

RESEARCH ARTICLE

The Evolutionary Convergent Algorithm: A Guiding Path of Neural Network Advancement

EGHBAL HOSSEINI¹,  **ABBAS M. AL-GHAILI**^{1,2},  **DLER HUSSEIN KADIR**^{3,4}, **FATEMEH DANESHFAR**⁵, **SARASWATHY SHAMINI GUNASEKARAN**^{1,2}, **AND MUHAMMET DEVECİ**^{6,7,8}

¹Institute of Informatics and Computing in Energy (IICE), UNITEN, Kajang, Selangor 43000, Malaysia

²Department of Computing, College of Computing and Informatics (CCI), Universiti Tenaga Nasional (UNITEN), Kajang, Selangor 43000, Malaysia

³Department of Statistics, College of Administration and Economics, Salahaddin University-Erbil, Erbil 44002, Iraq

⁴Business Administration Department, Cihan University-Erbil, Erbil 44001, Iraq

⁵Department of Computer Engineering, University of Kurdistan, Sanandaj 66177-15175, Iran

⁶Department of Industrial Engineering, Turkish Naval Academy, National Defence University, 34942 Tuzla, Istanbul, Turkey

⁷Royal School of Mines, Imperial College London, SW7 2AZ London, U.K.

⁸Department of Information Technologies, Western Caspian University, Baku 1001, Azerbaijan

Corresponding authors: Eghbal Hosseini (kseghbalhosseini@gmail.com) and Abbas M. Al-Ghaili (abbas@uniten.edu.my)


This work was supported in part by the Dato' Low Tuck Kwong International under Grant 20238008DLTK, in part by Tenaga Nasional Berhad (TNB) and UNITEN through the BOLD Refresh Publication Fund under Grant J510050002-IC-6 and Grant BOLDREFRESH2025-Centre of Excellence, and in part by Tan Sri Leo Moggie Chair of Energy Informatics Publication Fund under Grant 20211ECHAIR.

ABSTRACT In the past few decades, there have been multiple algorithms proposed for the purpose of solving optimization problems including Machine Learning (ML) applications. Among these algorithms, metaheuristics are an appropriate tool to solve these real problems. Also, ML is one of the advanced tools in Artificial Intelligence (AI) including different learning strategies to teach new tasks according to data. Therefore, proposing an efficient meta-heuristic to improve the inputs of the trainer in ML would be significant. In this study, a new idea centered on seed growth, Seed Growth Algorithm (SGA), as a conditional convergent evolutionary algorithm is proposed for optimizing several discrete and continuous optimization problems. SGA is used in the process of solving optimization test problems by neural networks. The problems are solved by the same neural network with and without SGA, computational results prove the efficiency of SGA in neural networks. Finally, SGA is proposed to solve very extensive test problems including IoT optimization problems. Comparative results of applying the SGA on all of these problems with different sizes are included, and the proposed algorithm suggests effective solutions within a reasonable timeframe.

INDEX TERMS Meta-heuristic approaches, seed growth algorithm, machine learning, neural networks.

I. INTRODUCTION

Recently, a lot of Meta-heuristic algorithms have been proposed by optimization scientists. Almost all of these algorithms are based on the behavior of animals and life natural systems [1], [2], [3], [4]. Only a handful of algorithms have been developed using natural phenomena or scientific theories [5], [6], [7], [8]. Among them are some popular ones such as Particle Swarm Optimization (PSO) [1], Artificial Bee Colony Algorithm [2], and Social Spider

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Li .

Optimization [3], Laying Chicken Algorithm (LCA) [4], Big Bang Algorithm (BBA) [5], Volcano Eruption Algorithm (VEA) [6], COVID-19 Optimizer Algorithm (CVA) [7], and Multiverse Algorithm (MVA) [8].

Machine learning is one of the progressive methods in artificial intelligence in which different learning strategies are used to teach new tasks according to data [9]. One of the important applications of machine learning is to use historical data as input and to accurately predict results without explicit planning. In addition, according to the nature of different machine learning algorithms, this method can be a computational approach to learning [10]. Considering that

different machine learning algorithms have different parameters, the use of optimization methods in these algorithms is very common. The purpose of using different optimization methods in machine learning is to minimize the cost function determined by the model parameters. In this method, commonly known as gradient descent, the objective is to minimize a convex function using frequent iterations and updates of network parameters [11]. After optimizing the network parameters, these models are ready to be used in different environments. In the gradient-based optimization methods, which are utilized for the purpose of training machine learning models, the parameters of machine learning models are optimized by minimizing the error between the actual and the expected results [12]. Actually, neural networks, which are the basis of many machine learning methods, are trained using the backpropagation algorithm [13]. Using this method, the weights of the neural network are adjusted based on the amount of error obtained in the previous iteration. Backpropagation is the process of calculating the derivatives of the weights, and gradient descent is the process of descending through the gradient, i.e. adjusting the model parameters to descend through the loss function. Since the performance of the backpropagation algorithm relies heavily on the training data, the biggest problem is the relative sensitivity of this algorithm to noisy and irregular data. In addition, it needs a lot of time to train the machine learning model. Another major disadvantage of the backpropagation method is getting stuck in local minima which is computationally expensive. On the other hand, gradient-based methods have many limitations too. Commonly their quality is very dependent on the step size. If the step size is too large, we may never converge to a local minimum because we pass through it every time. And if the step size is small, it may converge after a long time [14]. Another important drawback of gradient-based methods is the extraction of multiple locally optimal solutions in problems that even have a global optimization solution. Also, the high computational complexity due to frequent updates and the use of all resources to process a training sample at any time are the other risks of these optimization methods too [15].

Over the past few years, there has been a growing trend in research towards incorporating meta-heuristic techniques for optimizing machine learning algorithm parameters. The purpose of this integration was to solve the problems of the existing optimization methods based on the gradient in optimizing the parameters of neural networks and to avoid the complex process of backpropagation. In addition, there has been much research in the field of using machine learning in meta-heuristic methods to solve optimization problems. By employing these techniques, the performance of meta-heuristic methods can be enhanced in terms of solution quality, convergence rate, and robustness, leading to more efficient and effective searches. Many of these meta-heuristic methods based on machine learning have produced high-quality and robust results so far and represent advanced optimization algorithms.

Very recently there are some attempts to optimize the outcome of machine learning methods by optimization algorithms which the following works are significant among others. A new non-convex and parameter-free surrogate has been proposed which converges and optimizes by closed-form solutions [16]. A method optimizes Two-way Partial Area Under the ROC Curve (TPAUC) taking into account the challenges associated with conducting gradient-based optimization during end-to-end stochastic training [17]. Also, a novel manifold neural network based on non-gradient optimization, with considering that the activation function is generally invertible has been proposed [18]. Finally, a Riemannian meta-optimization method and a framework have been presented to automatically learn a Riemannian optimizer and to optimize the graph structure respectively [19], [20].

Introduces the Honey Badger Algorithm (HBA), a novel metaheuristic optimization method inspired by the intelligent foraging behavior of honey badgers, was introduced offering an efficient approach to solving optimization problems [21]. Orchard Algorithm (OA), drawing inspiration from fruit gardening practices was introduced in [22]. OA employs actions like irrigation, fertilization, trimming, and grafting to cultivate fruitful optimization solutions. Gannet Optimization Algorithm (GOA) has been presented as nature-inspired metaheuristic [23]. GOA leverages gannets' foraging behaviors, including U-shaped and V-shaped diving patterns, for effective exploration and exploitation within the search space. [24] applies the Shrimp and Goby Association Search Algorithm (SGASA) to tackle large-scale global optimization problems, evaluating its performance across a range of benchmarks and real-world engineering applications. The Sine Cosine Algorithm (SCA) employs random candidate solutions, guided by sine and cosine functions, emphasizing exploration and exploitation throughout the optimization process. [25]. Harris Hawks Optimizer (HHO), drawing inspiration from the surprise pounce cooperative behavior of Harris' hawks in nature. [26]. [27] introduces the groundbreaking Monarch Butterfly Optimization (MBO), a novel nature-inspired metaheuristic algorithm.

A bidirectional LSTM networks employed to predict annual peak loads [28]. A novel deep learning model for short-term load forecasting in P2P energy trading has been proposed in [29]. An incentive-based DR program using modified deep learning and reinforcement learning was proposed [30]. [31] introduces the Multi-Objectives Renewable Energy-Generation (MORE-G) model for wind-based electricity generation. Very recently a new Deep Learning intrusion detection system has been proposed for IoT devices, featuring a four-layer Fully Connected network to identify malicious traffic [32]. This paper introduces a novel deep learning-based forecasting model for microgrid (MG) operation was proposed recently by [33] with addressing uncertainties in RESs, load, and day-ahead prices. Reference [34] introduces SConvLSTM, a gait recognition model using wearable sensors. It autonomously extracts features

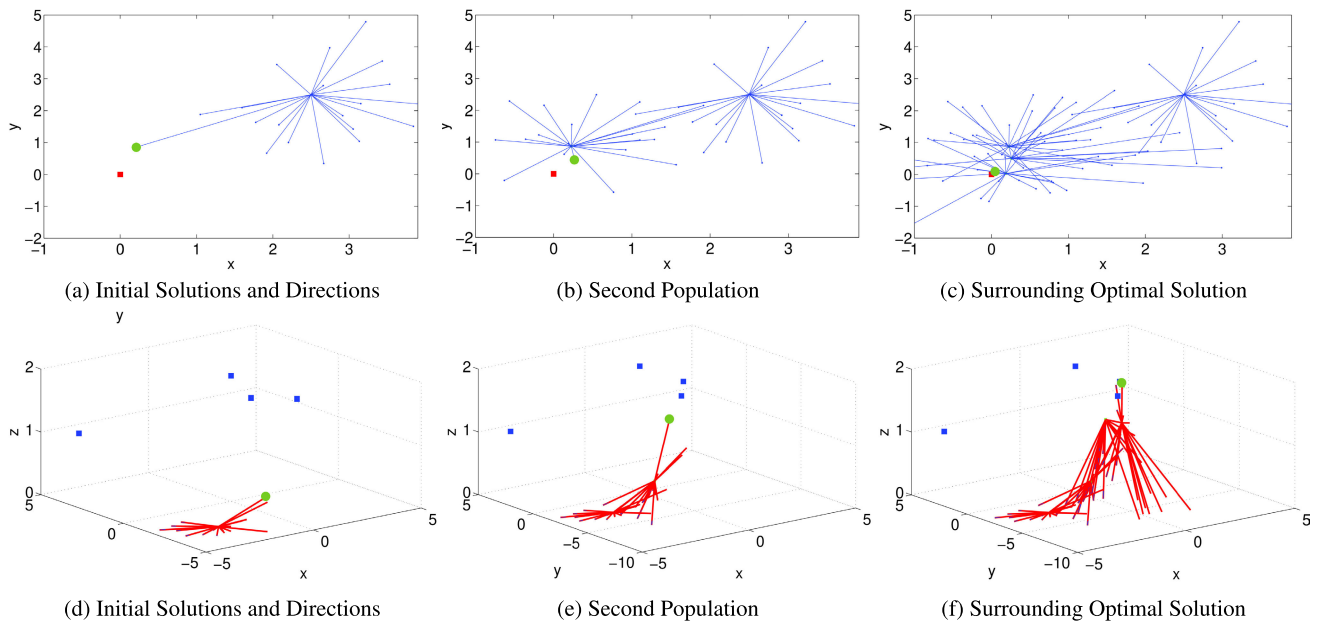


FIGURE 1. Behavior of SGA in 2D and 3D for a given optimization problem with a global optimal.

from raw sensor data, eliminating the need for manual feature design. An integrative machine learning model was presented to predict residential building energy parameters: annual thermal energy demand (DThE) and annual weighted average discomfort degree-hours (HDD) [35].

In this paper, we present a novel optimization technique for mathematical problems that exist in various continuous states. Our approach is based on the Seed Growth Algorithm (SGA), which we developed by simulating a natural event model. Seed priming of crop seeds has been suggested by numerous recent studies as a potentially beneficial method for improving germination, seedling growth, establishment, and yield. Priming for enhanced resistance to abiotic stress is operating via various pathways involved in different metabolic processes. The seedlings emerging from primed seeds showed early and uniform germination. Moreover, the overall growth of plants is enhanced due to the seed-priming treatments. The Germination stage is essential for the life cycle of all plants. Typically, the germination requirements of a species are considered adaptations to the specific environments where they grow. Due to adaptive radiation into variable and discontinuous habitats, each plant species has unique seed Germination requirements. Germination proportion and rate increased with seed storage length and were increased by short and decreased by long dry periods occurring after imbibition. Vitamins C and E and β -carotene were barely detectable in the dry grains. However, upon germination, the concentrations of these antioxidant vitamins steadily increased with increasing germination time [36], [37].

