are particularly effective for finding solutions to optimization problems (i.e., problems where the objective is to minimize or maximize a function's value) where the function to be optimized may be non-differentiable and may exhibit multiple local minima (as seen in some single-variable functions).

The framework implemented by GAs is inspired by Darwin's theory [2], which posits that individuals within a population who are best suited to their environment are more likely to reproduce, thereby producing a new generation that is better adapted than the previous one through the inheritance of traits via genetic code. This concept is computationally realized through the integration of several key components:

- A fitness measure: this measure [3] evaluates the degree of adaptation of each individual to their environment.
- A selection operator: this operator selects the most fit individuals from a population to reproduce. Selection is generally based on the fitness value of individuals, and an individual may be selected multiple times.
- A genome coding scheme: a coding scheme is necessary for the computational representation of an individual or genome, enabling the execution of the reproduction process.
- A reproduction operator: this operator generates new genomes (typically two) from two parent genomes. The reproduction process must ensure the transmission of genetic material from both parents to the offspring. The reproduction operator usually involves two procedures: a crossover procedure, which involves the exchange of genome segments between the two parent genomes, and a mutation procedure, where certain genes in the offspring genomes are randomly altered with a very low probability.
  - Crossover: combines segments from two chromosomes to create offspring that inherit advantageous traits from the parents.
  - Mutation: randomly alters genes to introduce diversity.
  - Inversion: if utilized, it rearranges the genes within a segment to explore new solutions.

2.1.1. Key principles of genetic algorithms

The operation of GAs is based on several fundamental principles [2]. Understanding these principles can help you better grasp how GAs function.

- Population: a set of potential solutions to a particular problem.
- Fitness function: a method for evaluating or scoring each individual in the population.
- Selection: the process of choosing individuals, based on their fitness scores, to reproduce and pass their genes to the next generation.
- Crossover: also known as reproduction. It involves combining the genetic information of two parents to create offspring.
- Mutation: random alterations of certain individuals in the population to maintain and introduce diversity.

2.1.2. Building a genetic algorithm

Constructing a basic GA [32] from scratch involves several systematic steps. The following guide presents these steps in a simple yet comprehensive manner:

- Initialization: begin by randomly generating a population of candidate solutions.
- Fitness evaluation: assess each candidate in the population using a fitness function.
- Selection: based on fitness scores, select the parents who will reproduce to create new individuals for the next generation.
- Crossover or reproduction: combine the genetic information of two parent candidates to create offspring.
- Mutation: randomly alter some genes in the offspring to maintain diversity within the population.
- New generation: replace the old population with the newly created offspring to form a new generation.
- Termination condition: repeat steps 2 to 6 until a termination condition is met, such as finding a solution with sufficiently high fitness or reaching a fixed number of generations.

2.2. Fitness function

A fitness function is a mathematical expression that evaluates the quality of a solution based on the problem's objective [33], [34]. It may be based on a single criterion, such as minimizing costs or maximizing profits, or on multiple criteria, such as balancing efficiency and sustainability. The fitness function must be consistent, scalable, and computable, meaning it should always produce the same output for the same input, handle different sizes and complexities of solutions, and be easy to compute and compare.

In general, the fitness function assesses the quality of potential solutions according to specific criteria. Depending on the nature of the problem and the representation of the solution, various types of fitness functions can be used. However, incorporating the percentages of crossover, mutation, and inversion operators can add an additional dimension to this evaluation.

- Crossover: assesses how the crossover of parents contributes to the performance of the offspring.
- Mutation: examines how mutation affects the diversity and improvement of solutions.
- Inversion: evaluates how inversion impacts the quality of solutions.

There is a variety of sources and contexts in the literature on GAs that demonstrate an even broader range of percentages for the operators. Typical operator percentages in GAs are generally as follows:

- Inversion: typically, less than 1% of the population, as it is a less common and more specialized operation.
- Mutation: usually between 1% and 5%, sometimes up to 10%, to maintain diversity without introducing too much noise.
- Crossover: typically, around 60% to 90%, meaning that most individuals in the population undergo a crossover operation in each generation. This rate may vary depending on the specific problem and the type of crossover used.
  - Single-point crossover: typically, between 70% and 80%, though some algorithms use higher rates, up to 90%.
  - Two-point crossover: rates are often comparable to those of single-point crossover, with values around 70% to 90%.
  - Uniform crossover: this type of crossover can have rates ranging from 60% to 90%, depending on the exploration and exploitation strategy
- Typical mixing (5% to 60%): when you mention a percentage between 5% and 60%, it could refer to a strategy where crossover and mutation rates are adjusted to achieve an optimal balance. Here are some typical scenarios where this mixing might appear:
  - Balanced crossover and mutation rates: in some algorithms, crossover and mutation rates are adjusted to optimize exploration and exploitation. For example, a higher crossover rate (close to 60%) with a lower mutation rate (around 5%) can favor exploiting good solutions found while maintaining moderate genetic diversity.
  - Exploration vs. Exploitation: a higher mutation rate (around 5% to 10%) with a more moderate crossover rate (around 60%) might be used for problems requiring greater exploration of the search space.
  - Dynamic mixing: some algorithms dynamically adjust crossover and mutation rates based on performance or population diversity. This could mean that the crossover and mutation percentages might vary within a range between these values to optimize results.

These percentages can vary depending on the specific problem and the objectives of the GA.

3. METHOD

We designed our fitness function [35] by using, on one hand, the hyperbolic tangent values of chaotic data and, on the other hand, using this data in a specific algorithm.

3.1. Chaotic system used

A chaotic system is a nonlinear, deterministic dynamic system characterized by its unpredictability due to extreme sensitivity to initial conditions. Chaos is defined as the strange and unpredictable behavior of a deterministic dynamic system. The idea behind designing the fitness function using a 5D chaotic system follows [36] (1).

$$\begin{cases} \dot{x} = x + y + yz - au + bw \\ \dot{y} = (y - x)z \\ \dot{z} = b - z - xy \\ \dot{u} = x \\ \dot{w} = cz \end{cases} \quad (1)$$

The system has chaotic behavior for the following parameters: $a = 0.8, b = 0.4,$ and $c = 0.2$. Using the following conditions: $x_0 = 0.1$ $y_0 = 0.13$ $z_0 = 0.2,$ $u_0 = 0.05$ and $w_0 = 0.11$, we obtain three series of Figure 1.

a) The first series illustrates the temporal coordinates of the 5D system in Figure 1, where each of the five coordinates is represented by its respective curve: *x* (Figure 1(a)), *y* (Figure 1(b)), *z* (Figure 1(c)), *u* (Figure 1(d)), and *w* (Figure 1(e)).

