Received 25 December 2025, accepted 16 March 2026, date of publication 31 March 2026
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.25.1.4498

Neuromesh Universal Genetic Optimizer in Typical Business Decision Algorithms

ANDRII SHKITOV[1], PETRO TALANCHUK[1], OLEKSII ZIVENKO[2], ANATOLII TYMOSHENKO[1], VOLODYMYR PAVLENKO[1], VITALIA KROPYVNYTSKA[3]

[1]Department of Computer Engineering, Open International University of Human Development "Ukraine", 04071 Kyiv, Ukraine
[2]Marine Instrumentation Department, Admiral Makarov National University of Shipbuilding, 54025 Mykolaiv, Ukraine
[3]Department of Computer Systems and Networks, Ivano-Frankivsk National Technical Oil and Gas University, Ivano-Frankivsk, Ukraine

Corresponding author: Andrii Shkitov (e-mail: opncore@gmail.com)

ABSTRACT This research presents the development of a universal genetic optimizer (UGO) aimed at solving optimization problems across a wide range of test functions, focusing on achieving reliable global extrema for both theoretical and practical applications. The study utilizes a genetic algorithm (GA) enhanced with neural network-based approaches, implementing key genetic operators such as crossover, mutation, and elitism. The algorithm was tested on benchmark functions including Rosenbrock, De Jong, and Griewank, among others, with statistical analysis identifying optimal parameter settings. The results demonstrate superior performance compared to traditional tools like Excel Solver and GAMS, particularly when elitism is included. The proposed framework highlights the adaptability of evolutionary computation when combined with machine learning techniques. Moreover, it opens perspectives for applying UGO in real-world optimization challenges such as logistics, energy systems, and engineering design.

KEYWORDS genetic algorithm; universal optimizer; neural networks; optimization; business decisions; elitism; benchmark functions; parameter tuning.

I. INTRODUCTION

Artificial intelligence (AI) and its associated tools are rapidly becoming an indispensable component of the modern digital ecosystem, evolving from a specialized technology into a fundamental layer of digital infrastructure [1–3]. Its ability to process large volumes of data, identify complex nonlinear patterns, and generate informed decisions in real time without direct human intervention is fundamentally transforming numerous technological and economic processes. These capabilities enable the automation of highly complex tasks and significantly improve operational efficiency across a wide range of industries, including medical diagnostics, pharmaceutical research, logistics, and high-frequency financial trading.

Through advanced methodologies such as machine learning (ML) and deep learning (DL), intelligent systems exhibit a high degree of adaptability. They are capable of continuous learning, adjusting to new operational conditions, and progressively improving their performance. As a result, such systems can effectively perform tasks that traditionally required sophisticated analytical reasoning or complex predictive modeling [4, 5].

At the same time, the application of genetic algorithms (GAs) continues to expand both theoretically and practically. As a prominent class of evolutionary computation methods, GAs have demonstrated strong performance in solving global optimization problems, particularly in cases where conventional optimization techniques are limited by the complexity of the search space, discrete parameter structures, or the absence of an analytical formulation of the problem.

Although genetic algorithms are heuristic in nature and therefore cannot guarantee the discovery of a true global optimum in problems lacking a proven analytical solution, their empirical robustness across a wide range of benchmark functions has consistently demonstrated their effectiveness in addressing complex real-world optimization tasks. This observation highlights the importance of developing a universal genetic optimizer capable of providing reliable optimization performance across diverse problem domains. Such an approach is not only scientifically relevant but also practically valuable, as evolutionary optimization methods often demonstrate greater robustness when applied to non-smooth or multimodal objective functions compared with classical gradient-based techniques implemented in widely used optimization tools such as Excel Solver or GAMS.

A. LITERATURE REVIEW

The body of academic literature devoted to genetic algorithms

is extensive and continues to grow rapidly. In addition to peer-reviewed publications, numerous online resources and technical repositories dedicated to evolutionary computation have emerged. In this study, the sources presented in [6–8], together with several specialized online materials, were used to select benchmark functions for experimental evaluation and to validate the proposed optimization model.

The study presented in [9] introduces NeuroMesh, an innovative distributed AI learning platform designed to utilize decentralized computational infrastructure for scalable model training. The NeuroMesh architecture enables efficient utilization of distributed computing resources, which is particularly important for large-scale business applications.