We introduce the SGA concept and explain how we adapted it from the natural simulation system to the optimization algorithm. Our experiments demonstrate that

the proposed SGA successfully achieves the desired results on a range of test problems. Also, we will examine the various opportunities for using SGA in DL. Our goal is to motivate researchers to find the optimal values of DL parameters by meta-heuristic methods.

Neural networks have undeniably become indispensable tools for addressing real-world challenges, thanks to their remarkable ability to navigate intricate, non-linear data. Nevertheless, as the complexity and dimensionality of problems grow, neural networks encounter difficulties in consistently delivering optimal solutions. Simultaneously, the promise of meta-heuristic algorithms in optimizing intricate problems is evident. However, these algorithms often bring substantial computational demands, especially in the case of high-dimensional data. The motivation behind this research stems from the fusion of these two formidable approaches, seamlessly integrating meta-heuristic algorithms within neural network frameworks. This integration endeavors to empower neural networks to more efficiently tackle real-world, non-linear challenges, potentially revolutionizing diverse domains.

The significance of this paper lies in its proposition of a convergent meta-heuristic approach, known as the Seed Growth Algorithm (SGA), poised to enhance optimization processes. SGA's importance extends to its potential to revolutionize the landscape of neural network training and its application in a broad spectrum of problem domains. The algorithm's efficiency is underscored by empirical comparisons with competitive meta-heuristics for benchmark continuous optimization. SGA's design leverages effective strategies for exploration and exploitation, enabling it to navigate the search space adeptly. In fact, this research aims

to bridge the gap by introducing an innovative and efficient meta-heuristic approach, poised to elevate the capabilities of neural networks and optimize complex real-world problems across various fields.

The main contributions of this paper are as follows:

- 1) The paper unveils SGA, a novel seed-inspired optimization method with wide applicability, redefining problem-solving paradigms.
- 2) It introduces a game-changing metaheuristic for neural network training, significantly improving performance across diverse domains.
- 3) SGA proves to be an efficient meta-heuristic, excelling in benchmark optimization problems with its unique exploration-exploitation approach.
- 4) SGA's convergence is noteworthy since the convergence of meta-heuristics is typically not guaranteed.
- 5) Simple and deep neural networks are evaluated both with and without the SGA technique. The computational results robustly demonstrate the remarkable efficiency of SGA in optimizing neural networks. Furthermore, SGA's versatility shines through in its proficient adaptation to a wide range of optimization problem domains, encompassing vehicular networks such as Software Defined Internet of Vehicles (SDIoV), Vehicular Ad hoc Networks (VANETs), and IoT traffic management within Smart Sustainable Connected Vehicles (Smart-SCV).

II. THE SEED GROWTH ALGORITHM (SGA) CONCEPT

This section serves as an introduction to the fundamental concepts of the SGA and its connection to the natural phenomena that inspired its development. The process of simulating natural system of seed within the proposed optimization algorithm includes three key stages: initialization, getting to the ground, and the emergence of roots and stems.

In the first stage, much like a solitary seed in nature, the algorithm commences with an initial solution. Just as a seed instinctively seeks a path upwards, the initial solution strives to progress toward a superior solution.

In the second stage, analogous to the way a seed matures and continues to grow in nature, the SGA builds upon the improved solution from the previous stage, progressing as far as the feasible region allows.

The third stage mimics the behavior of a seed generating roots and stems. Here, the SGA generates a set of solutions and a population, with a focus on the vicinity of the best solution obtained in the previous stage. Numerous solutions are created, and from this pool, the algorithm identifies the optimal solution once more. This three-stage process emulates the growth and evolution of natural phenomena to achieve optimized solutions.

A. THE INITIAL SOLUTION AND THE BETTER SOLUTION

The proposed method is based on this main concept of population growth, same as seeds; each solution produces two other solutions (root and stem). In this stage, each seed tries

to find a direction with solutions better than that solution. Each solution moves to a better solution to achieve the soil; this is because agents are intelligent here, unlike the natural behavior of seeds. The algorithm searches from the initial solution, x_0 , to a better solution such as x_j according to Equation 1 where the α is an arbitrary positive number, and d_j is an arbitrary direction:

$$x_i = x_0 + \alpha d_j \quad (1)$$

Algorithm 1 Pseudo-Code of Initial Solution and the Better Solution

- 1: Generate initial solution x_0 randomly
 - 2: Let a constant number α
 - 3: **while** 1 > 0 **do**
 - 4: Create a random direction d
 - 5: $x_j = x_0 + \alpha d$
 - 6: **if** $f(x_j)$ is better than $f(x_0)$ **then**
 - 7: Break for loop
 - 8: **end if**
 - 9: $j = j + 1$
 - 10: **end while**
-

B. GETTING TO THE GROUND

From the better solution in the previous stage, the movement of the initial solution will be continued until the feasible region allows. The solution moves according to Equation 2 where ϵ is a minimal positive number and d_i is the last direction in the previous stage:

$$x_{i+1} = x_i + \epsilon d_i \quad (2)$$

Algorithm 1 presents the pseudo-code for this stage, while Algorithm 2 outlines the movement to the soil in greater detail.

Algorithm 2 Pseudo-Code of Movement to the Ground

- 1: Let positive given number ϵ
 - 2: $d = d_i$
 - 3: $n = \epsilon$
 - 4: **while** x_{i+1} is in feasible space **do**
 - 5: $x_{i+1} = x_i + kd$
 - 6: $n = n + \epsilon$
 - 7: **end while**
-

C. CREATION ROOTS AND STEMS

In this stage of the SGA algorithm, a set of solutions is created as roots and stems. Near to the best solution from the previous stage, a number of roots and stems will be created and then among these solutions the best solution is found again. Finally, Equations 3 or 4 will generate a population of solutions that are highly proximate to the best solution at this stage:

$$\|X - Y\| \leq k \quad (3)$$

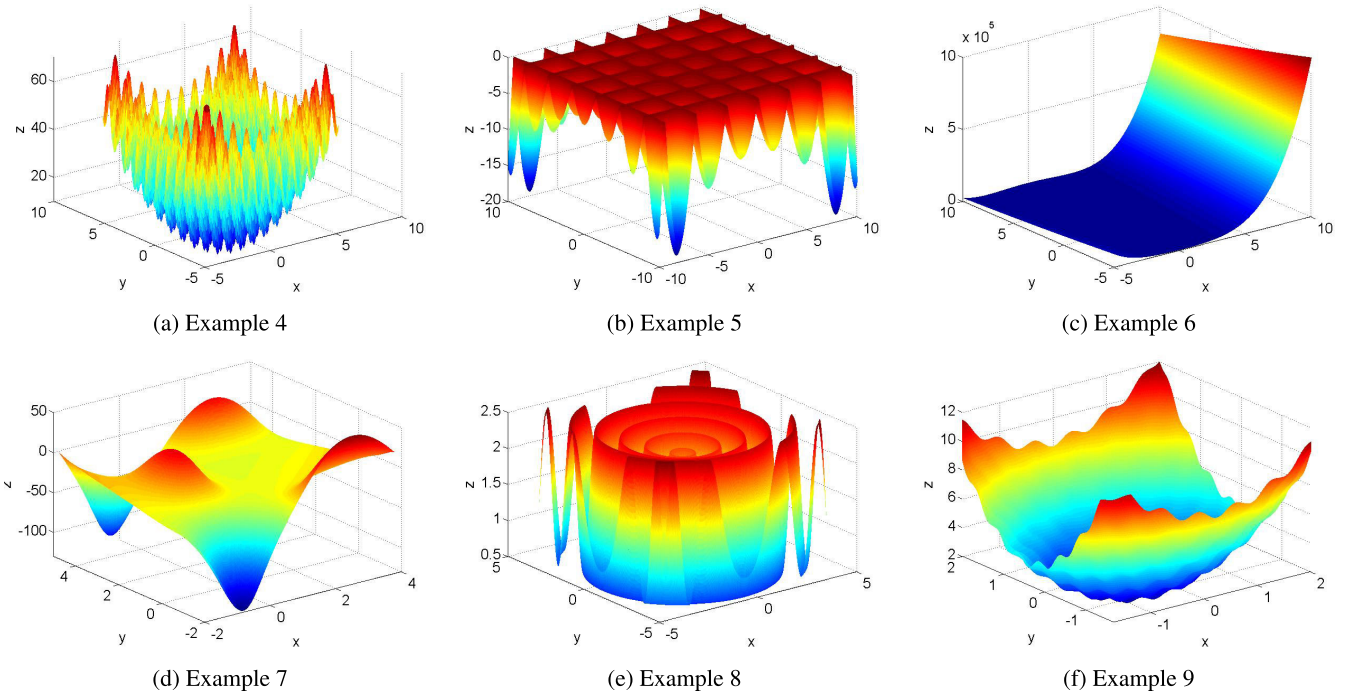


FIGURE 2. Benchmarks optimization problems.

or

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} \leq k \quad (4)$$

The n-dimensional vector Y and a small positive constant k are defined, with X representing the best solution. A population of seeds has been randomly created within the closest possible neighborhood of the best solution. The best solution of the population will be found according to the objective function; then, the SGA algorithm goes back to the first stage by the best solution.

D. STEPS OF THE SGA

In Figure 1, we illustrate the steps of SGA for optimizing two different problems, in two and three dimensions. Where the global optimal solution will be surrounded by the generated random solutions in four iterations. We represent the optimal solutions with prominent red points in the two-dimensional case (Figure 1 (a-c)) and big blue points in the three-dimensional case (Figure 1 (d-f)). The algorithm initiates with an initial solution and then proceeds to search for an improved solution. The best solution within the current population are denoted by green points. Then the algorithm commences from the best solution within the population and continues to enhance the solution further.

Here are the proposed main steps of the Seed Growth Algorithm (SGA) in the two-dimensional space R^2 :

- 1) An initial feasible solution (x_0, y_0) is randomly generated within the feasible region. The following values

are also given: N for the number of solutions, ϵ and ϵ_1 as two arbitrary small positive numbers, and M_1 as the number of iterations.

- 2) Initial solution finds a better solution using $x_i = x_0 + \alpha d_j$.
- 3) Solution i in step 2 is changed in direction of the last vector while the solution is feasible according to $x_{i+1} = x_i + \epsilon d_i$. The better solution between x_{i+1}, x_i will be selected.
- 4) By the best solution from step 3, a set of roots and stems are created, and the best will be updated again.
- 5) The initial population is generated near the best solution, and the best solution for the population is found.
- 6) Let $(x_0, y_0) = (x_{best}, y_{best})$ go back to step 2.
- 7) If $f(x_{(i+1)best}) - f(x_{ibest}) < \epsilon_1$, or the number of iteration is more than M_1 the algorithm will terminate at this point, $x_{ibest}, x_{(i+1)best}$ are the best solutions in two consecutive generations.

In our experimental setup, the algorithm's parameters are defined as follows:

- 1) **M1 (Maximum Number of Iterations):** M1 controls the maximum number of allowed iterations, and we empirically set it to 20.
- 2) **N (Number of Solutions):** The number of solutions, N , varies based on the problem. For small examples, it's set to 1; for larger cases, it's set to 50.
- 3) ϵ_1 (**Termination Threshold**): We determined ϵ_1 as 0.0001, and the algorithm stops when the improvement falls below this threshold.

- 4) x_0 (**Initial Solution**): x_0 is randomly generated and serves as the algorithm's starting point.
- 5) α (**Arbitrary Positive Number**): α is used to generate the initial population from the initial solution. We determine α through experimentation, typically within the range (0, 2]. A smaller α enhances accuracy but reduces speed, while a larger α improves exploration and speed at the cost of accuracy.
- 6) ϵ (**Control Parameter for Feasible Region**): To keep the Stochastic Genetic Algorithm within the feasible region, we set ϵ . Its value is empirically determined, and for our experiments, we used 0.1. Choosing a large ϵ can lead to infeasible solutions, so its value depends on the problem's constraints.
- 7) k (**Control Parameter for Seed Population**): A positive constant, k ensures that the seed population is randomly created close to the best solution. We empirically set k to 1. A smaller k focuses on the best solution but may lead to local optima, while a larger k enhances exploration but might miss better solutions nearby.
- 8) d_j (**Arbitrary Direction**): d_j represents an arbitrary direction used within the algorithm.

These parameter values are selected based on a combination of theoretical considerations and empirical testing, with the aim of optimizing the algorithm's performance for various problem scenarios.