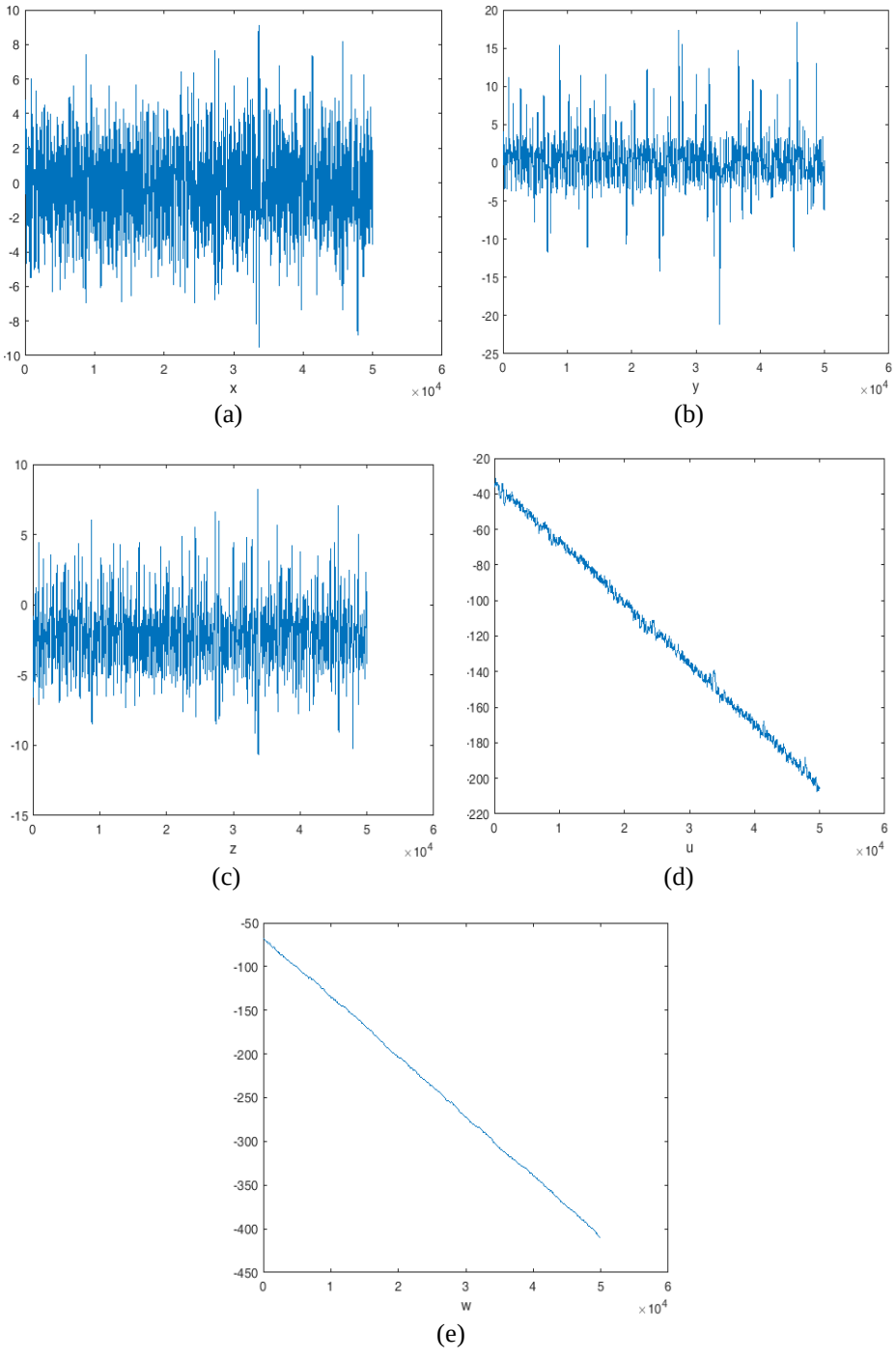
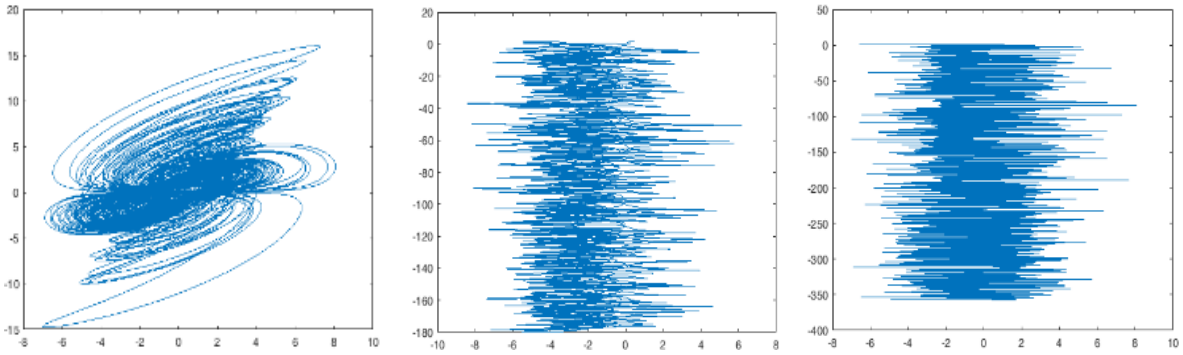


Figure 1. Temporal coordinates of the 5D system: (a) x, (b) y, (c) z, (d) u, and (e) w

In Figure 1, each subfigure shows different shapes, which indicates that the five series produce different results.

b) The second series represents the chaotic system in 2D in Figure 2, focusing on three coordinate plane curves: the plane (x, y) in Figure 2(a), the plane (z, u) in Figure 2(b), and the plane (x, w) in Figure 2(c).

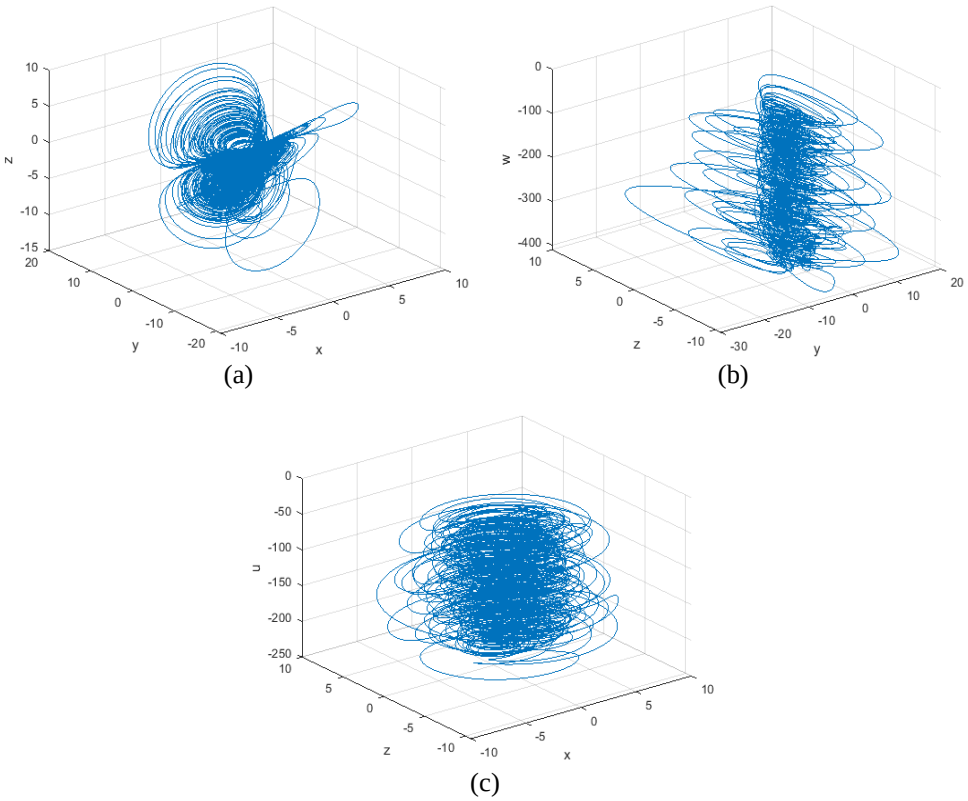


(a) (b) (c)

Figure 2. The 2D representation of the chaotic system; (a) x-y, (b) z-u, and (c) x-w

In Figure 2, each subfigure shows a distinct projection of an attractor with complex and irregular trajectories, which confirms the chaotic behavior of the series in the 2D plane.

c) The third series represents the chaotic system in 3D in Figure 3, highlighting three coordinate space curves: (x, y, z) Figure 3(a), (y, z, w) Figure 3(b), and (x, z, u) Figure 3(c).



(a) (b) (c)

Figure 3. The 3D representation of the chaotic system; (a) x-y-z, (b) y-z-w, and (c) x-z-u

In Figure 3, each subfigure shows a unique shape of an attractor in 3D space, which demonstrates the chaotic behavior of the series in 3D. Figures 1-3, show that the uniqueness of this system lies in the shape of its attractor, and that all five series have chaotic behavior.

3.2. Steps of the fitness function

In the following, we present the different steps of the proposed fitness function:

a. Choose the initial conditions for the 5D hyperchaotic system.
b. The hyperchaotic system generates 5 sequences based on the chosen initial conditions.

c. The variable x is used to create the fitness value.

d. Calculate $y = \text{actanh(x)} = \frac{e^x+e^{-x}}{e^x-e^{-x}}$.

e. The values of the new sequence x modified by arctan (2) are used before being applied in the calculation of the fitness function:

$$y = \text{actanh(x)} = \frac{e^x+e^{-x}}{e^x-e^{-x}} \quad (2)$$

f. Each of the 10 values in the sequence x creates a vector to form N vectors (where N=8192). For each fitness vector (a vector containing 10 values), the following steps are taken:

- Separate the positive values $\boldsymbol{x_p}$ into a vector, and the negative values into another vector $\boldsymbol{x_n}$ (negative values are stored as their absolute values.) (3).

$$\begin{cases} x_p = x_1(i) \;\; if\; x_1(i) > 0 \\ x_n = |x_1(i)| \;\; if\; x_1(i) < 0 \end{cases} \quad i = 1 \ldots, 10 \quad (3)$$

- The values in the vector $\boldsymbol{x_p}$ are divided by their maximum value, and the values in the vector $\boldsymbol{x_n}$ are divided by their maximum value (4).