The work in [10] proposes a hybrid decision-support framework that integrates neural networks with genetic algorithms for complex business decision-making, particularly in the context of investment analysis. The proposed system demonstrates adaptive learning capabilities, making it a promising tool for forecasting financial market dynamics. In [11], the authors investigate the application of soft computing techniques, including genetic algorithms, in combination with other computational approaches for improving business analytics processes. Special attention is given to the optimization of marketing strategies with the objective of maximizing return on investment (ROI). The study in [12] explores the use of genetic algorithms for optimizing business processes such as supply chain management and financial modeling. By simulating mechanisms of natural selection, genetic algorithms enable efficient exploration of complex search spaces in order to identify high-quality solutions. Research presented in [13] focuses on the application of genetic algorithms to optimize neural network architectures. This approach enables the automated discovery of efficient network configurations for specific tasks, thereby improving model performance and predictive accuracy. In [14], genetic algorithms are integrated with neural network models to address enterprise classification problems within supply chain finance systems. In this framework, evolutionary optimization techniques are used to tune model parameters, leading to improved classification accuracy. The work reported in [15] investigates the use of genetic algorithms to enhance the performance of decision tree models in machine learning applications. Evolutionary search methods allow efficient exploration of possible tree structures, which contributes to improved predictive performance. In addition to evolutionary approaches, a number of studies have investigated alternative optimization strategies based on exact or hybrid computational methods. For example, authors [16] investigated optimal design problems for multiproduct batch plants using a parallel branch-and-bound approach, demonstrating the potential of parallel optimization techniques for solving complex combinatorial problems. In a later study, authors [17] conducted a comparative analysis of GPU-parallelized metaheuristics and branch-and-bound methods for batch plant optimization, highlighting the advantages and limitations of different optimization paradigms in large-scale industrial applications. These studies illustrate the diversity of optimization methodologies and emphasize the importance of developing flexible optimization frameworks capable of addressing complex nonlinear and high-dimensional problems.

Despite the significant progress reported in the literature, many studies primarily focus on the practical effectiveness of genetic algorithms in solving applied problems, while a comprehensive understanding of their broader optimization potential remains an open research area. Consequently, further investigation is required to fully explore their capabilities and limitations in complex optimization environments.

B. PURPOSE OF THE STUDY

The purpose of this paper is to develop, test, and validate a universal genetic optimizer (UGO) that integrates genetic algorithm mechanisms with neural network–based techniques. The study aims to evaluate the effectiveness of the proposed optimizer and to demonstrate its potential advantages over conventional optimization tools in identifying reliable global optima across a diverse set of benchmark functions relevant to business decision-making problems.

II. THE THEORETICAL BACKGROUNDS

In the context of the digital economy, business environments are increasingly characterized by high levels of complexity, uncertainty, and rapid structural change. Under such conditions, traditional deterministic optimization methods often encounter significant limitations when dealing with large and irregular search spaces. As a result, metaheuristic approaches - particularly genetic algorithms (GAs) - have attracted considerable attention as effective tools for solving computationally intensive optimization problems [18].

Genetic algorithms are inspired by the principles of natural selection and biological evolution and translate these mechanisms into computational procedures for solving optimization tasks. Within this framework, the search for an optimal solution is interpreted as an evolutionary process in which a population of candidate solutions evolves over time. Each candidate solution is encoded as a chromosome, and the quality of the solution is evaluated with respect to a predefined objective function. In most implementations, chromosomes are represented as vectors composed of discrete elements referred to as genes. Each gene corresponds to a decision variable or parameter within the optimization model. The algorithm begins by generating an initial population of such chromosomes, typically using random initialization in order to ensure sufficient diversity and to avoid premature convergence toward suboptimal regions of the search space.

The quality of each candidate solution is assessed using a fitness function that quantifies how effectively the chromosome satisfies the optimization objective. This evaluation determines the probability that a given solution will survive and contribute to the formation of subsequent generations. During the selection phase, chromosomes with higher fitness values are more likely to be chosen as parents, while weaker individuals are gradually eliminated from the population. The evolutionary search process is driven by several genetic operators, including crossover, mutation, and inversion [19, 20]. Crossover, also referred to as recombination, simulates biological reproduction by exchanging segments of genetic information between two parent chromosomes. For example, if the chromosomes (1, 2, 3, 4, 5) and (0, 0, 0, 0, 0) are divided between the third and fourth genes, recombination produces two new offspring: (1, 2, 3, 0, 0) and (0, 0, 0, 4, 5). This mechanism enables the combination of advantageous genetic traits and contributes to a more effective exploration of the search space.

Mutation introduces small random changes in individual genes, which helps maintain population diversity and prevents the search process from becoming trapped in local optima [21].

In addition, the inversion operator rearranges the order of genes within a chromosome, thereby modifying structural relationships between decision variables and improving the exploration of alternative solution configurations.

Through repeated cycles of evaluation, selection, and genetic transformation, the population gradually evolves toward higher-quality solutions. Each generation represents an incremental refinement of the candidate solution set, allowing the algorithm to progressively approach an optimal or near-optimal solution. The evolutionary process typically terminates when a predefined stopping criterion is satisfied, such as reaching a maximum number of generations or observing no significant improvement in the fitness value over successive iterations. The overall workflow of the genetic algorithm is illustrated in Fig. 1. From a methodological perspective, the algorithm can be interpreted as a sequence of iterative computational experiments in which each generation represents an improved approximation of the optimal configuration of decision variables.