E. PSEUDOCODE AND FLOWCHART OF THE SGA

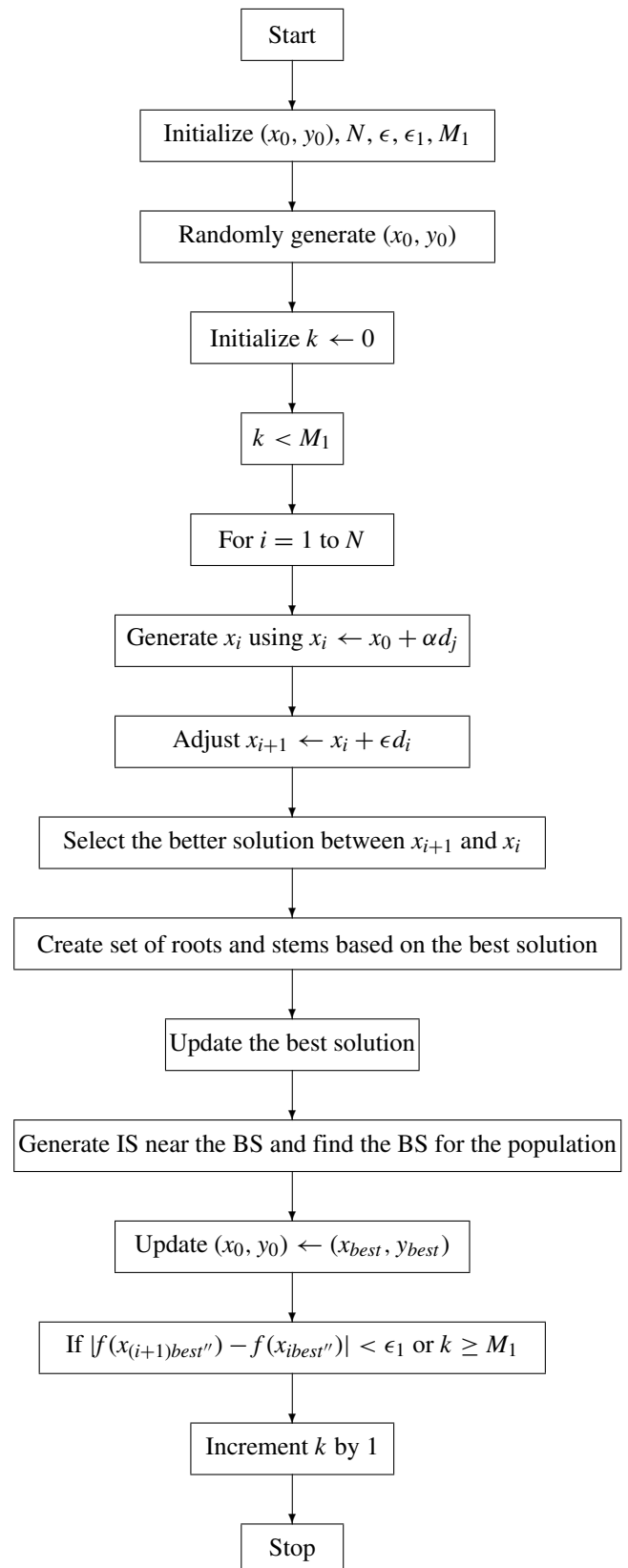
1) PSEUDOCODE

Algorithm 3 Seed Growth Algorithm (SGA) in R^2

- 1: Initial feasible solution (x_0, y_0) , number of solutions N , small positive numbers ϵ , ϵ_1 , and maximum iterations M_1
- 2: Randomly generate an initial feasible solution (x_0, y_0) within the feasible region
- 3: Initialize iteration counter $k \leftarrow 0$
- 4: **while** $k < M_1$ **do**
- 5: **for** $i = 1$ to N
 - Generate a new solution x_i using $x_i \leftarrow x_0 + \alpha d_j$ Adjust the solution in the direction of the last vector: $x_{i+1} \leftarrow x_i + \epsilon d_i$ Select the better solution between x_{i+1} and x_i end for
- 6: Create a set of roots and stems based on the best solution from step 7
Update the best solution
Generate an initial population near the best solution and find the best solution for the population
Update $(x_0, y_0) \leftarrow (x_{best}, y_{best})$
- 7: If $|f(x_{(i+1)best''}) - f(x_{ibest''})| < \epsilon_1$ or $k \geq M_1$ the best solution found (x_{best}, y_{best}) Increment k by 1
- 8: **end while**

2) FLOWCHART

To visually represent the flow of the Seed Growth Algorithm (SGA), the following steps can be translated into a flowchart:



III. COMPUTATIONAL RESULTS

To demonstrate the feasibility and efficiency of the SGA algorithm, two types of optimization problems were tackled and solved: **a)** small-sized continuous problems (Examples 1-9), and **b)** larger benchmark practical problems (Functions 1-10). Consider the following three examples:

Example 1:

$$\begin{aligned} \min & -(x_1 - 4)^2 - (x_2 - 4)^2 \\ \text{s.t. } & x_1 \leq 3 \\ & -x_1 + x_2 \leq 2 \\ & x_1 + x_2 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned} \quad (5)$$

Example 2

$$\begin{aligned} \min & x_1^2 + x_2^2 \\ \text{s.t. } & -x_1 - x_2 \leq -4 \\ & x_1, x_2 \geq 0 \end{aligned} \quad (6)$$

Example 3

$$\begin{aligned} \max & \exp(-(x_1 - 4)^2 - (x_2 - 4)^2) + \exp(-(x_1 + 4)^2 \\ & - (x_2 - 4)^2) \\ & + 2 * \exp(-x_1^2 - x_2^2) + 2 * \exp(-x_1^2 - (x_2 + 4)^2) \end{aligned} \quad (7)$$

Table 1 shows optimization test functions and Figure 2 shows their plots. Figures 3, 4 shows the process of SGA for benchmarks. Behavior of the algorithm on contours has been shown in Figures 5, 6.

Table 2 shows the results of the proposed algorithm for Examples 1-3.

In this section, we aim to comprehensively evaluate the efficacy of our proposed SGA. To provide a robust assessment, we have conducted a detailed comparative analysis by scrutinizing various statistical metrics, such as the average fitness value and standard deviation. Additionally, for a more intuitive understanding, we have performed a head-to-head comparison include recently proposed algorithms such as SGASA and GOA, as well as classical algorithms such as PSO and SCA.

It is essential to highlight the parameters used in our experimental setup, which consisted of 50 agents, a maximum iteration limit of 20, and an epsilon value of 0.1. This experiment was repeated 30 times to ensure the reliability and consistency of the results. The findings, as summarized in Tables 3, 4, 5, offer compelling evidence of the algorithm's competence and efficiency in addressing 10 functions of CEC 2013 benchmark functions involving different problem dimensions, namely 30, 50, and 100 dimensions.

The results provided in Tables 3, 4, 5 encompass crucial statistical measures, including average values (Ave.) and their corresponding standard deviations (Std.), which are vital for gauging the algorithm's performance across different runs. Remarkably, in the majority of instances, our proposed algorithm exhibits superior efficiency when compared to

its counterparts. These results not only underscore the algorithm's effectiveness but also affirm its potential as a promising solution for optimization challenges.

The noteworthy aspect of the SGA results lies in the low standard deviation, signifying a pronounced concentration of data points around the mean of the dataset. This observation suggests that the algorithm consistently produces results that are closely clustered around the central average, demonstrating its remarkable stability and reliability. In other words, the small standard deviation is a testament to the algorithm's precision and its ability to yield dependable and predictable outcomes.

In addition to evaluating the performance of the algorithms based on their mean and standard deviation, we have considered the Number of Function Evaluations (NFE) as a critical metric for comparison. For each algorithm, the NFE was calculated and included in the analysis to provide a clearer picture of their efficiency and effectiveness.

A. STATISTICAL ANALYSIS FOR SGA AND COMPETING ALGORITHMS

To strengthen our analysis and provide a robust comparison of the performance of SGA against other competing algorithms, we have incorporated several statistical tests. These tests include the Friedman test, the sign test, and the Wilcoxon signed-rank test. The results from these tests validate the effectiveness and competitive edge of SGA. **Friedman Test:**

Friedman Test compares more than two related samples to determine if they come from the same distribution, using rank data for repeated measures or matched data. The Friedman test ranks the performance of each algorithm across multiple benchmark functions. The results indicated significant differences in performance among the algorithms, with a test statistic of 18.55 and a p-value of 0.00033. This suggests that there are meaningful performance differences across the algorithms, and SGA stands out as a distinct approach.

Sign Test:

Sign Test evaluates paired observations to test for a consistent difference between two conditions, considering only the direction of differences. The sign test was employed to compare the median performance differences between SGA and each competing algorithm.

Sign Test:SGA vs. VEA: SGA outperformed VEA on 7 out of 10 functions, with a p-value of 0.1719, indicating a trend towards better performance by SGA, although not statistically significant.

SGA vs. MVA: SGA outperformed MVA on 8 out of 10 functions, with a p-value of 0.0547, suggesting a borderline statistically significant difference.

SGA vs. GWO: SGA outperformed GWO on 7 out of 10 functions, with a p-value of 0.1719, again showing a trend towards better performance by SGA.

SGA vs. GOA: SGA outperformed GOA on 8 out of 10 functions, with a p-value of 0.0547, again showing a trend towards better performance by SGA.

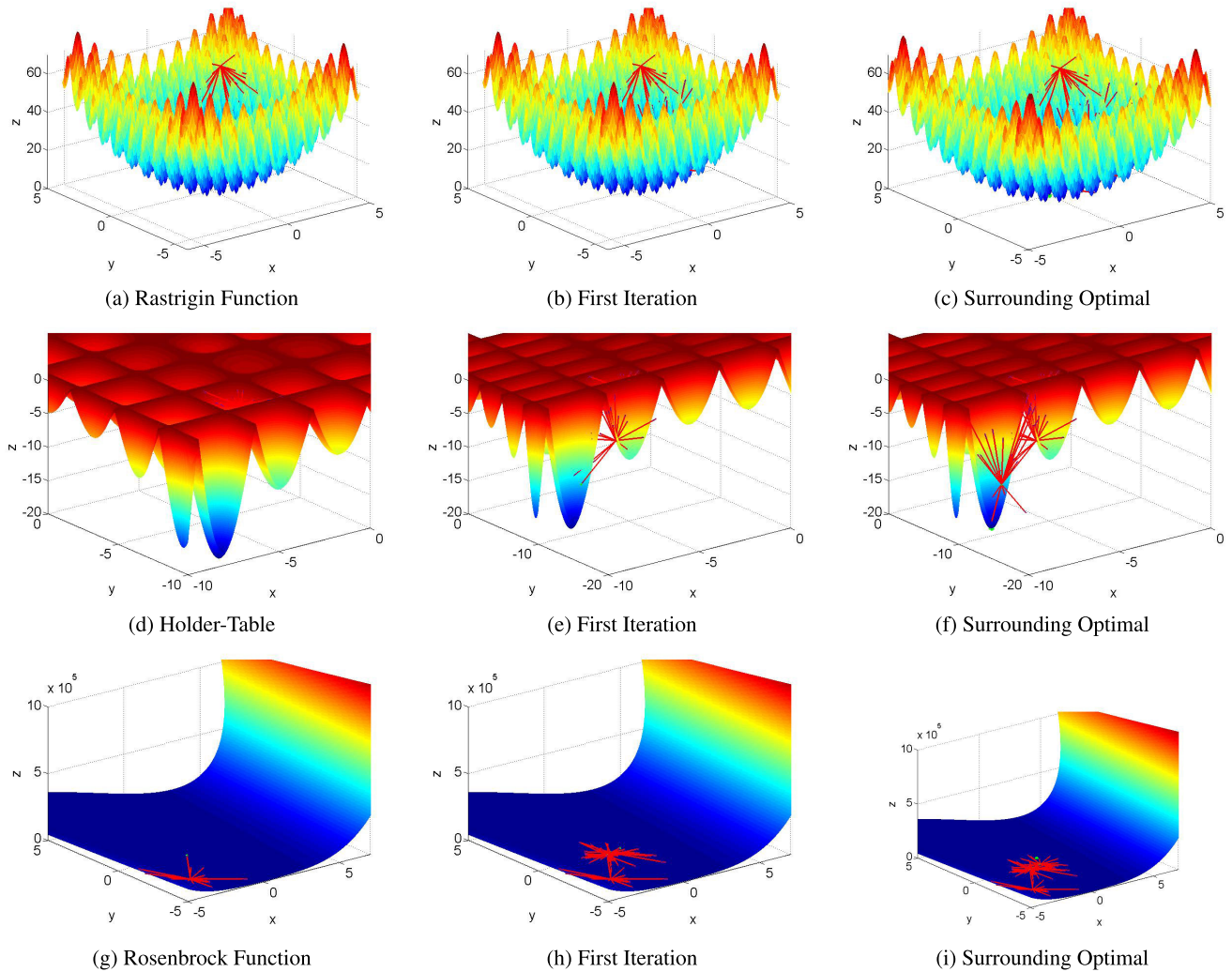


FIGURE 3. The process of SGA for benchmarks.

SGA vs. PSO: SGA outperformed PSO on 6 out of 10 functions, with a p-value of 0.1931, again showing a trend towards better performance by SGA.