$$\begin{cases} x_p = x_p/\max(x_p) \\ x_n = x_n/\max(x_n) \end{cases} \quad (4)$$

- The values in the vector $\boldsymbol{x_n}$ are modified according to the following (5):

$$x_n(i) = 1 - x_n(i) \quad (5)$$

- The values of the vectors $\boldsymbol{x_p}$ and $\boldsymbol{x_n}$ are multiplied by 10 (6):

$$\begin{cases} x_p = x_p \times 10 \\ x_n = x_n \times 10 \end{cases} \quad (6)$$

- Add the two vectors to obtain the fitness value (7).

$$f(i) = x_p + x_n \quad (7)$$

This gives us a program written in MATLAB as mentioned in Figure 4:

```
function f=fitn(x)
x=((exp(x))+(exp(-x)))./(((exp(x))-(exp(-x))));
x1=reshape(x',10,[])';
for i=1:size(xx,1)
   x11=x1(i,:);
   for j=1:10
      if(x11(j)>=0)
         xp(j)=x11(j);
         xn(j)=0;
      else
          xn(j)=abs(x11(j));
         xp(j)=0;
      end
   end
   if(max(xp)>0)
      X1p=xp/max(xp);
   else
      X1p=xp;
   end
   if(max(xn)>0)
      X1n=xn/max(xn);
      for j=1:10
         if(X1n(j)>0)
            X1n(j)=1-X1n(j);
         end
      end
   else
      X1n=xn;
   end
      f(i)=sum(X1p)*10+sum(((X1n))*10);
   end
end
```

```
initialConditions=[0.1  0.13  0.2  0.05  0.11];
[X ,Y ,Z ,H ,K ] = chaos(initialConditions );
N=8192;
for j=1:16
 XX=X(((j-1)*(N*10))+1:j*(N*10));
x2=fitn(XX);
x3=sort(x2);
figure;
plot(x3)
end
```

Figure 4. Algorithm of the fitness function

4. RESULTS AND DISCUSSION

4.1. Sensitivity to initial conditions

We test the sensitivity of the chaotic system to initial conditions, specifically for the coordinate x used in generating the fitness function values [37]. Consider the following initial conditions: $x_{01} = 0.1$, $x_{02} = 0.1 + 10^{-15}$, $y_0 = 0.13$ $z_0 = 0.2$ $u_0 = 0.05$ $w_0 = 0.11$. We observe from Figure 5 the sensitivity of the system to initial conditions with negligible change in a single coordinate change.

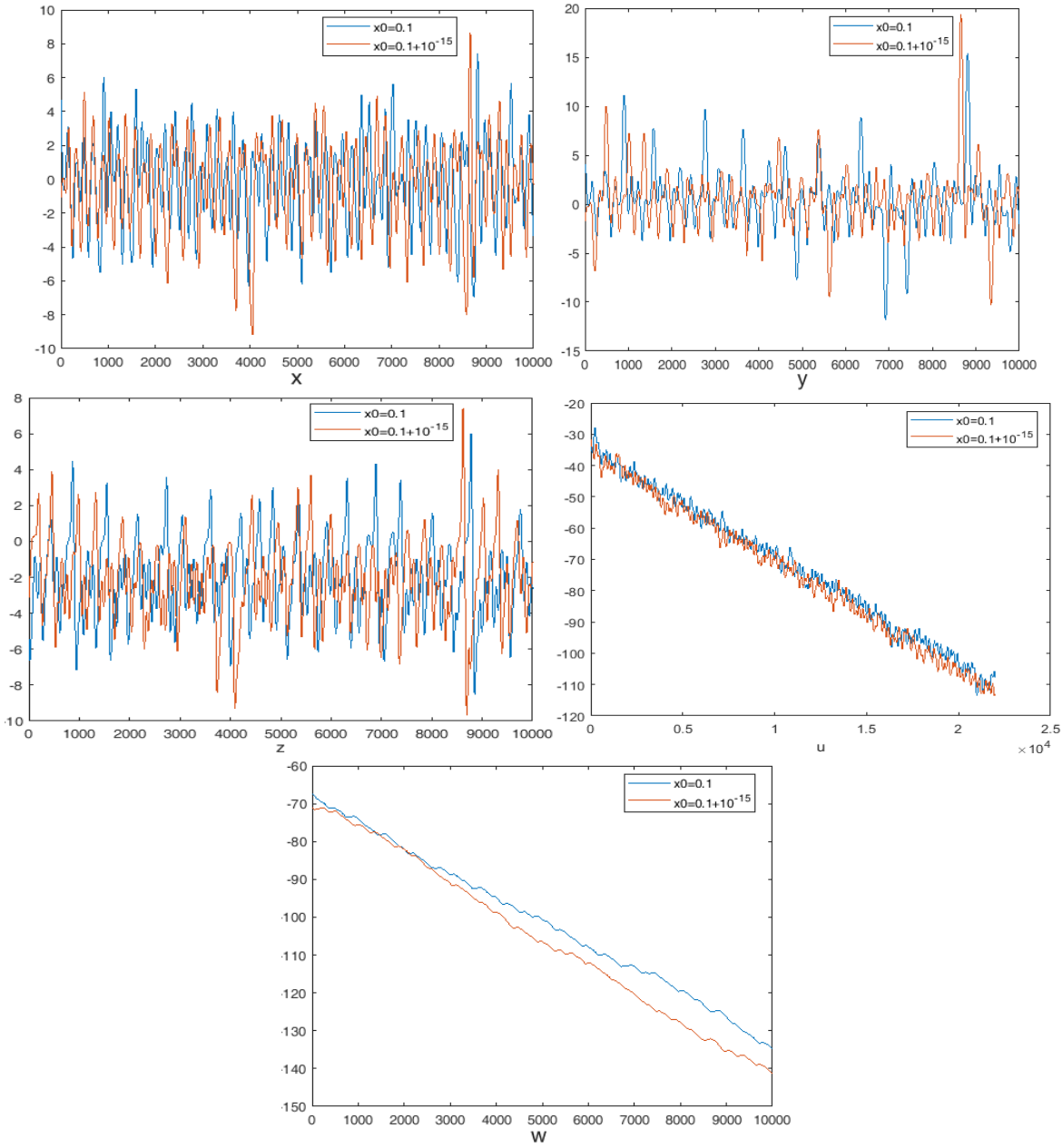


Figure 5. Sensitivity to initial conditions of the chaotic system

4.2. Shape of the fitness function: fitness landscape

The fitness function is often characterized by its shape, as this shape determines how different solutions in a search space are evaluated and compared. The fitness landscape is the graphical representation of the fitness function relative to possible solutions. It can present valleys, peaks, plateaus, etc., which influence the difficulty of finding a global optimum.

Different values of the fitness function were generated and then sorted in ascending order to visualize the shape. In the ten figures that follow (from Figure 6 to Figure 15), each figure represents three graphs for different values of the fitness function:

- The first graph represents the number of values generated successively (f_v) according to their fitness values.
- The middle graph represents the number of values generated successively (f_v)and linked to each other according to their fitness values.
- The third graph represents the values generated, sorted in ascending order: fitness landscape (f).