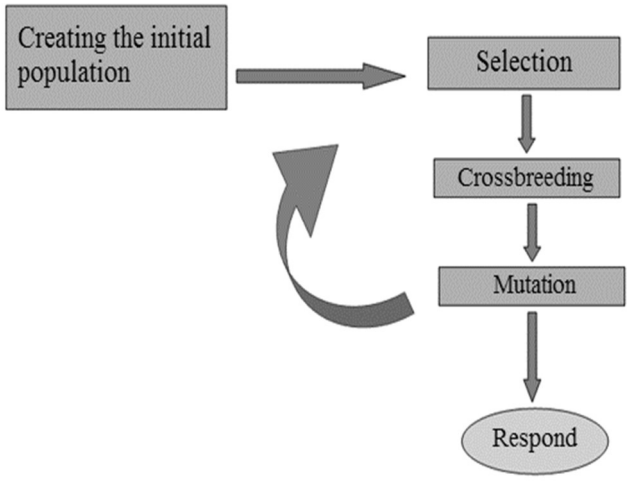


Figure 1. Block diagram of the genetic algorithm

The methodological framework for determining optimal parameter settings for genetic algorithms (GAs) can be viewed as a structured process that combines computational experimentation, statistical analysis, and mathematical modeling [22]. The primary goal of such a framework is to identify parameter configurations that ensure high algorithmic efficiency, robustness, and scalability when solving diverse classes of optimization problems. Since genetic algorithms are stochastic in nature and their performance is highly sensitive to parameter tuning, the systematic investigation of algorithmic parameters plays a critical role in achieving stable and reproducible optimization results.

The optimization problem considered in this study can be formally expressed as the search for a vector of decision variables

$$x = (x_1, x_2, ..., x_n)$$

which finds the objective function

$$f(x),\ x \in \Omega$$

where x represents a chromosome encoding a candidate solution, f(x) is the fitness (utility) function, and Ω denotes the feasible search space.

The goal of the optimization process is therefore to determine

$$x^* = \arg\min_{x \in \Omega} f(x)$$

or, in the case of maximization problems,

$$x^* = \arg\max_{x \in \Omega} f(x)$$

This formulation provides a general framework for applying evolutionary optimization techniques to a wide class of nonlinear and multidimensional problems.

In practice, the development of an effective parameterization strategy involves several interconnected stages that together form an integrated experimental workflow.

The first stage involves selecting a representative set of benchmark functions that emulate different types of optimization landscapes. These test functions are used to evaluate the behavior of the algorithm under varying levels of complexity and structural properties of the search space. Typical benchmark problems include convex, multimodal, discontinuous, and noisy objective functions, each designed to assess specific capabilities of the algorithm - such as escaping local optima, maintaining population diversity, or exploring high-dimensional search spaces. Frequently used benchmark functions include the Rosenbrock, Rastrigin, Ackley, and Griewank functions, which are widely adopted in the optimization literature for algorithmic validation [23].

Computational experiments and numerical simulations. The next stage consists of conducting a series of computational experiments in which the genetic algorithm is executed under different combinations of control parameters. These parameters typically include population size, crossover probability, mutation rate, selection pressure, and elitism mechanisms. During each experimental run, a set of performance indicators is recorded, including convergence rate, computational cost (CPU time), solution accuracy, and stability across repeated runs. To ensure statistical reliability, the experiments are usually repeated multiple times with different random seeds. The resulting data are analyzed using statistical indicators such as the mean best fitness value, standard deviation, and convergence curves. These metrics provide insight into the consistency and robustness of the algorithm's performance under varying parameter settings [24].

Statistical modeling and regression analysis. The experimental results obtained in the previous stage serve as the basis for constructing statistical models that describe the relationship between algorithm parameters and performance outcomes. Regression analysis enables the identification of significant factors, interaction effects, and potential nonlinear dependencies among variables. In many studies, techniques such as multivariate regression, response surface methodology (RSM), and analysis of variance (ANOVA) are applied to formalize these relationships and to construct predictive models of algorithm performance. Such models allow researchers to better understand how individual parameters influence convergence behavior and computational efficiency, thereby facilitating the systematic tuning of algorithm configurations.

Optimization of algorithm parameters. Once the regression model has been established, it can be used to determine parameter values that maximize the performance of the genetic algorithm. Key parameters that are typically optimized include population size, crossover probability, mutation rate, and inversion probability [25]. The optimal configuration can be obtained either analytically - by identifying extrema of the regression surface - or through additional targeted computational experiments performed near the predicted optimal region. The resulting parameter configuration reflects

a balance between exploration and exploitation within the evolutionary search process, enabling the algorithm to efficiently converge toward a global optimum while avoiding excessive computational overhead.

Performance evaluation criteria. A commonly used performance indicator in evolutionary optimization is the number of generations required for the algorithm to reach an optimal or near-optimal solution. This measure reflects both the computational efficiency of the algorithm and its dynamic convergence behavior. In general, a smaller number of generations indicates better adaptation of the algorithm to the structure of the optimization landscape and more effective parameter tuning.

To obtain a comprehensive evaluation of algorithm performance, additional criteria are often considered, including convergence stability, variance of fitness values across multiple runs, and total execution time. Hybrid optimization architectures. The proposed methodological framework also provides a basis for developing hybrid optimization systems in which genetic algorithms are integrated with other intelligent computational paradigms. Such hybrid architectures may combine evolutionary search with artificial neural networks (ANNs), fuzzy inference systems, swarm intelligence techniques, or reinforcement learning models. By integrating complementary computational mechanisms, hybrid systems can exploit the strengths of different approaches: the global search capability and adaptability of genetic algorithms, the learning capacity of neural networks, and the rule-based reasoning of fuzzy systems. As a result, hybrid optimization frameworks often demonstrate improved adaptability, knowledge transfer, and dynamic learning during the optimization process.