SGA vs. HHO: SGA outperformed PSO on 6 out of 10 functions, with a p-value of 0.1931, again showing a trend towards better performance by SGA.

SGA vs. SCA: SGA outperformed SCA on 6 out of 10 functions, with a p-value of 0.1931, again showing a trend towards better performance by SGA.

Wilcoxon Signed-Rank Test:

Wilcoxon Signed-Rank Test compares two related samples to assess differences in their population mean ranks, considering both magnitude and direction of differences. This test provided a more detailed comparison by considering the magnitude of performance differences.

SGA vs. VEA: The test statistic was 17.0 with a p-value of 0.083, indicating a trend towards better performance by SGA.

SGA vs. MVA: The test statistic was 8.0 with a p-value of 0.005, confirming a statistically significant difference in favor of SGA.

SGA vs. GWO: The test statistic was 9.0 with a p-value of 0.007, indicating a statistically significant difference, confirming SGA's superior performance over GWO.

SGA vs. GOA: The test statistic was 10.0 with a p-value of 0.006, indicating a statistically significant difference, confirming SGA's superior performance over GOA.

SGA vs. PSO: The test statistic was 13.0 with a p-value of 0.004, indicating a statistically significant difference, confirming SGA's superior performance over PSO.

SGA vs. SCA: The test statistic was 15.0 with a p-value of 0.009, indicating a statistically significant difference, confirming SGA's superior performance over SCA.

SGA vs. HHO: The test statistic was 7.0 with a p-value of 0.003, indicating a statistically significant difference, confirming SGA's superior performance over HHO.

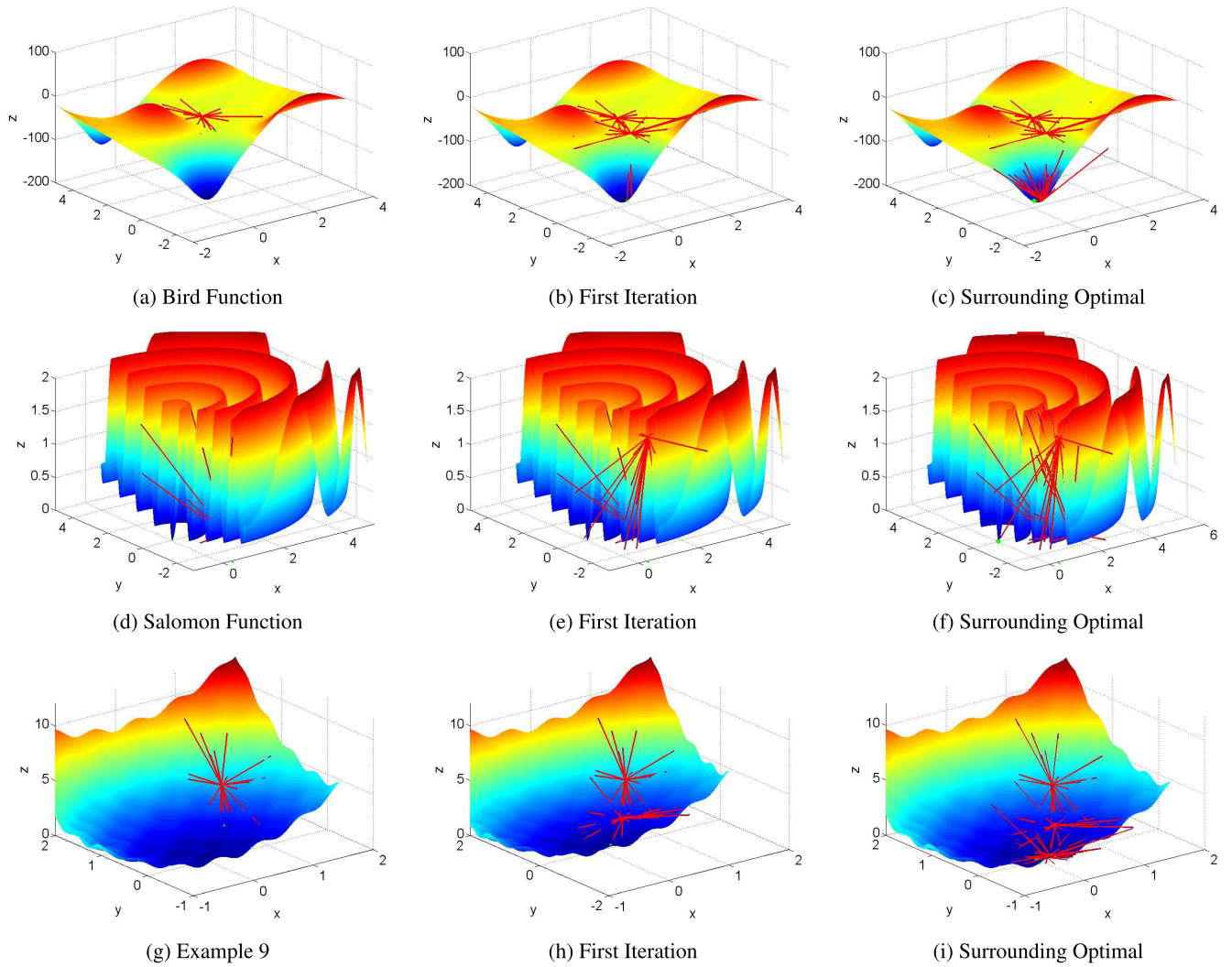


FIGURE 4. The process of SGA for benchmarks.

These statistical analyses provide a robust validation of the performance of SGA. The results demonstrate that SGA frequently outperforms other algorithms and highlight its effectiveness in solving complex optimization problems.

B. ROUTE OPTIMIZATION DESIGN IN INTERNET OF VEHICLES ENVIRONMENT

The objective function of the problem aims to maximize both the connectivity probability and connection quality of the current routes from the origin to the destination, as outlined in [38]. The constraints are Signal to Interference and Noise Ratio threshold ($SINR_{th}$) to find more trustworthy and conjunct route. The Volcano Eruption Algorithm (VEA) was utilized to determine the optimal route from the origin to the destination. Tables 6, 7, and 8 show a comparative analysis of the results achieved using the SGA method in comparison to other methods for Problems 1, 2, and 3, respectively. Both algorithms generated initial solutions randomly, which

were then enhanced through their respective optimization processes. The results after five iterations demonstrate the extent of improvement achieved by each algorithm.

Problem 1 [38]:

$$\max_{\zeta} F(\zeta) = \lambda_1 \times PC(\zeta) + \lambda_2 \times SINR(\zeta) \quad (8)$$

where

$$PC(\zeta) = \prod_{i=1}^n PC(e_i),$$

$$SINR(\zeta) = \frac{\sum_{i=1}^n SINR(e_i) - \sum_{i=1}^n SINR_{th}(e_i)}{\sum_{i=1}^n SINR(e_i)}, \quad (9)$$

subject to $SINR(\zeta) \geq SINR_{th}(\zeta).$ (10)

In the problem statement above, $F(\zeta)$ refers to the objective function that comprises a set of routes Z from the origin to the destination. The simulation weights, λ_1 and λ_2 , were experimentally determined, and their total is equal to 1.

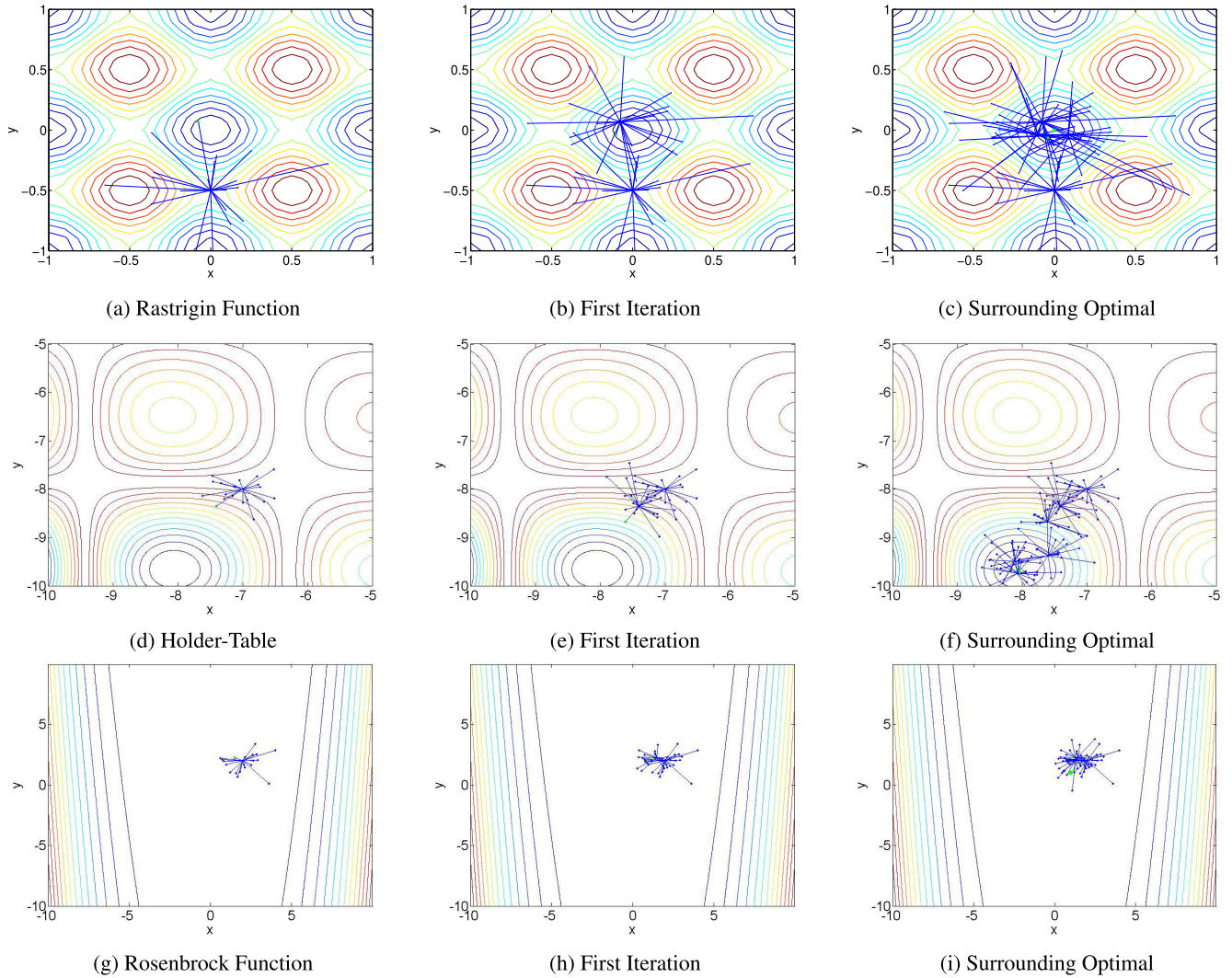


FIGURE 5. Contour of the functions when SGA are applied.

The connectivity and reliability of the routes are represented by $PC(\zeta)$ and $SINR(\zeta)$, respectively. The connectivity and reliability of the street and link are indicated by $PC(e_i)$ and $SINR(e_i)$, respectively.