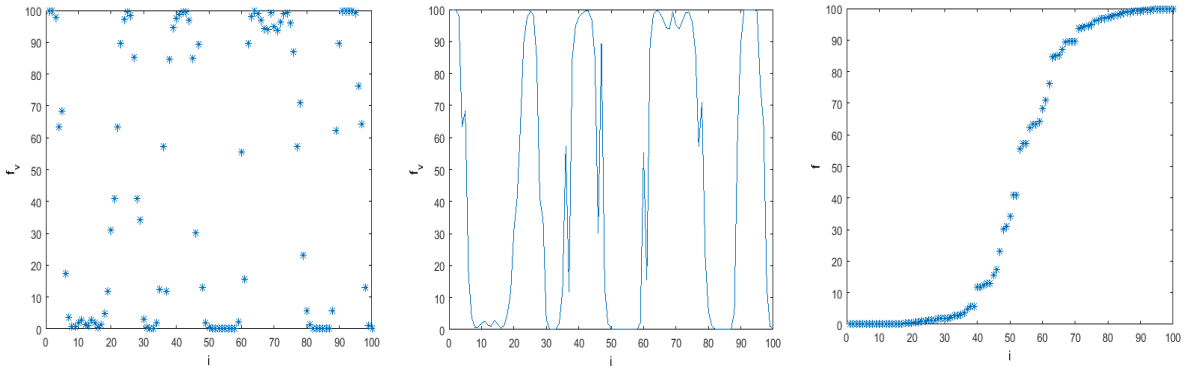


Figure 6. The first 100 values

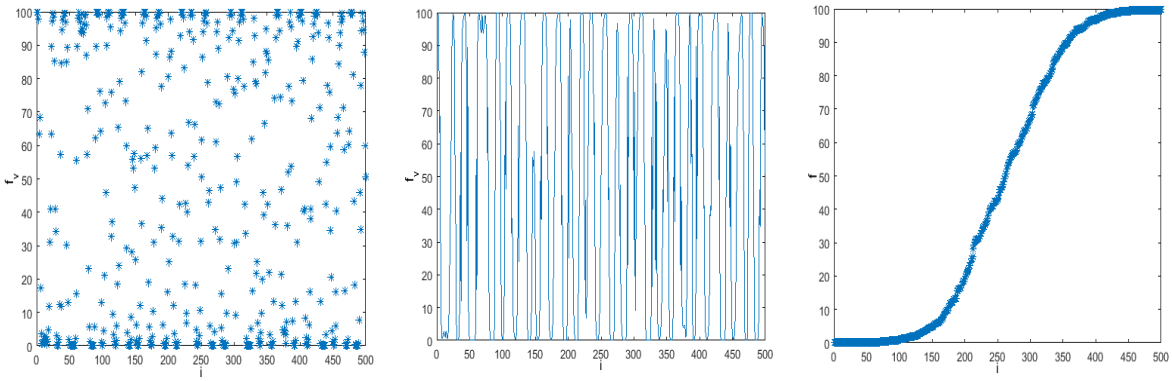


Figure 7. The first 500 values

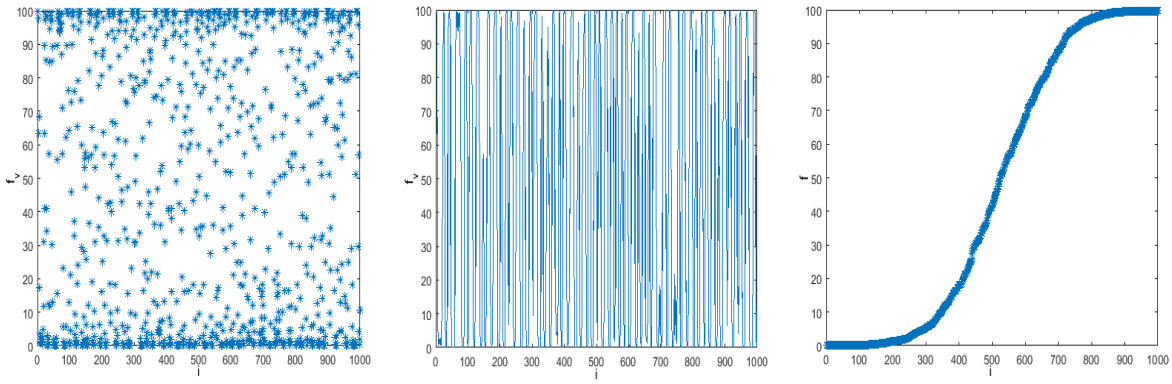


Figure 8. The first 1,000 values

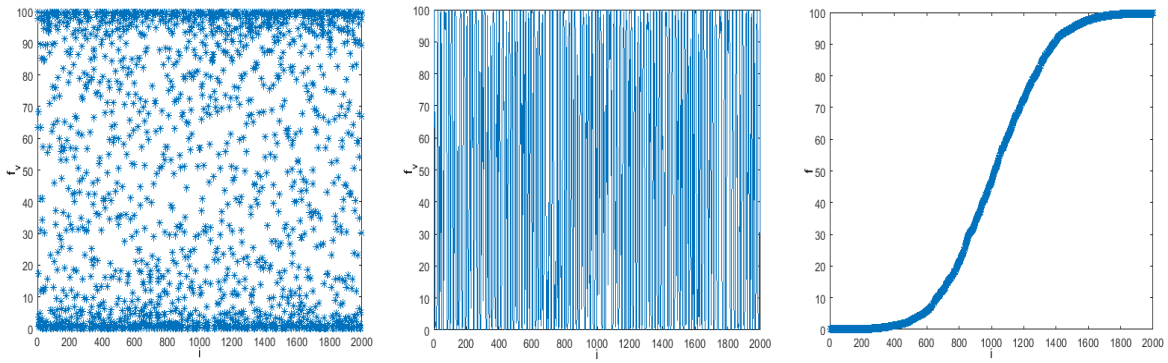


Figure 9. The first 2,000 values

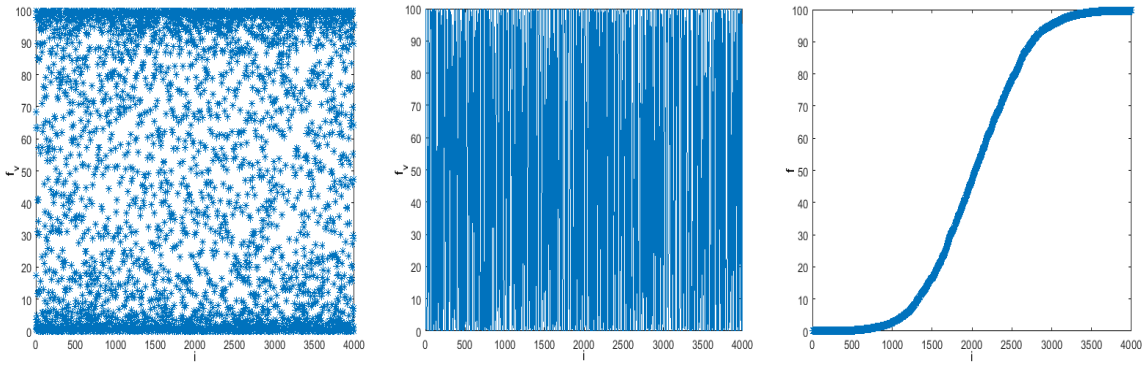


Figure 10. The first 4,000 values

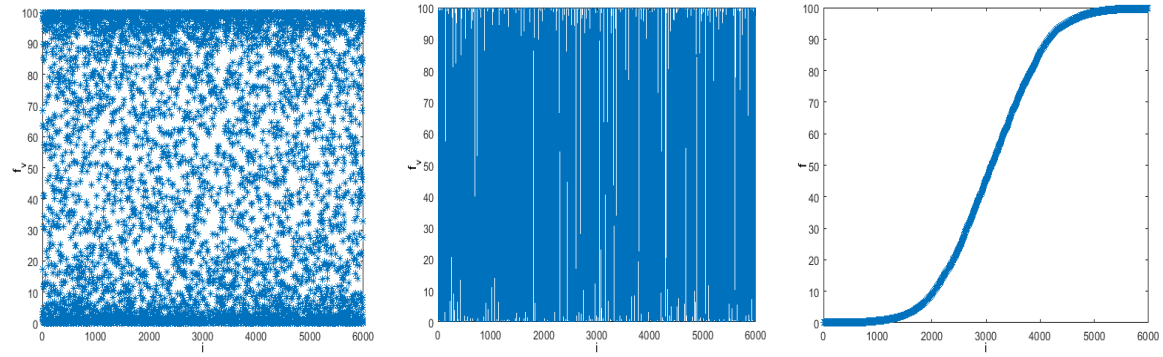


Figure 11. The first 6,000 values

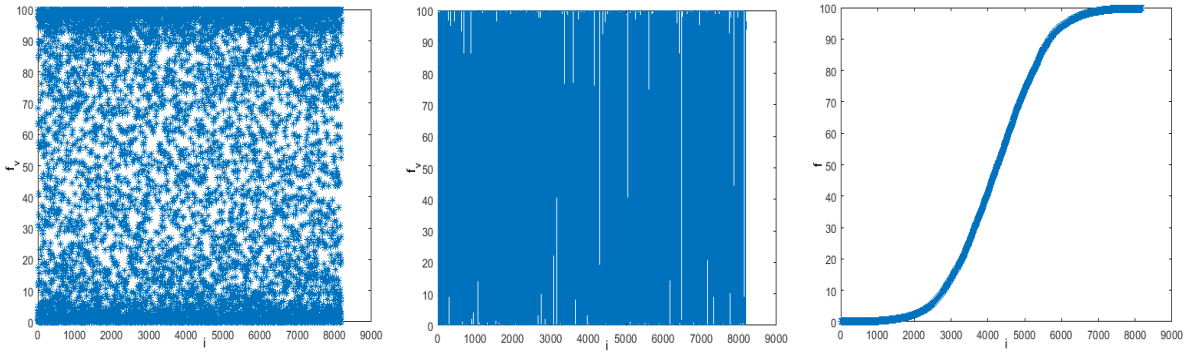


Figure 12. The first 8,192 values

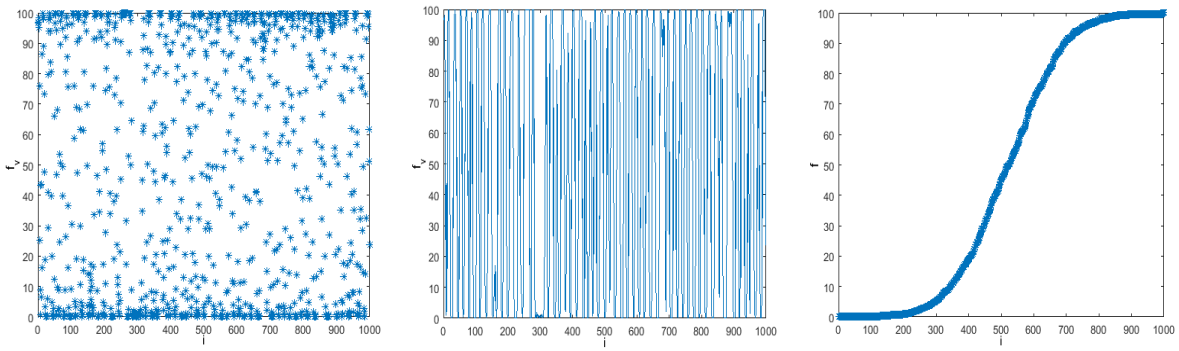


Figure 13. The 1000 values: between 4,000-5,000

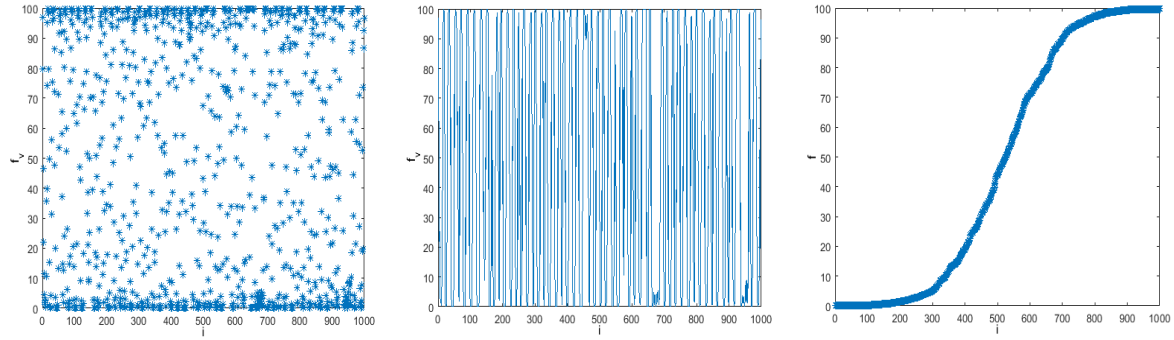


Figure 14. The 1,000 values: between 7,192-8,192

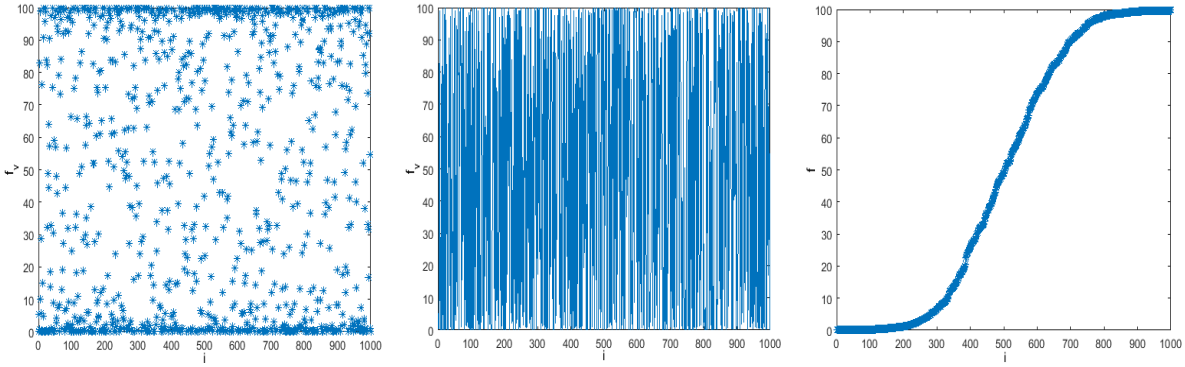


Figure 15. The 1,000 random values between 1-8,192

We can summarize the figures into four categories:

- Figures 6 to 12: we gradually increase the number of generated values from 100 to 8192 and visualize the graphs.
- Figure 13: we visualize the graphs for values generated between 4,000-5,000.
- Figure 14: we visualize the graphs for values generated between 7,192-8,192.
- Figure 15: we visualize the graphs for the 1000 values randomly selected from the 8192 generated values.

The landscape of our fitness function has the shape of a hyperbolic tangent.

4.3. Distribution of fitness function values

To analyse the structure and behaviour of the values generated by the proposed fitness function, we performed the process over 16 independent generations, each producing 8,192 values. Each generation was analysed according to five percentage intervals:

- < 1%
- 1% – 5%

– 5% – 60%
– 60% – 90%
– > 90%