Applications in business analytics and decision support. In the context of business analytics and decision-support systems, such hybrid optimization approaches are particularly useful for solving complex problems characterized by multiple conflicting objectives, uncertainty, and dynamically changing constraints. Typical applications include resource allocation, supply chain optimization, investment portfolio management, production scheduling, and risk assessment. Unlike deterministic optimization techniques used in classical mathematical programming, genetic algorithms are capable of efficiently exploring large, nonlinear, and multidimensional solution spaces while maintaining robustness in uncertain and dynamic environments. This property makes evolutionary optimization methods especially attractive for decision-making problems in modern data-driven economic systems.

Thus, a hybrid optimization algorithm is proposed that is performed on selected control functions. The algorithm integrates a genetic algorithm with a neural-network-based parameter adaptation mechanism. Within the proposed framework, the neural network is used as an adaptive modeling component that captures the relationship between the parameters of the genetic algorithm and its optimization performance. The network receives as input a set of indicators describing the current state of the evolutionary process, such as convergence speed, population diversity, and fitness improvement between successive generations. Based on these inputs, the neural network estimates adjusted values of key control parameters, including population size, crossover probability, and mutation rate. In this way, the neural network provides a dynamic mechanism for parameter adaptation, allowing the genetic algorithm to better balance exploration and exploitation during the optimization process.

The general workflow of the algorithm consists of the following steps:

Step 1: Generate initial population P
Step 2: Evaluate fitness function F(x)
Step 3: Train neural network using collected performance data
Step 4: Predict optimal GA parameters
Step 5: Apply selection operator
Step 6: Apply crossover operator
Step 7: Apply mutation operator
Step 8: Generate new population
Step 9: Check termination condition
Step 10: Return best solutiontion of benchmark functions.

The algorithm iteratively updates the population until the stopping criterion is satisfied.

This hybrid architecture improves the robustness of the search procedure and reduces the risk of premature convergence when solving complex multimodal optimization problems.

III. THE RESULTS AND DISCUSSION

All experiments were performed on a standard desktop computer equipped with an Intel Core i5 (8th generation) processor, 16 GB of RAM, and running a 64-bit Windows 10 operating system. The optimization experiments were executed without the use of specialized high-performance computing infrastructure. Depending on the dimensionality of the benchmark function and algorithm parameters, the runtime of a single optimization run typically ranged from several seconds to several minutes.

Due to the population-based nature of genetic algorithms, the evaluation of candidate solutions can be performed independently, which makes the proposed optimizer well suited for parallel implementation on multi-core CPUs or GPU-based systems. This property suggests strong scalability potential for solving larger and more computationally intensive optimization problems.

A. EXPERIMENTAL FUNCTIONS

With the aim of implementing the aforementioned research in practice, the author independently identified the following experiments based on their own program, referencing relevant literature as outlined in the textual material.

Experimental functions: 1. Rosenbrock; 2. Two-dimensional exponential2; 3. De Jong2; 4. De Jong5; 5. Rasstrigin Griewank; 6. Beale; 7. Freudenstein and 8. Roth. The intermediate processes of their processing are presented in Fig. 2.

B. ROSENBROCK FUNCTION

The minimum equals 0 at the point where x1 = 1.0 and x2 = 1.0. The function has a two-valley (bimodal) nature, which complicates finding its minimum for some cyber-computational algorithms:

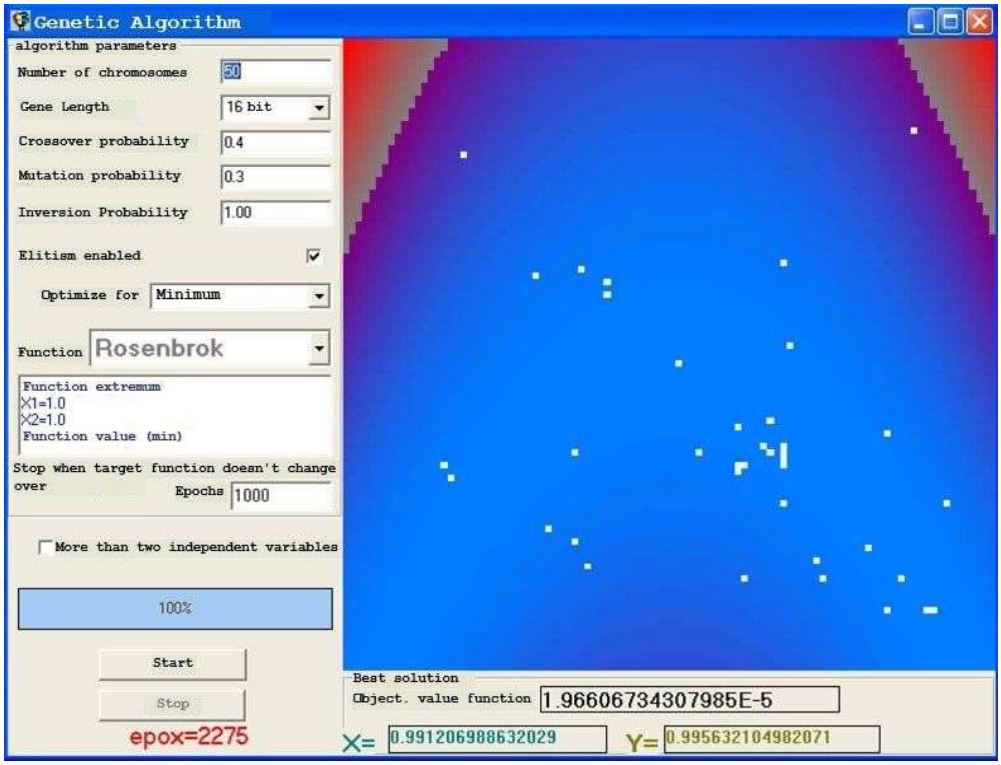
$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \rightarrow \min \qquad (1)$$