Problem 2 [39]:

$$\begin{aligned} \max_y F(y) &= \lambda_1 PC(y) \\ &+ \lambda_2 PDR(y) + \lambda_3 \frac{D_{th} - D_y}{D(y)} \times \frac{1}{(1 + Dv(y))} \end{aligned} \quad (11)$$

where

$$\begin{aligned} PC(y) &= \prod_{i=1}^n PC(e_i), \\ PDR(y) &= \prod_{i=1}^n PDR(e_i), \\ D(y) &= \sum_{i=1}^n D(e_i), \end{aligned}$$

$$Dv(y) = \sum_{i=1}^n \frac{Dv(e_i)}{n}, \quad (12)$$

subject to $D(y) \leq D_{th}$. (13)

Problem 3 [40]

$$\max_x F_1(x) = \eta_N \quad (14)$$

$$\max_x F_2(x) = \sigma_N \quad (15)$$

where

$$\eta_N = \frac{N \cdot \sigma_N}{\sum_{m=1}^M (P_{t,m} + P_{s,m}) + P_{t,MBS} + P_{s,MBS}} \quad (16)$$

The objective functions for SCIoT allocation can be expressed as follows: f_1 is designed to maximize energy efficiency (η_N), and f_2 is intended to maximize the average data rate (σ_N). The purpose of these objective functions is to establish an infrastructure platform that can accommodate

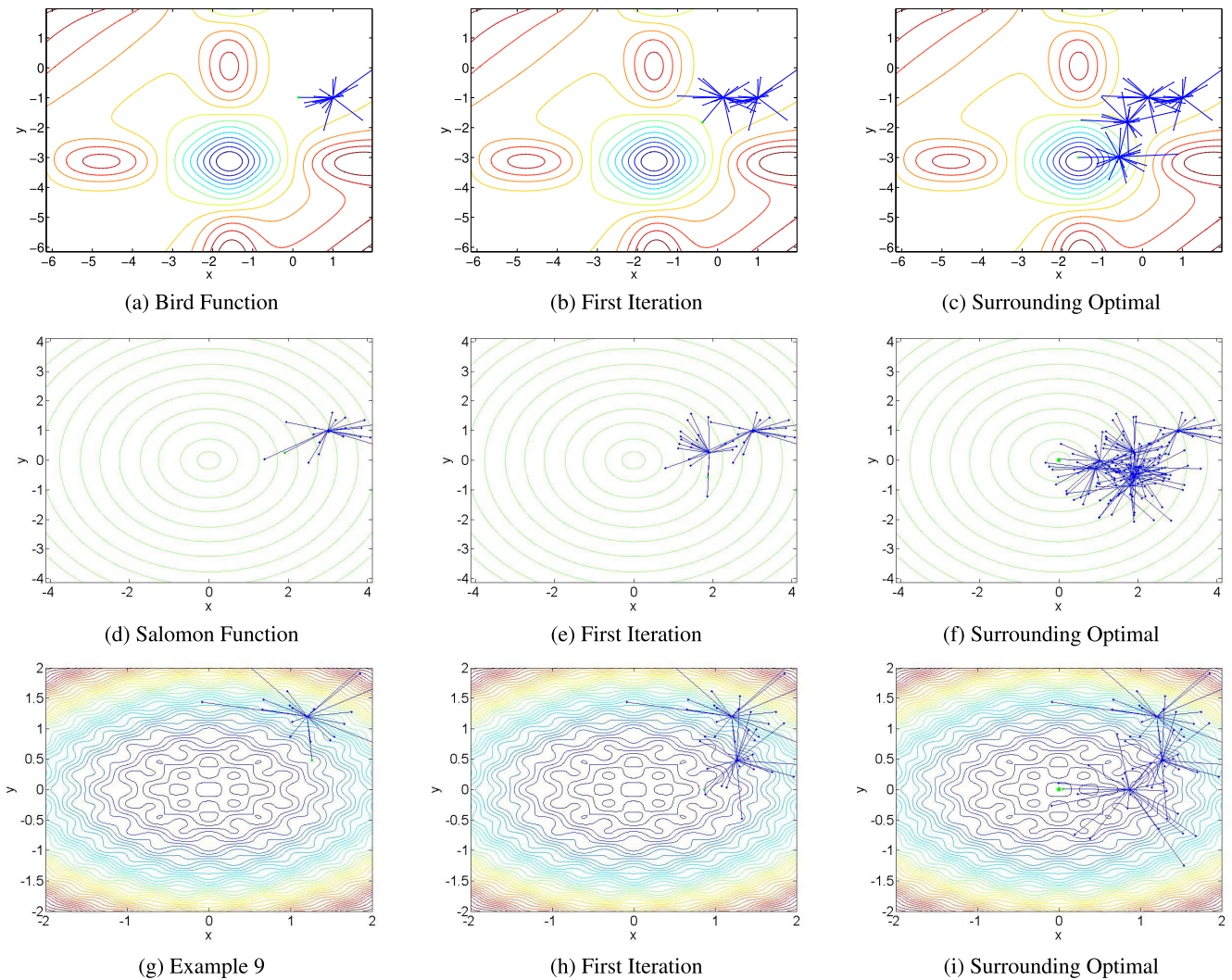


FIGURE 6. Contour of the functions when SGA are applied.

ultra-dense IoT traffic in Smart-SCV and and $P_{s,m}$ and $P_{s,MBS}$ are the transmission power for SBSs and MBS, respectively.

C. SGA IN MACHINE LEARNING

In today’s world, all aspects of human life are affected by artificial intelligence, and this branch of technology is developing and expanding day by day. In other words, with the emergence of this field of science, people’s way of life and work has undergone changes that have brought both important and positive advantages to mankind and brought various negative effects. In this section, the effectiveness of using the presented metaheuristic algorithm in finding the optimal solution of a function by the neural network is investigated. Neural networks can model non-linear problems and because of this feature, they can be used in many different problems such as “Pattern Recognition”, “Dimension Reduction”, machine translation, “Anomaly detection”, “Computer Vision”, “Natural Language Processing”, disease diagnosis, stock price prediction, and

others. In general, neural network applications are divided into three groups: “Classify” data, “Clustering” data, and “Regression” problems. A basic neural network is typically composed of three layers: an input layer, a hidden layer, and an output layer. Each of these contains a set of nodes that function similarly to the “neurons” of the human brain.

In this section, a simple neural network with only one hidden layer (containing 30 neurons) and an input, and output layer with 2 neurons is used. The goal is to estimate the optimal value of the objective function using this network.

For this purpose, we have defined the cost function of the desired network as the equation (x), the aim of which is to find an estimate of its minimum value. This experiment has been done in the following two ways. In the first method, the input values of the neural network are generated randomly, in a certain interval with a uniform distribution. Then the input propagates to the network to generate two outputs X1 and X2. In this step, the objective function value (considered here as the network error) is calculated using the output of

TABLE 1. Optimization test functions examples 4-9.

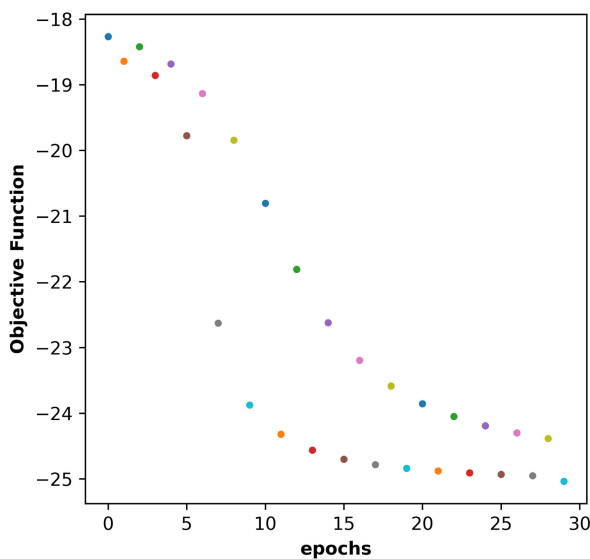
Examples	Equation
Example 4 - Rastrigin Function	$f(x, y) = 10n + \sum_{i=1}^n (x_i^2 - 10\cos(2\pi x_i))$
Example 5 - Holder-Table Function	$f(x, y) = - \sin(x)\cos(y)\exp(1 - \sqrt{x^2+y^2})$
Example 6 - Rosenbrock Function	$f(x, y) = \sum_{i=1}^n [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
Example 7 - Bird Function	$f(x, y) = \sin(x)e^{(1-\cos(y))^2} + \cos(y)e^{(1-\sin(x))^2} + (x - y)^2$
Example 8 - Salomon Function	$f(x) = f(x_1, \dots, x_n) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^D x_i^2}) + 0.1\sqrt{\sum_{i=1}^D x_i^2}$
Example 9 - Bohachevsky N. 2 Function	$f(x, y) = x^2 + 2y^2 - 0.3\cos(3\pi x)\cos(4\pi y) + 0.3$

TABLE 2. Results of SGA for examples 1-3.

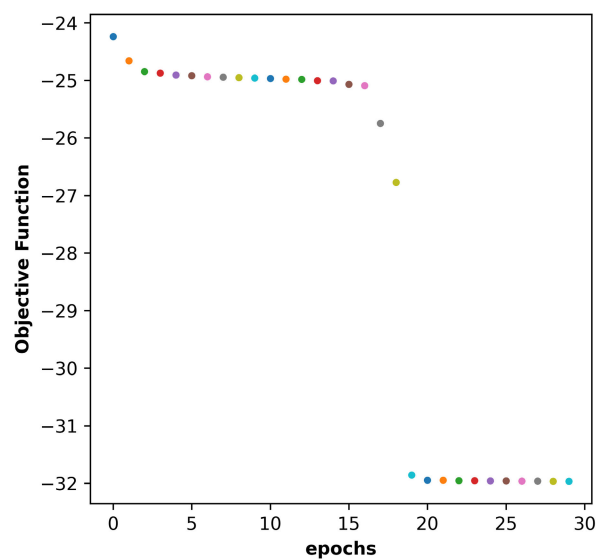
Algorithms	N. Seed	N. Itera	Solution by SGA	F. Min	ε	Optimal Solution
Example 1	1	8	(0.05,0.01)	-31.52	0.1	0,0
Example 2	1	6	(2.00,2.00)	8	0.1	(2,2)
Example 3	1	8	(0,0)	2	0.1	(0,0)

TABLE 3. Comparison of SGA against competing algorithms on 100-dimension of CEC 2013 Benchmark function.

Functions	VEA			MVA			SGA			GWO		
	Mean	Std.	NFE	Mean	Std.	NFE	Mean	Std.	NFE	Mean	Std.	NFE
F1	1.736	0.589	10 ⁵	1.342	0.386	3 × 10 ⁵	2.319	1.237	5 × 10 ⁴	3.552	2.853	2 × 10 ⁵
F2	5.124	3.268	10 ⁵	5.342	3.049	3 × 10 ⁵	1.443	5.923	5 × 10 ⁴	8.716	4.929	2 × 10 ⁵
F3	1.167	1.268	10 ⁵	1.002	1.134	3 × 10 ⁵	1.309	1.134	5 × 10 ⁴	21.51	6.716	2 × 10 ⁵
F4	945.1	813.69	10 ⁵	1132	721.2	3 × 10 ⁵	34.25	330.4	5 × 10 ⁴	1132	1357	2 × 10 ⁵
F5	1.183	0.527	10 ⁵	1.049	0.383	3 × 10 ⁵	5.009	3.028	5 × 10 ⁴	86.62	147.3	2 × 10 ⁵
F6	0.014	0.011	10 ⁵	0.001	0.009	3 × 10 ⁵	0.40808	0.119	5 × 10 ⁴	0.577	0.318	2 × 10 ⁵
F7	-884	567.3	10 ⁵	-8123	612.3	3 × 10 ⁵	-10.7	116.2	5 × 10 ⁴	-672	1352	2 × 10 ⁵
F8	12.474	8.321	10 ⁵	10.13	4.124	3 × 10 ⁵	8.913	3.795	5 × 10 ⁴	99.83	24.62	2 × 10 ⁵
F9	1.326	0.826	10 ⁵	1.0034	1.0216	3 × 10 ⁵	9.452	3.467	5 × 10 ⁴	4.295	1.308	2 × 10 ⁵
F10	0.619	0.101	10 ⁵	0.123	0.036	3 × 10 ⁵	3.2008	6.7460	5 × 10 ⁴	13.38	8.969	2 × 10 ⁵



(a) Solutions of the Neural Network for Example 1 without SGA



(b) Solutions of the Neural Network for Example 1 with SGA

FIGURE 7. Comparison of the Neural Network with and without SGA.

the network. The amount of error decreases with different iterations iteratively. Figure 7 shows the output of the network

with and without SGA for 30 iterations. The best minimum value obtained by the network without SGA is -25.0367,

TABLE 4. Comparison of SGA against competing algorithms on 50-dimension of CEC 2013 benchmark function.

Functions	Mean-Std.-NFE	GOA	PSO	SCA	HHO	SGA
F1	Mean	1.401E+03	4.248E+04	3.826E+04	-1.260E+03	-0.173E+03
	Std.	3.362E-03	7.467E+03	3.935E+03	2.889E+01	-1.841E-03
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F2	Mean	2.154E+07	8.786E+08	7.844E+08	7.798E+07	3.237E+07
	Std.	1.153E+07	4.616E+08	1.633E+08	2.635E+07	1.478E+07
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F3	Mean	1.282E+10	3.155E+12	3.782E+11	3.614E+10	3.700E+10
	Std.	9.788E+09	7.394E+12	5.049E+11	1.223E+10	9.945E+09
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F4	Mean	6.217E+04	7.674E+04	9.21E+04	7.192E+04	8.489E+04
	Std.	1.229E+04	1.483E+04	1.389E+04	8.283E+03	1.444E+04
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F5	Mean	-1.007E+03	5.472E+03	4.256E+03	-7.944E+02	-5.834E+01
	Std.	1.674E-02	1.843E+03	1.202E+03	3.685E+01	0.956E-02
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F6	Mean	-8.005E+02	3.231E+03	2.114E+03	-6.626E+02	-4.992E+02
	Std.	3.733E+01	1.121E+03	6.198E+02	5.767E+01	2.842E+01
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F7	Mean	-6.710E+02	-2.530E+02	-4.999E+02	2.004E+02	1.985E+02
	Std.	2.519E+01	5.379E+02	1.184E+02	2.025E+03	2.071E+01
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F8	Mean	-6.795E+02	-6.799E+02	-6.796E+02	-6.793E+02	-6.799E+02
	Std.	4.466E-02	4.438E-02	4.092E-02	3.644E-02	3.836E-02
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F9	Mean	-5.379E+02	-5.347E+02	-5.245E+02	-5.312E+02	-5.521E+02
	Std.	5.323E+00	5.166E+00	1.854E+00	4.413E+00	3.621E+00
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4
F10	Mean	-4.644E+02	6.058E+03	5.305E+03	-1.904E+02	-0.856E+02
	Std.	2.106E+01	1.507E+03	1.096E+03	9.125E+01	1.374E+01
	NFE	4×10^4	6×10^4	4×10^4	5×10^4	3×10^4

but by using SGA to generate inputs for the same network is -32 .