These intervals were chosen to reflect the statistical distribution of the function's chaotic behaviour and to evaluate the density of extreme, moderate, and intermediate values. The detailed results for each generation are presented in Tables A1 to A16. Each table indicates the number of values falling within each interval. This level of granularity allows for a generation-by-generation examination and helps verify the stability of the model. To improve the readability of the results in the main body of the article, we provide below a synthesis of the data obtained from the 16 generations. Table 1 presents the average and approximate standard deviation of the number of values observed in each interval, along with their mean proportions. To further understand how the fitness values evolve across generations and intervals, we analyze the distribution per range for each sequence. Figure 16 presents a comparative visualization that captures the variation of values across the five percentage ranges over the 16 generations.

Table 1. Average distribution of fitness values over 16 generations

Interval (%)	Average (Number of values)	Approx. Std. Dev.	Mean percentage (%)
|< 1	1,553	± 28	18.96
1 – 5	835	± 28	10.19
5 – 60	2,113	± 32	25.80
60 – 90	1,142	± 21	13.94
> 90	2,546	± 25	31.10

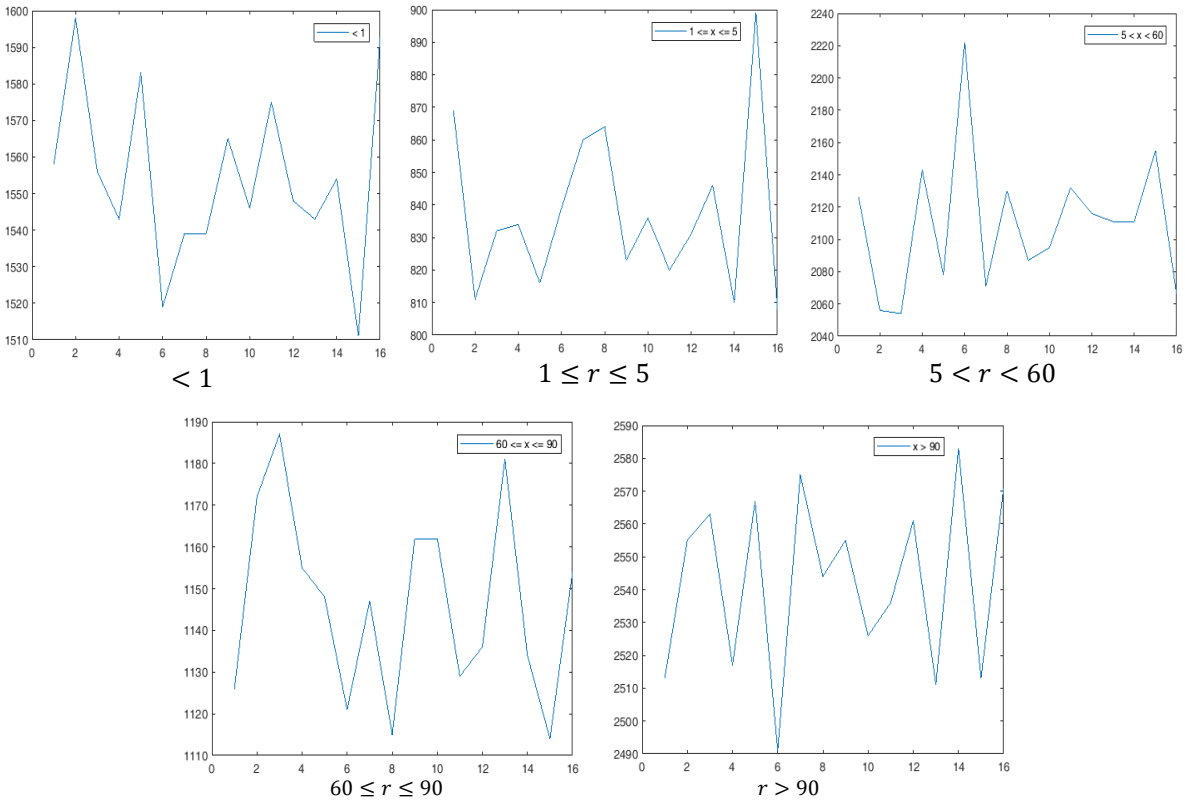


Figure 16. Variation of values for different ranges

From Tables 1-16 and Figure 16, it can be observed that the values in different ranges vary for each sequence. We observe that the distribution of fitness values for each sequence differs from that of other sequences, demonstrating the random behavior of the values generated by the proposed algorithm.