C. EXPONENTIAL FUNCTION (Expfunc 2)

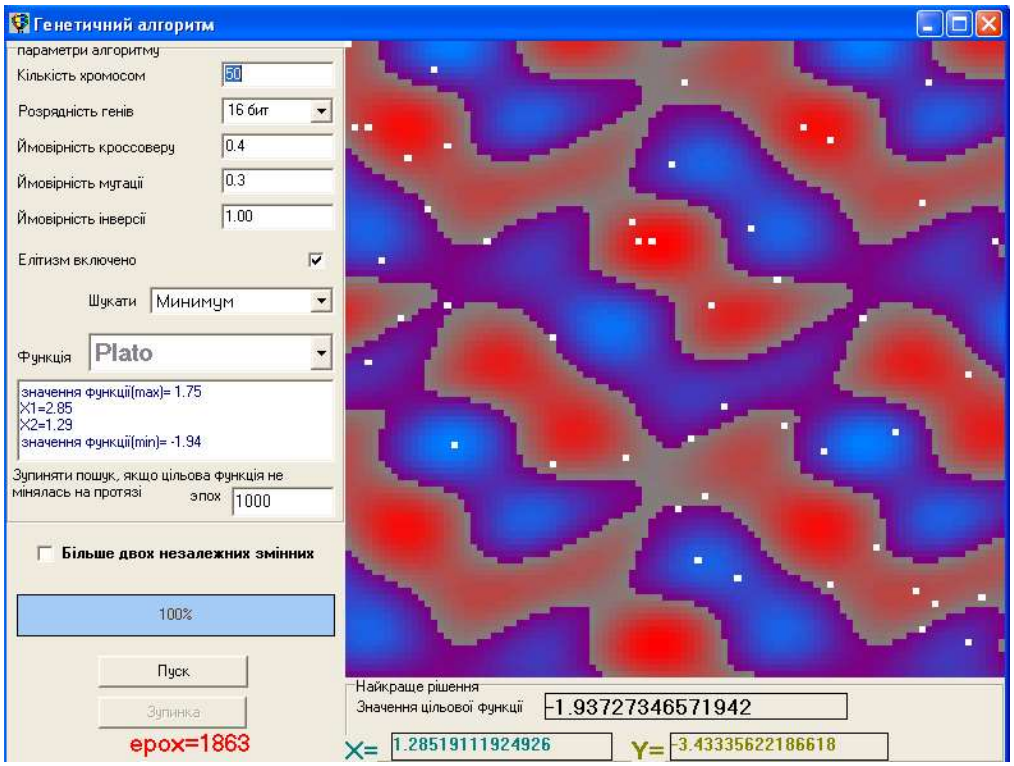
Based on the aforementioned open information sources, the author conducts experimental research on typical algorithms for finding optimal business solutions, which is outlined as follows.

$$f(x_1, x_2) = \sum_{a=0.1(0.1)}^{1} [(e^{-ax_1} - e^{-ax_2}) - (e^{-a} - e^{-10a})]^2 \rightarrow \min \quad (2)$$

A distinctive feature of this function is its almost flat nature near the minimum, which makes it difficult for computational algorithms to find the optimal value. The minimum equals 0 at the point x = (1; 10).



a



b

Figure 2. Action of the genetic algorithm for a) Rosenbrok and b) Plato: the red areas are regions with high function values (maxima); the blue areas are regions with low function values (minima); the white dots are the current chromosomes (solutions) that the algorithm has placed in the search space.

D. DE JONG 2 FUNCTION

The minimum equals 0 at the point where x = (1.0). The function has a large, rapidly decreasing plateau, x (-2,048; 2,048):

$$f(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2) \quad (3)$$

E. DE JONG 5 FUNCTION

Where i = 2:

$$f=0.002+\sum_{j=1\partial o25}{}_j(1/[\sum(x_i-a_i)^6]) \quad (4)$$

F. RASTRIGIN FUNCTION

A function with a complex landscape. It is considered difficult to optimize. x ∈ (-5.12; 5.12). The minimum equals 0 at the point where xi = 0.0. A local minimum is located at the point where one coordinate equals 1.0, and all others equal 0.0.