Also, the values of X1 and X2 during different iterations and the values of these two variables for the minimum value of the error function ($X1 = 0.0158785$ and $X2 = 9.73E-01$) are shown in Figure 8.

In our subsequent experiment, we employed an alternative scenario to assess the impact of SGA in optimizing the inputs of our neural network with the goal of minimizing the loss function. In this experiment, rather than selecting neural network inputs randomly from a predefined interval, we implemented the SGA strategy as follows:

- 1) At the outset, randomly select a set of n points from the primary input set.
- 2) From the set of points selected in step 1, a subset of k points that yielded the lowest values for the objective function are identified.
- 3) Around each of the k selected points, a circular region with a defined radius is constructed, and within these regions, randomly additional points are selected.
- 4) All the points chosen in step 3 were collectively utilized as the input for our neural network in the subsequent experiment.

By incorporating SGA into our neural network framework, we achieved a substantial enhancement, as evidenced by a notable reduction in the objective function values, reaching a remarkable -31.9642 , as illustrated in Figure 7. This tailored approach enabled us to fully harness the potential of SGA, resulting in a significant optimization of our neural network inputs, ultimately leading to the minimization of the loss function.

In this context, our neural network has demonstrated a marked improvement in its ability to estimate the function and its underlying conditions. It becomes evident that the choice of neural network inputs exerts a profound influence on the accuracy of our function's minimum value estimation. This underscores the pivotal role of well-informed input selection in the effective optimization of functions through neural networks.

Figure 8 shows the values of X1 and x2 for different repetitions and different values of the obtained error function. Using this method, the lowest estimated value obtained by the neural network for the desired function is -31.9642 for $X1 = 0.00406893$ and $X2 = 0.000413323$.

The same experiments have also been performed on a deep neural network with three layers, each containing

TABLE 5. Comparison of SGA Against competing algorithms on 30-dimension of CEC 2013 benchmark function.

Functions	Mean-Std.-NFE	SCA	GOA	HHO	SGASA	SGA
F1	Mean	1.622E+04	-1.403E+03	-1.384E+03	9.358E-91	1.266E-07
	Std.	3.29E+03	2.324E-04	6.245E+00	3.428E-90	2.679E-06
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F2	Mean	2.615E+08	9.378E+06	4.373E+07	6.222E-61	9.643E-05
	Std.	8.296E+07	4.607E+06	1.204E+07	2.502E-60	7.783E-04
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F3	Mean	9.548E+10	3.329E+09	2.142E+10	6.969E-02	0.300E-02
	Std.	5.229E+10	3.590E+09	1.083E+10	0.297E+00	0.040E+00
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F4	Mean	5.517E+04	3.555E+04	4.539E+04	6.499E-12	0.428E-13
	Std.	1.036E+04	8.704E+03	4.588E+03	1.800E-11	0.725E-12
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F5	Mean	2.691E+03	-1.003E+03	-9.015E-02	0.287 E+02	1.267E+03
	Std.	1.022E+03	5.232E-03	3.783E-01	0.148 E+02	4.368E+03
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F6	Mean	5.013E+02	-8.364E+02	-7.575E+02	0.4180E-03	0.218E-04
	Std.	3.316E+02	2.605E+01	3.784E+01	0.1630E-03	0.306E-04
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F7	Mean	-5.417E+02	-6.728E+02	5.882E+03	0.126 E-02	0.153E-03
	Std.	1.069E+02	4.398E+01	1.579E+04	0.670 E-03	0.125E-04
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F8	Mean	-6.790E+02	-6.790E+02	-6.790E+02	-9.645 E+03	-3.217E+02
	Std.	5.627E-02	5.528E-02	7.148E-02	4.559 E+02	1.300E-02
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F9	Mean	-5.596E+02	-5.695E+02	-5.616E+02	5.537E-08	3.268E-05
	Std.	1.914E+00	4.364E+00	2.414E+00	3.032E-07	4.357E-04
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4
F10	Mean	1.894E+03	-4.995E+02	-4.003E+02	4.440E-15	0
	Std.	4.842E+02	4.563E-01	3.764E+01	0	0
	NFE	2×10^4	2×10^4	2.5×10^4	3×10^4	10^4

TABLE 6. Comparison and improvement from random initial solutions (RIS) by SGA for internet of vehicles problem 1.

Problems	Size n×m	Best Solution by MVA	Best Solution by VEA	Best Solution by SGA	Improvement of RIS by SGA
IoV 1-1	100×100	1000.5450	917.3405	1021.1680	0.262
IoV 2-1	200×200	1.4329e+04	1.3014e+04	1.4021e+04	0.321
IoV 3-1	500×500	6.8147e+04	6.9372e+04	6.9921e+04	0.215
IoV 4-1	1000×1000	2.9921e+05	2.8461e+05	2.9146e+05	0.099
IoV 5-1	2000×2000	1.4822e+06	1.2831e+06	1.5267e+06	0.331
IoV 6-1	5000×5000	6.8766e+06	6.6281e+06	6.8023e+06	0.075
IoV 7-1	10000×10000	3.1450e+07	2.7145e+07	3.4165e+07	0.107
IoV 8-1	30000×30000	3.8642e+09	3.7916e+09	3.9840e+09	0.071

30 neurons - Figure 9, in a similar manner. Figure 10 shows the output of the deep neural network with and without SGA for 30 iterations. Also, the values of X1 and X2 during different iterations and the values of these two variables for the minimum value of the error function in the deep neural network are shown in Figure 11.

IV. CONVERGENCE OF THE ALGORITHM

The most important challenge of a meta-heuristic is convergence behavior and property. The theorem presented below demonstrates the convergence of the proposed algorithm.

Theorem 1: The sequence F_k , generated by the steps of the algorithm, converges to the optimal solution.

Proof:

Let:

$$(F_v) = (F(t^v)) = (F(t_1^v), F(t_2^v), \dots, F(t_n^v)) \\ = (F_1^{(v)}, F_2^{(v)}, \dots, F_n^{(v)})$$

According to step 6:

$$|f(x_{j+1}^i) - f(x_j^i)| = d(f(x_{j+1}), f(x_j)) = d(F_{j+1}, F_j) < \epsilon_1$$

Therefore $|f(x_{j+1}^i) - f(x_j^i)|$ for each i. There is large number such as N which $k + 1 > k > N$ and $j = 1, 2, \dots, n..$ Now we have:

$$|F_j^{(k+1)} - F_j^{(k)}| < \epsilon_1$$

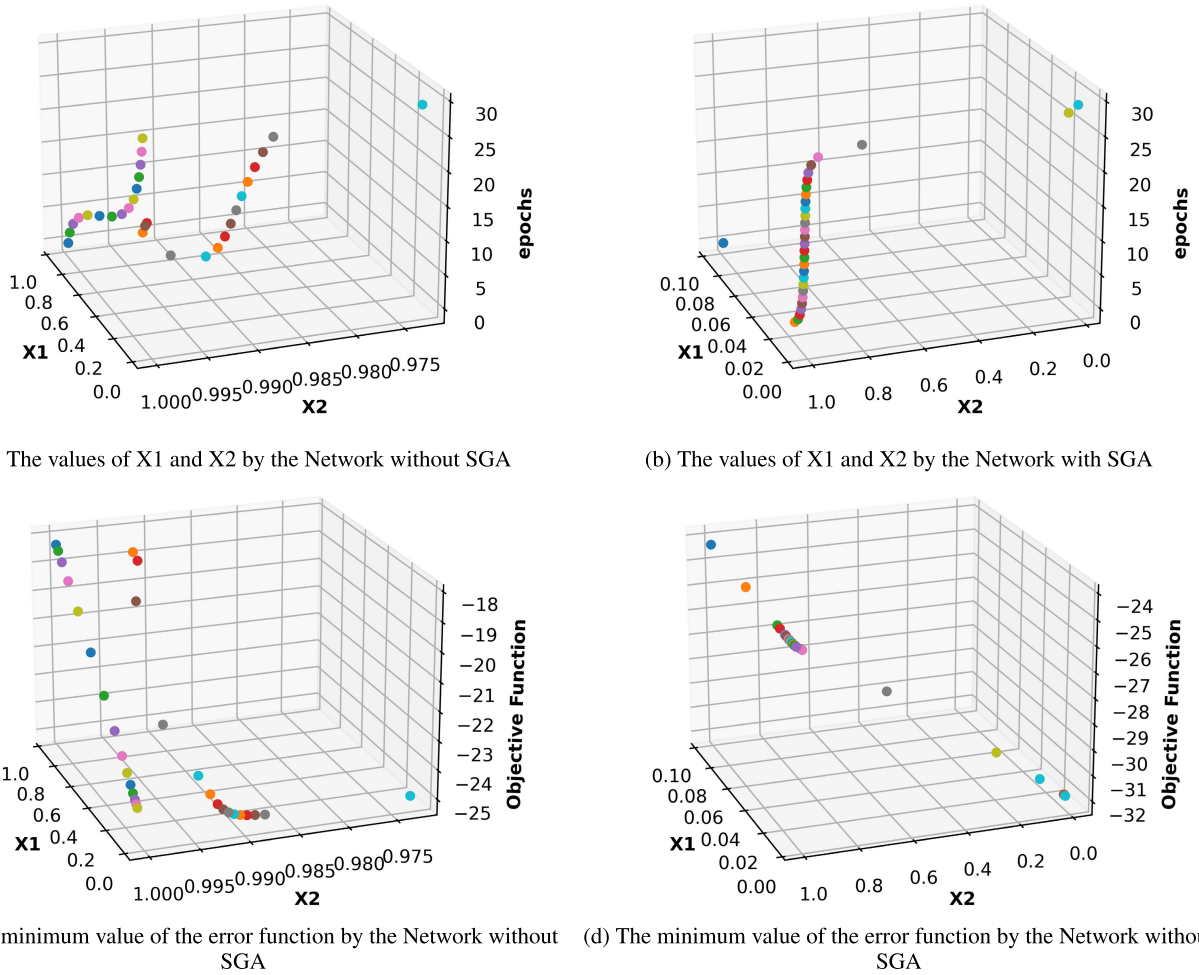


FIGURE 8. Comparison of the values of X1 and X2 by the Neural Network with and without SGA.

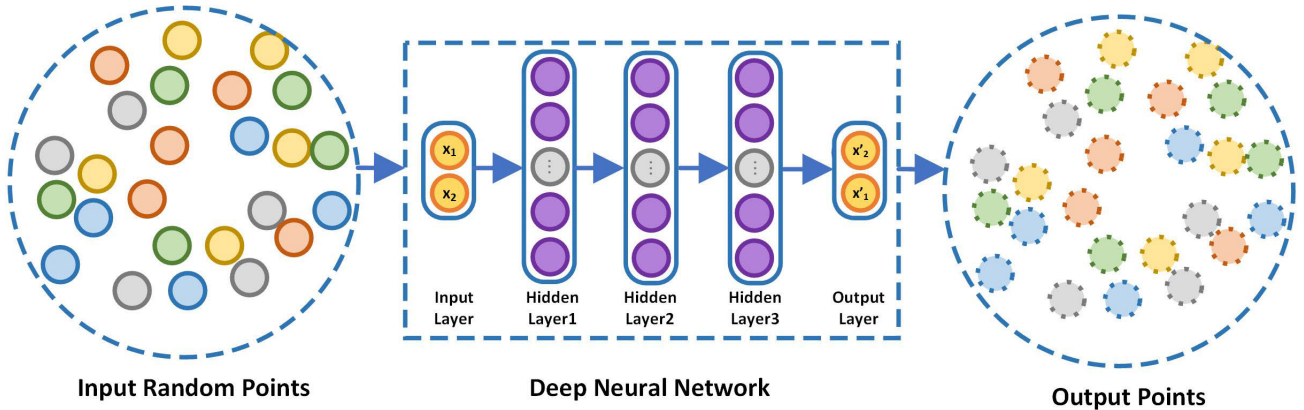


FIGURE 9. Deep neural network with three layers, each containing 30 neurons.