Figure 17 this pie chart visually represents the average distribution of fitness values across the five defined intervals. The predominance of extreme values (< 1% and > 90%) confirms the chaotic nature of the distribution, as previously observed in Table 1.

From Table 1, it can be observed that most values are concentrated at the two extremes of the distribution:

- 18.96% of the values are below 1%,
- while 31.10% exceed 90%.

This distribution is characteristic of a well-exploited chaotic behaviour, where the algorithm intensely explores extreme regions while maintaining density in intermediate zones (notably between 5% and 60%, which account for 25.80% of the values). These results suggest that the fitness function:

- Generates highly diverse populations.
- Is strongly sensitive to initial conditions.
- Successfully captures both promising solutions (values > 90%) and low-performing regions (values < 1%) necessary for maintaining global exploration.

In practice, this distribution promotes:

- The search for optimal solutions.
- The avoidance of local optima trapping.
- A strong generalization capacity of the model in chaotic contexts (e.g., cryptography and multi-objective optimization).

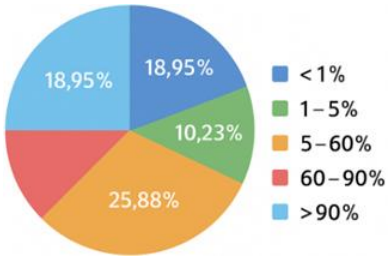


Figure 17. Proportional distribution of fitness values per range over 16 generations (in %)

To avoid overloading the main section, the full detailed data for the 16 generations are provided in Tables A1 to A16. This allows interested readers to examine each generation individually without compromising the clarity of the article.

To complement the analysis presented in Section 4.3, this appendix provides the full detailed results for each of the 16 independent generations tested. Each generation produced 8192 fitness values. These values were categorized into five percentage intervals: (< 1%); (1% – 5%;) (5% – 60%); (60% – 90%) and (> 90%).

Tables A1 through A16 present the number of values in each interval for Generations 1 to 16, respectively. These tables offer fine-grained insight into the variation and consistency of the fitness value distribution across generations. This level of detail is particularly useful for readers interested in the behaviour and stability of the proposed chaotic fitness function over time.

Table A1. First generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1558	869	2126			1126	2513
			$5 < r \leq 16.56$	$16.56 < r \leq 35.75$	$35.75 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.65$		$19.65 < r \leq 60$		
			40%		60%		
	Minimum value: 2.26287×10^{-7}				Maximum value: 99.999999859933		

Table A2. Second generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1598	811	2056			1172	2555
			$5 < r \leq 16.40$	$16.40 < r \leq 35.85$	$35.85 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.73$		$19.73 < r \leq 60$		
			40%		60%		
	Minimum value: 8.358124×10^{-8}				Maximum value: 99.99999989508		

Table A3. Third generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1556	832	2054			1187	2563
			$5 < r \leq 16.55$	$16.55 < r \leq 35.03$	$35.03 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 20.06$		$20.06 < r \leq 60$		
			40%		60%		
		Minimum value: 4.523204×10^{-6}			Maximum value: 99.99998557983		

Table A4. Fourth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1543	834	2143			1155	2517
			$5 < r \leq 15.97$	$15.97 < r \leq 34.60$	$34.60 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.33$		$19.33 < r \leq 60$		
			40%		60%		
		Minimum value: $1.12323790 \times 10^{-6}$			Maximum value: 99.999974719915		

Table A5. Fifth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1583	816	2078			1148	2567
			$5 < r \leq 16.21$	$16.21 < r \leq 35.38$	$35.38 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.73$		$19.73 < r \leq 60$		
			40%		60%		
		Minimum value: $1.085034193 \times 10^{-6}$			Maximum value: 99.9999987504		

Table A6. Sixth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1519	839	2222			1121	2491
			$5 < r \leq 15.11$	$15.11 < r \leq 35.59$	$35.59 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.01$		$19.01 < r \leq 60$		
			40%		60%		
		Minimum value: $1.450138278 \times 10^{-5}$			Maximum value: 99.9999989166		

Table A7. Seventh generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1539	860	2071			1147	2575
			$5 < r \leq 16.27$	$16.27 < r \leq 35.66$	$35.66 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.68$		$19.68 < r \leq 60$		
			40%		60%		
		Minimum value: $2.6272518161 \times 10^{-7}$			Maximum value: 99.999959271		

Table A8. Eighth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1539	864	2130			1115	2544
			$5 < r \leq 15.39$	$15.39 < r \leq 34.43$	$34.43 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 18.73$		$18.73 < r \leq 60$		
			40%		60%		
		Minimum value: $5.5533067033 \times 10^{-9}$			Maximum value: 99.99999998573		

Table A9. Ninth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1565	823	2087			1162	2555
			$5 < r \leq 16.38$	$16.38 < r \leq 35.54$	$35.54 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.67$		$19.67 < r \leq 60$		
			40%		60%		
		Minimum value: $6.2743631645 \times 10^{-6}$			Maximum value: 99.99999959117		

Table A10. Tenth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1546	836	2095			1162	2526
			$5 < r \leq 15.99$	$15.99 < r \leq 35.24$	$35.24 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.70$		$19.70 < r \leq 60$		
			40%		60%		
	Minimum value: $3.9553095931 \times 10^{-6}$				Maximum value: 99.99999400792		

Table A11. Eleventh generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1575	820	2132			1129	2536
			$5 < r \leq 15.88$	$15.88 < r \leq 34.68$	$34.68 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.35$		$19.35 < r \leq 60$		
			40%		60%		
	Minimum value: $2.8225590753 \times 10^{-8}$				Maximum value: 99.9999997424		

Table A12. Twelfth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1548	831	2116			1136	2561
			$5 < r \leq 16.03$	$16.03 < r \leq 35.41$	$35.41 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.85$		$19.85 < r \leq 60$		
			40%		60%		
	Minimum value: $2.070461802006 \times 10^{-8}$				Maximum value: 99.999988916694		

Table A13. Thirteenth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1543	846	2111			1181	2511
			$5 < r \leq 15.29$	$15.29 < r \leq 33.66$	$33.66 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 18.65$		$18.65 < r \leq 60$		
			40%		60%		
	Minimum value: $8.16686406057 \times 10^{-6}$				Maximum value: 99.999994581067		

Table A14. Fourteenth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1554	810	2111			1134	2583
			$5 < r \leq 16.80$	$16.80 < r \leq 36.22$	$36.22 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 20.83$		$20.83 < r \leq 60$		
			40%		60%		
	Minimum value: $1.04604445105 \times 10^{-7}$				Maximum value: 99.99999879164		

Table A15. Fifteenth generation of 8,192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1511	899	2155			1114	2513
			$5 < r \leq 15.85$	$15.85 < r \leq 34.78$	$34.78 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.19$		$19.19 < r \leq 60$		
			40%		60%		
	Minimum value: $2.83692197511 \times 10^{-6}$				Maximum value: 99.99999697049		

Table A16. Sixteenth generation of 8192 values

Interval	< 1	$1 \leq r \leq 5$	$5 < r < 60$			$60 \leq r \leq 90$	$r > 90$
Number	1593	808	2066			1154	2571
			$5 < r \leq 16.34$	$16.34 < r \leq 34.33$	$34.33 < r < 60$		
			33%	33%	34%		
			$5 < r \leq 19.77$		$19.77 < r \leq 60$		
			40%		60%		
	Minimum value: $5.74361863625 \times 10^{-7}$				Maximum value: 99.9999890109		

These detailed distributions confirm the statistical consistency of the proposed fitness function across multiple generations. The persistent presence of extreme and intermediate values reflects the function's ability to preserve diversity and maintain chaotic dynamics throughout the optimization process. This supports the conclusions drawn in section 4.3 regarding the robustness and exploratory capacity of the Hadj-Said fitness function.