G. BEALE FUNCTION

The minimum equals 0 at the point (3, 0.5). The starting point is (1, 1):

$$\begin{aligned}&f(x,y)=(1.5-x(1-y))^2+(2.25-x(1-y^2))^2\\&+(2.625-x(1-y^3))^2\end{aligned} \quad (5)$$

H. FREUDENSTEIN AND ROTH FUNCTION

The minimum equals 0 at the point (5, 4), and the maximum equals 48.9842 at the point (11.41, -0.8986). The starting point is (0.5, -2).

$$\begin{aligned}&f(x,y)=(-13+x+((5-y)y-2)y)^2+(-29+x\\&+((y+1)y-14)y)^2\end{aligned} \quad (6)$$

I. L. GRIEWANK FUNCTION

x ∈ (-600; 600). The minimum equals 0 at the point where xi = 0.0:

$$f(x)=1+\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}, \quad (7)$$

J. TEST FUNCTIONS

To test the performance of the GA with the already determined parameters, we used functions that are considered more challenging for finding the global extremum. Additionally, these functions have both a global minimum and a global maximum.

K. Function "Powel Badly Scaled (Plato)"

The Powell function takes positive values, and the exact solution for this optimization function is: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$.

$$\begin{aligned}&f(x_1,x_2,x_3,x_4)=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-\\&2x_3)^4+10(x_1-x_4)^4\rightarrow\min\end{aligned} \quad (8)$$

Function:

$$\begin{aligned}&f(x,y)=0.8\exp(-r_1^2/(0.3)^2)+0.879008\exp(\\&-r_2^2/(0.03)^2)\end{aligned} \quad (9)$$

$$r_1^2=(x-0.5)^2+(y-0.5)^2$$

$$r_1^2=(x-0.6)^2+(y-0.1)^2$$

Table 1. Regression model coefficients: R – number of generations (epochs); Cr – number of chromosomes; Ge – gene bit depth; Cs – crossover probability; Mu – mutation probability; In – inversion probability

	Unstandardized Coefficients B	Std. Error	Standardized Coefficients Beta	T	Sig.
(Constant)	9452,833	968,465		9,761	0,000
CS	-14480,000	5832,323	-1,412	-2,483	0,014
IN	-10342,939	31784,091	-0,303	-0,325	0,745
CR_CS	4,108	65,072	0,052	0,063	0,950
CR_MU	-19,951	28,497	-0,252	-0,700	0,485
CR_IN	63,460	174,694	0,241	0,363	0,717
GE_CS	84,177	194,994	0,170	0,432	0,667
GE_MU	-344,497	408,038	-0,697	-0,844	0,400
GE_IN	184,959	733,187	0,112	0,252	0,801
CS_MU	3328,730	8201,748	0,218	0,406	0,685
CS_IN	3825,267	22174,574	0,075	0,173	0,863
CR_CR	1,094E-02	0,040	0,040	0,275	0,783
GE_GE	0,427	1,552	0,040	0,275	0,783
CS_CS	11425,327	4736,621	1,101	2,412	0,017
MU_MU	3544,380	3173,166	0,342	1,117	0,266

L. EXPERIMENTAL DESIGN

Before conducting the experiments, we determined an orthogonal plan for their execution using the statistical package SPSS. The experimental design includes the variation of the factors: number of chromosomes; bit depth of genes; probability of crossover; probability of mutation; probability of inversion. It should also be noted that "elitism" was always enabled because we observed from the very beginning that the algorithm with the elitism option enabled always yields better results.

All these values were then processed in the SPSS 9.0 for Windows program to find the regression equation.

$$R = 9452.883 - 14480Cs - 10342.939In + 4.108CrCs - 19.951CrMu + 63.460CrIn + 84.177GeCs - 344.497GeMu + 184.959GeIn + 3328.730CsMu + 3825.267CsIn + 0.01094Cr^2 + 0.427Ge^2 + 11425.327Cs^2 + 3544.380Mu^2 \quad (10)$$

where R - number of generations (epochs), Cr - number of chromosomes, Ge - bit depth of genes, Cs - probability of crossover, Mu - probability of mutation, In - probability of inversion

The results of the statistical analysis are presented in Table 1.

M. DETERMINATION OF THE OPTIMAL PARAMETERS OF THE GENETIC MODEL

In the Excel (Solver) program, we obtained the values of the GA parameters that minimize the function R(Cr, Ge, Cs, Mu, In) → min. To maintain objectivity, we did not use our optimizer for this purpose.

Optimal values of GA parameters: CR = 50; GE = 16; CS = 0,4; MU = 0,3; IN = 1; R = 2331,54312.

By substituting these values into the GeneBase program to check how it performs with test functions that were not used in the experiments, these test runs of the model, using the obtained optimal parameter values, showed very encouraging results on all test and experimental functions. The results of the GA test runs are presented in Table 2.

Table 2. Results of GA test runs with optimal parameters

№	Function name	Average number of generations to reach the global extremum
1.	Rosenbrock	1216
2.	Two-dimensional exponential2	854
3.	De Jong2	2543
4.	De Jong5	1645
5.	Rasstrigin	758
6.	Griewank	525
7.	Beale	2154
8.	Freudenstein and Roth	811
9.	Powel Badly Scaled (Plato)max	1823
10.	Powel Badly Scaled (Plato)min	1248
11.	$f(x,y) = 0.8\exp(-r_1^2/(0.3)^2) + +0.879008\exp(-r_2^2/(0.03)^2)\max$	1672

We would like to thank all the numerous authors whose works and developments we utilized in the search for optimal parameters of the genetic algorithm, and above all, the authors of the well-known GeneBase algorithm, who provided the GA code implemented in the Delphi programming language for further modification and experimentation. This enabled us to conduct a large number of experiments and find a good set of GA parameters that performed excellently across a wide array of experimental and test functions.