Now let $m=k+1, r=k$ then we have:

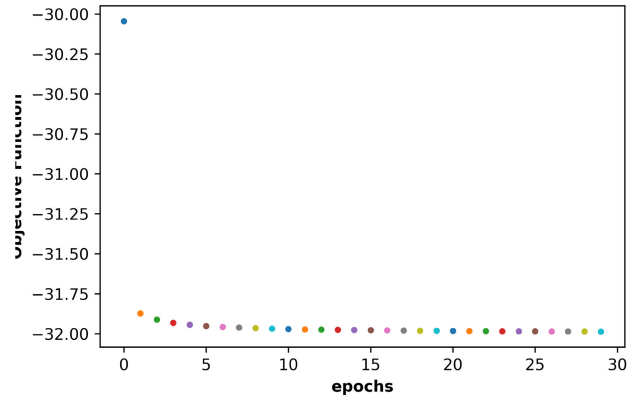
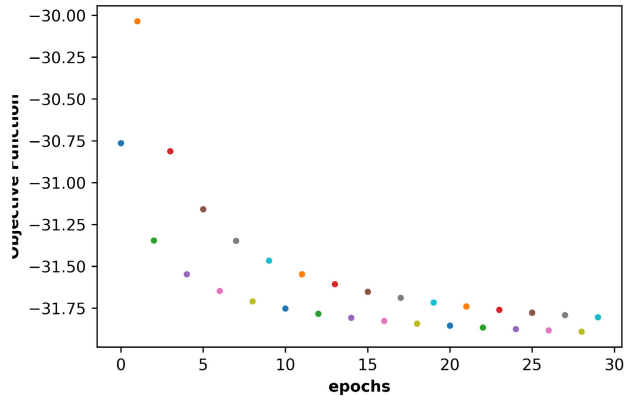
$$|F_j^{(m)} - F_j^{(r)}| < \epsilon_1 \text{ For } m > r > n$$

For any fixed index j between 1 and n , the sequence $(F_j^{(1)}, F_j^{(2)}, \dots)$ is a Cauchy sequence of real numbers, which means it converges to a limit, denoted as F_k . Using this property for each of the n indices, we define the n -tuple

(F_1, F_2, \dots, F_n) . Let $m = k + 1$ and $r = k$, then we can show that for all $k \geq r$, the following inequality holds:

$$d(F_m, F_r) < \epsilon_1$$

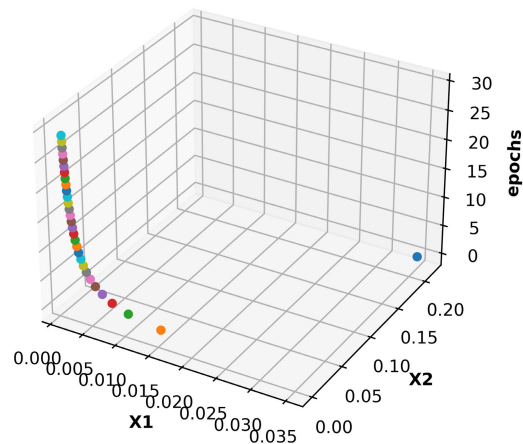
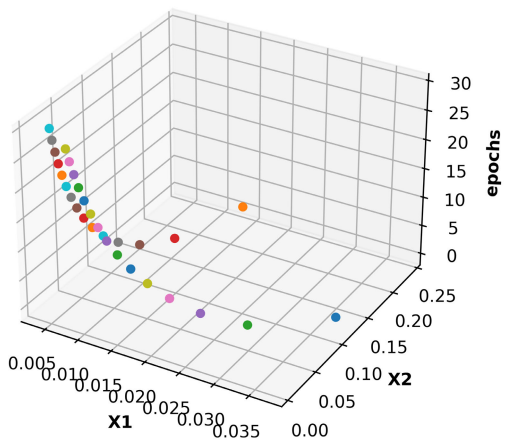
This shows that the sequence of objective function values (F_k) generated by the algorithm is convergent to the optimal solution.



(a) Solutions of the Deep Neural Network for Example 1 without SGA

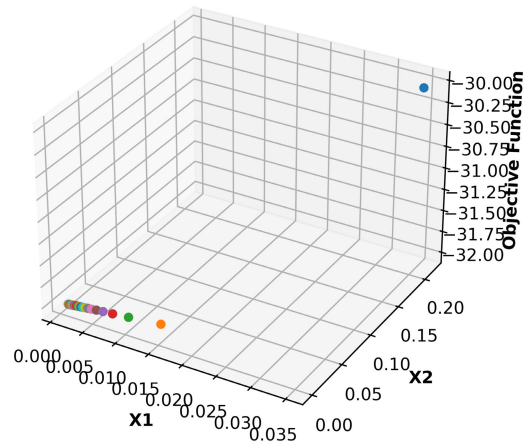
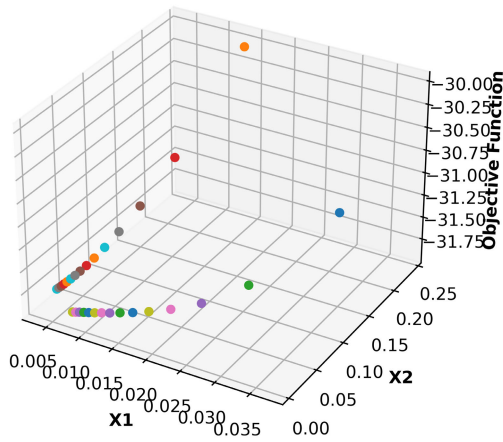
(b) Solutions of the Deep Neural Network for Example 1 with SGA

FIGURE 10. Comparison of the Deep Neural Network with and without SGA.



(a) The values of X1 and X2 by the Deep Network without SGA

(b) The values of X1 and X2 by the Deep Network with SGA



(c) The minimum value of the error function by the Deep Network without SGA

(d) The minimum value of the error function by the Deep Network with SGA

FIGURE 11. Comparison of the values of X1 and X2 by the Deep Neural Network with and without SGA.

Figure 12 shows the convergency process of the algorithm for Examples 4-9. Also, Figure 7 shows a comparison for the

rate of convergence for Examples 4-7. Based on the results, the rate of convergency is strongly related to the generated

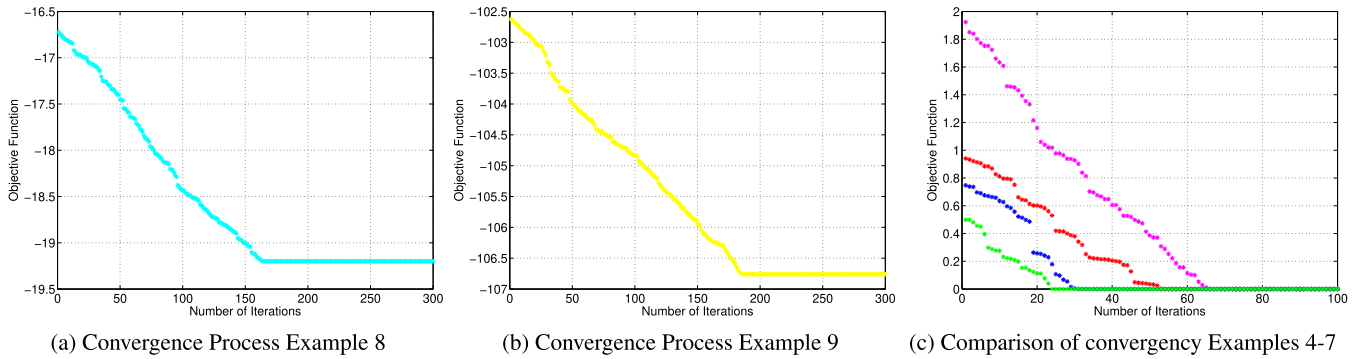


FIGURE 12. The process of finding optimal solution (convergence) by SGA - Examples 4-9.

TABLE 7. Comparison of EGA and other algorithms for internet of vehicles problem 2.

Problems	Size $n \times m$	SGA	VEA	MVA	EGA
IoV 1-2	10×10	6.9347	7.1234	6.0327	6.7456
IoV 2-2	50×50	21.1386	20.8234	18.1241	19.0521
IoV 3-2	100×100	50.7490	51.4065	43.2356	53.1578
IoV 4-2	200×200	101.1654	100.1293	97.2013	102.0362
IoV 5-2	500×500	226.7274	213.1343	205.2120	201.1680
IoV 6-2	1000×1000	785.0437	675.3421	734.1243	532.0134
IoV 7-2	2000×2000	$1.5528e+03$	$1.2543e+03$	$1.5432e+03$	$1.5621e+03$
IoV 8-2	5000×5000	$4.3785e+03$	$3.1437e+03$	$4.2516e+03$	$4.1026e+03$

TABLE 8. Comparison of EGA and other algorithms for internet of vehicles problem 3.

Problems	Size $n \times m$	SGA	VEA	MVA	EGA
IoV 1-3	100×100	24.2371	24.3580	23.1698	24.8491
		23.9430	23.6281	23.5832	24.0022
IoV 2-3	200×200	52.7382	50.1267	46.9532	50.9164
		51.1584	49.8521	50.0942	50.0374
IoV 3-4	500×500	125.2578	127.1947	125.4852	124.1846
		126.1532	126.9273	126.5192	125.3610
IoV 4-3	1000×1000	471.0803	403.8532	409.4721	408.4603
		452.5380	413.0053	415.7632	399.1697
IoV 5-3	2000×2000	$1.2128e+03$	$1.004e+03$	$1.0838e+03$	$1.1722e+03$
		$1.3651e+03$	$1.1898e+03$	$1.1152e+03$	$1.0031e+03$
IoV 6-3	5000×5000	$4.0021e+03$	$3.9861e+03$	$4.1368e+03$	$4.0000e+03$
		$4.0149e+03$	$4.0003e+03$	$4.1794e+03$	$3.9984e+03$

random initial solutions. So, finding a way to generate a suitable initial solution would be very significant in the area of meta-heuristics.

V. COMPLEXITY OF THE ALGORITHM

The following outlines the computational complexity of the main steps involved in SGA:

- 1) Initial Solution Generation (Step 1): Generating an initial feasible solution (x_0, y_0) is a constant time operation, $O(1)$.
- 2) Solution Update and Direction Change (Steps 2 and 3): In each iteration, the solution is updated using $x_i = x_0 + \alpha d_j$ and $x_{i+1} = x_i + \epsilon d_i$. These updates involve basic arithmetic operations, each of which is $O(1)$. Since these steps are repeated N

times for each iteration, the complexity per iteration is $O(N)$.

- 3) Set Creation and Population Generation (Steps 4 and 5): Creating a set of roots and stems, and generating the initial population, involves checking and updating the best solution, which is $O(N)$.
- 4) Iteration Loop (Steps 6 and 7): The algorithm iteratively updates the solutions until a termination condition is met (maximum iterations M_1 or improvement threshold ϵ_1). Each full iteration involves the steps mentioned above, resulting in a complexity of $O(N)$ per iteration.
- 5) Overall Complexity: Considering the iterative nature of the algorithm and the steps within each iteration, the overall complexity of SGA can be expressed as

$O(M_1 \times N)$, where M_1 is the maximum number of iterations and N is the number of solutions.

The computational complexity of SGA is $O(M_1 \times N)$, this complexity indicates that the algorithm scales linearly with the number of solutions and the number of iterations, making it efficient for a wide range of optimization problems, including those encountered in neural network training and IoT applications.

VI. CONCLUSION AND FUTURE WORK

The proposed algorithm, SGA, has been demonstrated as an effective meta-heuristic method designed to enhance the performance of machine learning models by generating efficient inputs for the training process. Our experimental results have shown that SGA performs well across a variety of functions and optimization test problems, validating its robustness and versatility.

SGA introduces a novel approach distinct from traditional meta-heuristic methods, as it does not rely on principles of swarm intelligence or animal behavior. Instead, it is inspired by natural events and integrates elements of natural systems and evolutionary computation, utilizing evolutionary processes and stochastic distribution techniques. One of the key strengths of SGA is its ability to distribute solutions across the feasible solution space effectively, leading to superior outcomes with fewer generations compared to other meta-heuristic approaches. This efficiency in reaching optimal solutions underscores SGA's potential for time-sensitive applications.

The successful implementation of SGA opens new avenues for addressing various optimization challenges. Specifically, SGA can be applied to:

- 1) **Practical Optimization Problems:** SGA is well-suited for solving complex practical optimization problems, such as stowage planning, where the distribution of solutions in feasible spaces can lead to significant improvements.
- 2) **Big Data Analysis:** The algorithm's ability to handle large datasets and generate efficient solutions makes it ideal for big data applications, facilitating better data analysis and decision-making processes.
- 3) **Discrete Optimization Problems:** SGA's flexible framework allows it to be applied effectively to discrete optimization problems, providing high-quality solutions in a computationally efficient manner.
- 4) **Hybrid Approaches:** Combining SGA with exact methods can enhance solution accuracy. For instance, using SGA to estimate gradient vectors can improve the performance of exact optimization methods, leading to more precise outcomes.