4.4. Randomness of values from the fitness function

We will examine the random behaviour of the values generated by the proposed algorithm by generating a sequence of 65,536 values and testing the randomness of these values.

4.4.1. Mean, variance, and autocorrelation factor test

This involves calculating the following factors: mean, variance, and autocorrelation factor of the generated value sequence [37]. Given that x_i, for i = 1…n, is a sequence uniformly distributed in the interval [0,1], we compare it to a sequence generated by a random data generator for which the calculation of the following factors: mean, variance, and autocorrelation factor are known. The results are given in Table 2 and Figure 18. We find that the data sequence satisfies only the mean test.

Table 2. Mean, variance, and autocorrelation factor results

Nature	Equation	Ideal value	Found value	(%) of ideal value
Average	$\mu = \frac{1}{n}\sum_{i=1}^{n} u_i$	0.5	0.4843	< 96.86
Variance	$v = \frac{1}{n}\sum_{i=1}^{n} u_i^2 - \mu^2$	0.0833	0.1713	> 48.64
Autocorrelation	$r = \frac{1}{n}\sum_{i=1}^{n=1} u_i \times u_{i+1}$	0.25	0.3777	> 66.19

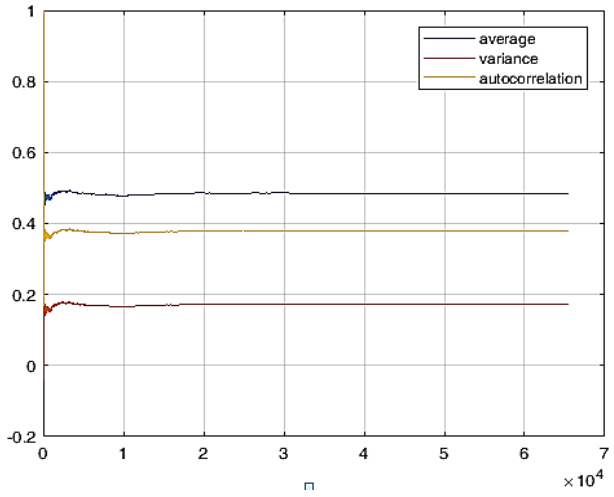


Figure 18. Mean, variance, and autocorrelation factor results

4.4.2. Spectral test

The spectral test is described as the most discriminating of all tests [37]. Indeed, no poor generator has been able to pass it. Very simple, the method involves visually examining the distribution of generated values in a k-dimensional space (2D or 3D) to assess quality. Figure 19 illustrates the spectral test results of the sequence in both 2D and 3D spaces:

- 2D representation: two consecutive values will be the coordinates of a point on the plane. In Figure 19, left, we check if the points are uniformly distributed within a square.
- 3D representation: three consecutive values will be the coordinates of a point in space. In Figure 19 right, we check if the points are uniformly distributed within a cube.

From Figure 19, it is evident that many points in both the 2D and 3D planes are not assigned. This implies that we cannot make a definitive statement about the pseudorandom behavior of the data generated by the proposed algorithm.

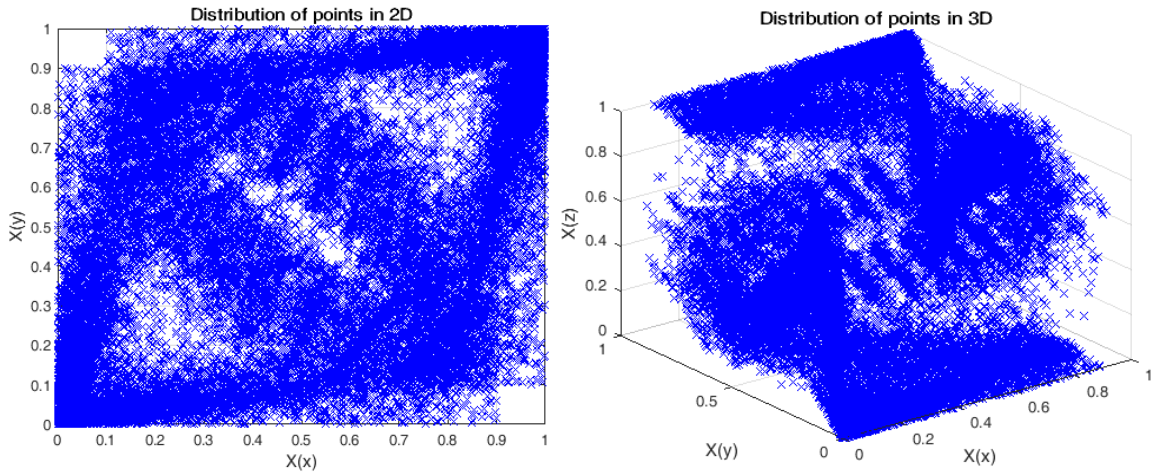


Figure 19. The 2D and 3D spectral test

4.4.3. Entropy test

Entropy [37] measures the amount of information contained in the sequence of data generated by the proposed algorithm:

$$X_i = mod\ (x_i \times 10^7, 256), \text{ for } i = 1 \ldots n, \quad (8)$$

The sequence (8) is considered as a series of 8-bit words. Entropy is calculated as follows (9):

$$H(X) = -\sum_{i=0}^{2^n-1} P_i(X) log_2(P_i(X)) \quad (9)$$

where X is the studied source, and Pi is the probability of occurrence of word i of n bits. The calculation of entropy represents the minimum number of bits per word required to contain the information. We consider a source with an alphabet of 256 characters. If all these characters are equiprobable, the entropy associated with each character is $\log_2(256) = \log_2(2^8) = 8$ bits, which means 8 bits are needed to transmit a character; thus, its entropy is 8 bits. The entropy result of our sequence is given in Table 3. According to Table 3, the entropy of the data delivered by our algorithm is 83.91% of that of an ideal source.

Table 3. Entropy results

Nature source	Entropy per bit/symbol	Found value	(%) of ideal value
Source providing equiprobable characters: ideal source	8	6.71283	83.91

5. CONCLUSION

The fitness function is a fundamental concept in GAs, serving to guide the evolution of solutions. A thorough understanding of the fitness function and its implications can greatly enhance the efficiency and performance of GAs. Its design and calculation must be carefully tailored to the specifics of the problem, considering the objectives, constraints, and how solutions are evaluated and compared. We have created a fitness function well-suited to the specifics of the problem we defined (a cryptosystem based on genetic algorithms), integrating constraints correctly and being tested and refined to achieve the best possible results (perfect data encryption). For a sequence of data in the fitness function, we determined the weights for each operator (inversion, mutation, crossover), and the weights generated as percentages for each operator are almost identical to those found in the literature. The significance of the proposed method integrates the genetic operator influences into the fitness function, which improves convergence and solution quality in

cryptographic contexts. This makes it particularly relevant for researchers developing secure and efficient encryption methods using evolutionary computation. In the cryptosystem, the key experiment to evaluate the proposed fitness function was to study the randomness of the output sequences, which is a critical property for ensuring unpredictability and security. The results demonstrate that the proposed method has strong random behavior.

The proposed algorithm, which features a fitness function with a hyperbolic tangent-shaped landscape, is named "Hadj-Said Fitness Function" for ease of reference. For future research, it can focus on adapting the proposed fitness function to different types of genetic models, such as hybrid or co-evolutionary algorithms. In addition, it can be integrated into explainable AI or combined with neuro- GAs to create new, efficient learning systems.

ACKNOWLEDGMENTS

Thanks to the support MSC2020: 37M05, 65P40, 68T05, 68T20, 90C59. We appreciate the motivation to conduct this collaborative work from institutions: Quartz Laboratory ECAM-EPMI, Cergy, France; LACOSI Laboratory, University of Sciences and Technologies Oran, Algeria; and IRIMAS Institute, University of Haute Alsace, France.

FUNDING INFORMATION

The authors did not receive support from any organization for the submitted work.

AUTHOR CONTRIBUTIONS STATEMENT

The paper has been written and revised by all authors. First Author written first draft and run the testing. Second Author supervised the work and checked its validity. Third author criticised its scientific contribution, remarked its discussion coverage and represented the corresponding author for the work to be published. Fourth Author revised the contribution merit and its influencing in write-up as well as noted the work main body overall representation completeness.