IV. CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

As a result of the conducted research, the key parameter values of the Genetic Algorithm (GA) were identified, leading to a configuration that demonstrated strong optimization performance across both benchmark and experimental test functions. When these optimized parameters were applied, the developed GA exhibited stable convergence, high search efficiency, and a robust ability to avoid local extrema. The algorithm's adaptability was particularly evident in its performance on nonlinear, multimodal functions, where conventional optimization techniques frequently fail to locate the true global optimum.

In the course of experimental testing, the neural network–based genetic optimizer showed exceptional effectiveness by integrating the exploratory power of evolutionary computation with the adaptive learning capabilities of neural architectures. The hybrid system dynamically adjusted its evolutionary parameters in response to the search landscape, thereby enhancing both convergence rate and solution accuracy. This self-organizing behavior enabled the optimizer to locate global extrema on a representative set of two-dimensional benchmark functions, including those with multiple local minima, discontinuities, or irregular search surfaces. Such results confirm the algorithm's high potential for application in real-world optimization problems, including those in antenna design, electromagnetic field tuning, and signal processing, where solution spaces are typically non-convex and multidimensional.

When compared with Excel Solver, which relies primarily on deterministic gradient-based methods, the developed optimizer achieved superior qualitative metrics across all studied functions. Specifically, it showed faster convergence, reduced sensitivity to initial conditions, and greater consistency in reaching the global optimum. The stochastic nature of the GA, reinforced by neural network adaptation, allowed it to maintain a balance between exploration (diversity of solutions) and exploitation (refinement of the best candidates).

The most effective GA configuration, determined under the enforced use of elitism to preserve the fittest individuals between generations, includes the following parameters:

- Number of chromosomes: 50
- Number of genes per chromosome: 16
- Crossover probability: 0.4
- Mutation probability: 0.3
- Inversion probability: 1.0

This parameter combination ensures sufficient population diversity and controlled convergence, allowing the algorithm to thoroughly explore the solution space while maintaining computational efficiency. Elitism was found to be particularly

crucial for stabilizing evolutionary progress and preventing premature convergence toward suboptimal solutions.

Given the promising results obtained on two-dimensional test functions, further research should focus on extending the method to multidimensional optimization tasks, where problem complexity increases exponentially. Such an extension will require refining the genetic representation, improving neural network feedback mechanisms, and integrating adaptive control of mutation and crossover probabilities. Additionally, applying this hybrid optimizer to practical engineering problems - such as antenna array pattern synthesis, impedance matching, or multi-objective electromagnetic optimization - could demonstrate its scalability and robustness in real-world conditions.

In summary, the presented neural network–enhanced genetic algorithm combines the global search capabilities of evolutionary computation with the intelligent adaptability of neural models. The obtained results confirm its potential as a powerful optimization tool, particularly suited for complex, nonlinear, and high-dimensional problems encountered in modern engineering and computational science.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Boston, MA, USA: Pearson, 2021.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[3] S. Robotko, A. Topalov, S. Nekrasov, V. Zaitsev, and D. Zaitsev, "Machine learning and modeling of the impact of trademark filings on GDP growth based on Python," *Proceedings of the 5th Int. Workshop Modern Mach. Learn. Technol. Data Sci. (MoMLeT+DS 2023)*, Lviv, Ukraine, Jun. 3, 2023, pp. 65–76. [Online]. Available at: https://ceur-ws.org/Vol-3426/paper6.pdf.

[4] Y. Zheng, J. Wang, S. Ryzhkov, O. Nechai, A. Topalov, O. Zivenko, S. Babchuk, and T. Harasymiv, "Computer system for forecasting water quality parameters based on machine learning," *Comptes Rendus de L'Academie Bulgare des Sciences*, vol. 77, no. 11, pp. 1629–1638, 2024. https://doi.org/10.7546/CRABS.2024.11.06.

[5] S. Robotko, A. Topalov, V. Lukashova, V. Zaytsev, and D. Zaytsev, "A machine learning study on world economic impact of intellectual property rights on GDP growth with a focus on Poland," *Proceedings of the 13th Int. Conf. Adv. Comput. Inf. Technol. (ACIT)*, Wrocław, Poland, 2023, pp. 261–264, https://doi.org/10.1109/ACIT58437.2023.10275475.

[6] Y. Davidor, "Epistasis variance: A viewpoint on GA-hardness," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1991, pp. 23–35. https://doi.org/10.1016/B978-0-08-050684-5.50005-7.

[7] K. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, MI, USA, 1975.

[8] D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. London, U.K.: Pitman, 1987, pp. 74–88.