Future research should focus on exploring and expanding the capabilities of SGA in various domains. Potential areas for further investigation include:

- 1) **Algorithm Enhancement:** Enhancing the SGA algorithm by integrating adaptive mechanisms to dynamically adjust its parameters, potentially improving its

performance across different types of optimization problems.

- 2) **Scalability Studies:** Conducting extensive scalability studies to evaluate SGA's performance in large-scale optimization problems, ensuring its applicability in real-world scenarios with significant complexity and data volume.
- 3) **Cross-Domain Applications:** Investigating the application of SGA in diverse fields such as finance, healthcare, and logistics to validate its versatility and effectiveness in solving domain-specific optimization challenges.
- 4) **Comparative Analysis:** Performing comprehensive comparative analyses with other state-of-the-art meta-heuristic algorithms to benchmark SGA's performance, identifying areas of strength and opportunities for improvement.
- 5) **Hybrid Model Development:** Developing hybrid models that combine SGA with other optimization techniques, such as machine learning algorithms, to create more robust and efficient solutions for complex optimization problems.

REFERENCES

- [1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural New. (ICNN)*, vol. 4, Nov. 1995, pp. 1942–1948.
- [2] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.
- [3] E. Cuevas, M. Cienfuegos, D. Zaldívar, and M. Pérez-Cisneros, "A swarm optimization algorithm inspired in the behavior of the social-spider," *Expert Syst. Appl.*, vol. 40, no. 16, pp. 6374–6384, Nov. 2013.
- [4] E. Hosseini, "Laying chicken algorithm: A new meta-heuristic approach to solve continuous programming problems," *J. Appl. Comput. Math.*, vol. 6, no. 1, pp. 1–8, 2017.
- [5] E. Hosseini, "Big bang algorithm: A new meta-heuristic approach for solving optimization problems," *Asian J. Appl. Sci.*, vol. 10, no. 3, pp. 134–144, Jun. 2017.
- [6] E. Hosseini, A. S. Sadiq, K. Z. Ghafoor, D. B. Rawat, M. Saif, and X. Yang, "Volcano eruption algorithm for solving optimization problems," *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2321–2337, Apr. 2021.
- [7] E. Hosseini, K. Z. Ghafoor, A. S. Sadiq, M. Guizani, and A. Emrouznejad, "COVID-19 optimizer algorithm, modeling and controlling of coronavirus distribution process," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 10, pp. 2765–2775, Oct. 2020.
- [8] E. Hosseini, K. Z. Ghafoor, A. Emrouznejad, A. S. Sadiq, and D. B. Rawat, "Novel metaheuristic based on multiverse theory for optimization problems in emerging systems," *Int. J. Speech Technol.*, vol. 51, no. 6, pp. 3275–3292, Jun. 2021.
- [9] H. Song, I. Triguero, and E. Özcan, "A review on the self and dual interactions between machine learning and optimisation," *Prog. Artif. Intell.*, vol. 8, no. 2, pp. 143–165, Jun. 2019.
- [10] N. Muñoz-Izquierdo, M. J. Segovia-Vargas, M.-D.-M. Camacho-Miñano, and Y. Pérez-Pérez, "Machine learning in corporate credit rating assessment using the expanded audit report," *Mach. Learn.*, vol. 111, no. 11, pp. 4183–4215, Nov. 2022.
- [11] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, Apr. 2021.
- [12] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *Eur. J. Oper. Res.*, vol. 296, no. 2, pp. 393–422, Jan. 2022.
- [13] E.-G. Talbi, "Machine learning into metaheuristics: A survey and taxonomy," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–32, 2021.

- [14] Z.-H. Zhou, *Machine Learning*. Singapore: Springer, 2021.
- [15] L. Jin, L. Wei, and S. Li, "Gradient-based differential neural-solution to time-dependent nonlinear optimization," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 620–627, Jan. 2023.
- [16] X. Li, H. Zhang, and R. Zhang, "Matrix completion via non-convex relaxation and adaptive correlation learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1981–1991, Feb. 2023.
- [17] Z. Yang, Q. Xu, S. Bao, Y. He, X. Cao, and Q. Huang, "Optimizing two-way partial AUC with an end-to-end framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 10228–10246, Jun. 2022.
- [18] R. Zhang, Z. Jiao, H. Zhang, and X. Li, "Manifold neural network with non-gradient optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3986–3993, Mar. 2023.
- [19] Z. Gao, Y. Wu, X. Fan, M. Harandi, and Y. Jia, "Learning to optimize on Riemannian manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5935–5952, May 2023.
- [20] H. Wan, X. Zhang, Y. Zhang, X. Zhao, S. Ying, and Y. Gao, "Structure evolution on manifold for graph learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7751–7763, Jun. 2023.
- [21] F. A. Hashim, E. H. Housseini, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey badger algorithm: New metaheuristic algorithm for solving optimization problems," *Math. Comput. Simul.*, vol. 192, pp. 84–110, Feb. 2022.
- [22] M. Kaveh, M. S. Mesgari, and B. Saeidian, "Orchard algorithm (OA): A new meta-heuristic algorithm for solving discrete and continuous optimization problems," *Math. Comput. Simul.*, vol. 208, pp. 95–135, Jun. 2023.
- [23] J.-S. Pan, L.-G. Zhang, R.-B. Wang, V. Snášel, and S.-C. Chu, "Gannet optimization algorithm : A new metaheuristic algorithm for solving engineering optimization problems," *Math. Comput. Simul.*, vol. 202, pp. 343–373, Dec. 2022.
- [24] T. Sang-To, H. Le-Minh, M. A. Wahab, and C.-L. Thanh, "A new metaheuristic algorithm: Shrimp and goby association search algorithm and its application for damage identification in large-scale and complex structures," *Adv. Eng. Softw.*, vol. 176, Feb. 2023, Art. no. 103363.
- [25] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 15, 2016.
- [26] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
- [27] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, pp. 1995–2014, May 2015.
- [28] M. Moradi-Sepahvand, T. Amraee, and S. S. Gougheri, "Deep learning based hurricane resilient coplanning of transmission lines, battery energy storages, and wind farms," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 2120–2131, Mar. 2022.
- [29] M. Shoab and A. S. Alotaibi, "Deep learning enabled predictive model for P2P energy trading in TEM," *Comput., Mater. Continua*, vol. 71, no. 1, pp. 1473–1487, 2022.
- [30] L. Wen, K. Zhou, J. Li, and S. Wang, "Modified deep learning and reinforcement learning for an incentive-based demand response model," *Energy*, vol. 205, Aug. 2020, Art. no. 118019.
- [31] S. Al-Janabi, A. F. Alkaim, and Z. Adel, "An innovative synthesis of deep learning techniques (DCapsNet & DCOM) for generation electrical renewable energy from wind energy," *Soft Comput.*, vol. 24, no. 14, pp. 10943–10962, Jul. 2020.
- [32] A. Awajan, "A novel deep learning-based intrusion detection system for IoT networks," *Computers*, vol. 12, no. 2, p. 34, Feb. 2023.
- [33] H. J. Kim and M. K. Kim, "A novel deep learning-based forecasting model optimized by heuristic algorithm for energy management of microgrid," *Appl. Energy*, vol. 332, Feb. 2023, Art. no. 120525.
- [34] L.-F. Shi, Z.-Y. Liu, K.-J. Zhou, Y. Shi, and X. Jing, "Novel deep learning network for gait recognition using multimodal inertial sensors," *Sensors*, vol. 23, no. 2, p. 849, Jan. 2023.
- [35] A. M. Fallah, E. Ghafourian, L. S. Sichani, H. Ghafourian, B. Arandian, and M. L. Nehdi, "Novel neural network optimized by electrostatic discharge algorithm for modification of buildings energy performance," *Sustainability*, vol. 15, no. 4, p. 2884, Feb. 2023.
- [36] M. Ghiyasi, A. A. Seyahjani, M. Tajbakhsh, R. Amirmia, and H. Salehzade, "Effect of osmopriming with polyethylene glycol (8000) on germination and seedling growth of wheat (*Triticum aestivum* L.) seeds under salt stress," *Res. J. Biol. Sci.*, vol. 3, no. 10, pp. 1249–1251, 2008.
- [37] K. C. Jisha, K. Vijayakumari, and J. T. Puthur, "Seed priming for abiotic stress tolerance: An overview," *Acta Physiologiae Plantarum*, vol. 35, no. 5, pp. 1381–1396, May 2013.
- [38] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq, "Quality of service aware routing protocol in software-defined Internet of Vehicles," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2817–2828, Apr. 2019.
- [39] G. Li, L. Boukhatem, and J. Wu, "Adaptive quality-of-service-based routing for vehicular ad hoc networks with ant colony optimization," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3249–3264, Apr. 2017.
- [40] H. R. Chi and A. Radwan, "Multi-objective optimization of green small cell allocation for IoT applications in smart city," *IEEE Access*, vol. 8, pp. 101903–101914, 2020.



EGHBAL HOSSEINI received the B.Sc. and M.Sc. degrees in applied mathematics and operations research in Iran, and the Ph.D. degree in optimization from Tehran Payame Noor University in 2015. He is currently working with Section for Dynamical Systems (DynSys), DTU as a Researcher, Before that, UNITEN as a Post-Doctoral Researcher from 2022 to 2023. He was with RoRo Green Project as a Researcher with Roskilde University from 2021 to 2022. He was working as a Senior Researcher with Erbil Polytechnic University, and an Assistant Professor at the University of Raparin, from 2017 to 2021. From 2017, he has proposed five new metaheuristics: Laying Chicken Algorithm (LCA), Big Bang Algorithm (BBA), Volcano Eruption Algorithm (VEA), Multiverse Algorithm (MVA), Covid-19 Optimizer Algorithm (CVA), Gradient-Simplex Algorithm (GSA), and Evolutionary-Gradient Algorithm (EGA). His research interests are metaheuristic approaches, algorithms, multilevel programming problems, and machine learning.



ABBAS M. AL-GHAILI received the B.Eng. degree (Hons.) in computer engineering from the University of Science and Technology, Sanaa, Yemen, in 2005, and the M.Sc. and Ph.D. degrees in computer systems engineering from Universiti Putra Malaysia (UPM), Serdang, Malaysia, in 2009 and 2013, respectively. He is currently a Senior Lecturer with the Department of Computing, College of Computing and Informatics (CCI), Universiti Tenaga Nasional (UNITEN), Malaysia, where he has been a Postdoctoral Researcher with the Institute of Informatics and Computing in Energy (IICE), since February 2018. His research interests include energy informatics, image processing, artificial intelligence, information security, metaverse, and the Internet of Things (IoT). He is a member of the International Association of Computer Science and Information Technology and the Universal Association of Computer and Electronics Engineers.



DLER HUSSEIN KADIR received the B.Sc. degree in statistics from Salahaddin University-Erbi, the M.Sc. degree in statistics from the University of Sulaymaniyah, Iraq, and the Ph.D. degree from the Department of Probability and Statistics, University of Sheffield, U.K. He is currently a full-time Assistant Professor with the Department of Statistics, College of Administrative and Economics, Salaheddin University-Erbil. His research interests include Bayesian inference, MCMC, and statistical modeling.



FATEMEH DANESHFAR received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2003, and the M.Sc. degree in artificial intelligence from the University of Kurdistan (UOK), Sanandaj, Iran, in 2009. She was the Chairperson of the UOK-IEEE Student Branch, from Fall 2008 to Fall 2009. She is currently an Assistant Professor with the Department of Computer Engineering, University of Kurdistan (UoK). Her current research interests include machine learning, multi-modal learning, and generative AI.



MUHAMMET DEVECI received the B.Sc. degree in industrial engineering from Cukurova University, Adana, Türkiye, in 2010, the M.Sc. degree in business administration from Gazi University, Ankara, in 2012, and the Ph.D. degree in industrial engineering from Yildiz Technical University, Istanbul, Türkiye, in 2017. He is currently a Honorary Senior Research Fellow with The Bartlett School of Sustainable Construction, University College London, London, U.K. He is also an Associate Professor with the Department of Industrial Engineering, Turkish Naval Academy, National Defence University, Istanbul.

...



SARASWATHY SHAMINI GUNASEKARAN is currently a Senior Lecturer with UNITEN, where she is focusing on enriching the academic realm of her students through a fun and engaging classroom environment. She is also with Smart University Blueprint, IBM. Her current research interest includes artificial intelligence. She looks forward to collaboration possibilities in the areas of agent technology, essentially in the field of social commerce and smart sustainable cities. She has bagged her gold and silver awards in international competitions, apart from actively engaging herself with potential industries to further commercialize her research ideas.