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Hana Ali Pacha	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Abderrahmene Hadj Brahim			✓	✓		✓					✓			
Pascal Lorenz		✓		✓				✓		✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data sharing is not applicable to this article as no modified datasets were generated or analysed during the current study.

REFERENCES

[1] J. H. Holland, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence," *Cambridge, MA, USA: MIT Press*, 1992.
[2] M. Mitchell, "An introduction to genetic algorithms. Cambridge," *Cambridge, MA, USA: MIT Press*, 1996.
[3] K. Deb, "Multi-objective optimization using evolutionary algorithms," *Chichester, U.K.: Wiley*, 2001.
[4] F. Hoffmeister and T. Bäck, "Genetic algorithms and evolution strategies: similarities and differences," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 496 LNCS, Berlin/Heidelberg: Springer-Verlag, 1991, pp. 455–469.

[5] J. L. Breeden, "GA-optimal fitness functions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1447, 1998, pp. 95–102.
[6] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700 LECTURE NO, 2012, pp. 9–48.
[7] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, Jun. 2002, doi: 10.1162/106365602320169811.
[8] J. Bossek, C. Grimme, and F. Neumann, "On the benefits of biased edge-exchange mutation for the multi-criteria spanning tree problem," in *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, Jul. 2019, pp. 516–523, doi: 10.1145/3321707.3321818.
[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.
[10] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.
[11] M. Li, M. López-Ibáñez, and X. Yao, "Multi-objective archiving," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 3, pp. 696–717, Jun. 2024, doi: 10.1109/TEVC.2023.3314152.
[12] A. Navon, A. Shamsian, E. Fetaya, and G. Chechik, "Learning the pareto front with hypernetworks," *ICLR 2021 - 9th International Conference on Learning Representations*, 2021.
[13] T. Mekhaznia, "Analyse cryptographique par les methodes heuristiques," Univ. M'sila, M'sila, 2017.
[14] A. Souici, H. Seridi, and A. M. Alimi, "Evolutionary algorithms for the development of a new image encryption algorithm OIEEA (occurrences-based images evolutionary encryption algorithm)," *International Journal of Computer Science and Applications*, vol. 12, no. 2, pp. 45–58, 2015, [Online]. Available: https://www.researchgate.net/publication/340741950.
[15] R. Cogranne, "Détection statistique d'informations cachées dans une image numérique," *Instituto Nacional de Telecomunicações*, 2012.
[16] S. Kouider, "Steganography and steganalysis, lecture notes," University of Montpellier, 2014.
[17] J. K. Leman *et al.*, "Macromolecular modeling and design in Rosetta: recent methods and frameworks," *Nature Methods*, vol. 17, no. 7, pp. 665–680, Jul. 2020, doi: 10.1038/s41592-020-0848-2.
[18] A. S. Darwich *et al.*, "Annual review of pharmacology and toxicology: model-informed precision dosing: background, requirements, validation, implementation, and forward trajectory of individualizing drug therapy," *Annual Review of Pharmacology and Toxicology*, vol. 61, no. 1, pp. 361–3621, Jan. 2021, doi: 10.1146/annurev-pharmtox-033020-113257.
[19] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Bitcoin*, 2008. https://bitcoin.org/bitcoin.pdf.
[20] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, vol. 13-17-August-2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
[21] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.
[22] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, Mar. 2011, doi: 10.1016/j.swevo.2011.02.001.
[23] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural evolution strategies," in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, Jun. 2008, pp. 3381–3387, doi: 10.1109/CEC.2008.4631255.
[24] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, Jan. 2005, doi: 10.1007/s00500-003-0328-5.
[25] J. Schmidhuber, "Evolutionary principles in self-referential learning. On learning now to learn: the meta-meta-meta...-hook," Technische Universität München, 1987.
[26] I. N. Egorov, "Indirect optimization on the basis of self-organizatione," *Air Force Eng Acad*, vol. 2, pp. 683–691, 1998.
[27] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, Sep. 1994, doi: 10.1162/evco.1994.2.3.221.
[28] E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis, "Evolutionary computation based cryptanalysis: a first study," *Nonlinear Analysis, Theory, Methods and Applications*, vol. 63, no. 5–7, pp. e823–e830, Nov. 2005, doi: 10.1016/j.na.2005.03.004.
[29] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger, "Barrier trees of degenerate landscapes," *Zeitschrift fur Physikalische Chemie*, vol. 216, no. 2, pp. 155–173, Jan. 2002, doi: 10.1524/zpch.2002.216.2.155.
[30] M. Mitchell, *An Introduction to Genetic Algorithms*, 7th ed. Cambridge: MIT Press, 2001.
[31] S. Kauffman and S. Levin, "Towards a general theory of adaptive walks on rugged landscapes," *Journal of Theoretical Biology*, vol. 128, no. 1, pp. 11–45, Sep. 1987, doi: 10.1016/S0022-5193(87)80029-2.
[32] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun. 1994, doi: 10.1007/BF00175354.
[33] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.
[34] Y. Pei *et al.*, "Dynamic algorithm for fitness function greatly improves the optimization efficiency of frequency selective surface for better design of radar," *Scientific Reports*, vol. 12, no. 1, p. 16596, Oct. 2022, doi: 10.1038/s41598-022-20167-x.
[35] AI Stack Exchange, "How to create a good fitness function?," *AI Stack Exchange*, 2024. https://ai.stackexchange.com/questions/9105/how-to-create-a-good-fitness-function (accessed Feb. 20, 2025).
[36] S. Vaidyanathan, A. Sambas, A. T. Azar, K. P. S. Rana, and V. Kumar, "A new 5-D hyperchaotic four-wing system with multistability and hidden attractor, its backstepping control, and circuit simulation," in *Backstepping Control of Nonlinear Dynamical Systems*, Elsevier, 2020, pp. 115–138.
[37] A. P. Hana, H. S. Naima, and A. P. Adda, "Image encryption by using a specific adaptation of Lehmer's algorithm," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 5, pp. 949–971, Jul. 2020, doi: 10.1080/09720529.2019.1652402.

BIOGRAPHIES OF AUTHORS

Hana Ali Pacha is a lecturer and researcher at ECAM-EPMI. She holds a Ph.D. in Telecommunications, specializing in cryptography and data security, from the University of Science and Technology of Oran Mohamed-Boudiaf (USTO-MB). Her research focuses on developing new encryption systems incorporating confusion and diffusion techniques to enhance the security of digital infrastructures. She is particularly interested in information system security, blockchain, and data traceability, as well as the integration of cryptographic systems into distributed architectures and cloud computing. She also contributes to collaborative research projects aimed at improving secret sharing protocols and enhancing system resilience against cyber threats. She can be contacted at email: h.ali-pacha@ecam-epmi.com.

Abderrahmene Hadj Brahim is an assistant professor at the University of Science and Technology of Oran Mohamed-Boudiaf (USTO-MB). He holds a Ph.D. in Telecommunications, specializing in network security, from USTO-MB. His research focuses on developing advanced encryption-compression systems to enhance both the security and efficiency of digital data storage and transmission. His areas of interest include information system security, data compression, and the development of random number generators. Additionally, he is a member of the Laboratory of Coding and Information Security at USTO-MB. He can be contacted at email: abderrahmene.hadjbrahim@univ-usto.dz.

Pascal Lorenz is a received his M.Sc. (1990) and Ph.D. (1994) from the University of Nancy, France. He has been a professor at the University of Haute-Alsace since 1995. His research focuses on Quality of Service (QoS), wireless networks, and high-speed networks. He is the author or co-author of three books, three patents, and over 200 international publications in refereed journals and conferences. He has served as Technical Editor for IEEE Communications Magazine (2000-2006), IEEE Networks Magazine (2015-2020), IEEE Transactions on Vehicular Technology (since 2017), and IEEE Internet of Things Journal (since 2024). He held leadership roles in IEEE ComSoc France (Chair, 2020-2024), IEEE France (Financial Chair, 2017-2022), and several IEEE technical committees. Additionally, he has been actively involved in organizing major IEEE conferences such as ICC, Globecom, and WCNC. Prof. Lorenz is a senior IEEE member, an IARIA fellow, and serves on editorial boards of multiple journals. He has chaired numerous technical sessions, delivered tutorials at major international conferences, and was an IEEE ComSoc Distinguished Lecturer (2013-2014). He can be contacted at email: pascal.lorenz@uha.fr.