[10] TechStartups, "NeuroMesh: Spearheading the new era of AI with a distributed training protocol," TechStartups, Apr. 9, 2024. [Online]. Available at: https://techstartups.com/2024/04/09/neuromesh-spearheading-the-new-era-of-ai-with-a-distributed-training-protocol/.

[11] Y. Wang, Q. Li, and S. Chen, "Using neural-genetic hybrid systems for complex decision support," *Neural Comput. Appl.*, vol. 35, pp. 11403–11416, 2023, https://doi.org/10.1007/s00521-023-08305-6.

[12] R. Sivakumar and R. Dhanasekaran, "Applying soft computing methods in business analytics using hybrid genetic algorithms," *ICTACT J. Soft Comput.*, vol. 15, no. 2, pp. 3539–3544, 2023. https://doi.org/10.21917/ijsc.2024.0493.

[13] Contino, "Optimising business processes: The Darwinian way to better decision making," Contino Insights. [Online]. Available at: https://www.contino.io/insights/optimising-business-processes-using-genetic-algorithms.

[14] D. Karaboga and B. Akay, "Selecting an optimal architecture of neural network using genetic algorithm," *Procedia Comput. Sci.*, vol. 187, pp. 252–258, 2021, https://doi.org/10.1016/j.procs.2021.04.055.

[15] X. Liu, Y. Zhang, and J. Wang, "Application of genetic algorithm and BP neural network in supply chain financing," *Eur. J. Oper. Res.*, vol. 284, no. 3, pp. 1074–1083, 2020, [Online]. Available at: https://www.sciencedirect.com/science/article/pii/S0377042720304611.

[16] A. Borisenko, P Kegel, S Gorlatch, "Optimal design of multi-product batch plants using a parallel branch-and-bound method," *Proceedings of the* International Conference on Parallel Computing Technologies, 2011, pp. 417-430. https://doi.org/10.1007/978-3-642-23178-0_36.

[17] A. Borisenko, S. Gorlatch, "Comparing GPU-parallelized metaheuristics to branch-and-bound for batch plants optimization," *The Journal of Supercomputing*, vol. 75, issue 12, pp. 7921-7933. 2019. https://doi.org/10.1007/s11227-018-2472-9.

[18] M. A. Al-Betar, M. A. Awadallah, I. A. Doush, K. Alyass, O. Alqaisia, "A comprehensive review of genetic algorithms: operators, analyses, and applications," *Applied Soft Computing*, vol. 97, 106721, 2020.

[19] Y. Al-Sahou, M. A. Al-Betar, "A review of genetic algorithm components and applications," *Archives of Computational Methods in Engineering*, vol. 29, issue 7, pp. 4165–4198, 2022.

[20] R. Poli, W. B. Langdon, N. F. McPhee, "A field guide to genetic programming," *Lulu Enterprises*. 2008.

[21 K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons. 2001.

[22] A. H. Gandomi, A. H. Alavi, "Cuckoo search algorithm: a review of the literature and its applications," *Journal of Applied Soft Computing*, vol. 12, issue 12, pp. 4338–4348, 2012.

[23] A. P. Piotrowski, J. J. Napiorkowski, "Performance of evolutionary algorithms on benchmark functions," *Entropy*, vol. 22, issue 8, 856, 2020.

[24] M. Črepinšek, M. Mernik, S. Liu, "Exploitation and exploration in evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, issue 3, pp. 1–33, 2013. https://doi.org/10.1145/2480741.2480752.

[25] J. Dréo, P. Siarry, A. Petrowski, E. Taillard. *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer. 2006.

***ANDRII SHKITOV** Postgraduate student and researcher at the Department of Computer Engineering, Open International University of Human Development "Ukraine", Kyiv, Ukraine. Scientific interests: algorithm optimization, genetic algorithms, Kalman filters, artificial intelligence, decision support systems, cybersecurity, and digital transformation.*

***PETRO TALANCHUK**, Dr. of Sciences, Professor, Academician of the National Academy of Pedagogical Sciences of Ukraine, President of the Open International University of Human Development "Ukraine".Scientific interests: education management, innovative pedagogical technologies, human development, informatization of education, social transformation processes.*

***OLEKSII ZIVENKO**, P.h.D, specialty 05.13.05 – Computer Systems & Components. Assoc. Professor at the Marine Instrumentation Department, Admiral Makarov National University of Shipbuilding. Scientific interests: automation and optimization of technological processes, information theory and measuring complexes, sensors.*

***ANATOLII TYMOSHENKO**, P.h.D, Professor at the Department of Computer Engineering, Open International University of Human Development "Ukraine". Scientific interests: computer systems architecture, network technologies, cybersecurity, intelligent systems, optimization of information processes.*

***VOLODYMYR PAVLENKO,** P.h.D, Assoc. Professor at the Department of Computer Engineering, Open International University of Human Development "Ukraine". Scientific interests: computer systems optimization, artificial intelligence, decision support systems, information security, digital transformation of business processes.*

***VITALIA KROPYVNYTSKA,** P.h.D, specialty 05.13.07 – Automation of technological processes. Assoc. Professor at the Department of Computer Systems and Networks, Ivano-Frankivsk National Technical University of Oil and Gas. Scientific interests: automation and complex mechanization of chemical and technological processes, computer engineering.*