SIRO: A Deep Learning-Based Next-Generation Optimizer for Solving Global Optimization Problems

Olaide N. Oyelade[1], Absalom E. Ezugwu[2(✉)], and Apu K. Saha[3]

[1] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, UK

[2] Unit for Data Science and Computing, North-West University, 11 Hoffman Street, Potchefstroom 2520, South Africa
absalom.ezugwu@nwu.ac.za

[3] Department of Mathematics, National Institute of Technology Agartala, Agartala 799046, Tripura, India

Abstract. This paper introduces the SIR Optimizer (SIRO), a novel next-generation learned metaheuristic algorithm inspired by biological systems and deep learning techniques. The optimizer uses the susceptible-infected-removed (SIR) epidemiological model to predict the population's susceptibility, active infections, and recoveries. To enhance the search process, SIRO incorporates deep learning into its initialization and parameter tuning components, enabling intelligent and autonomous behaviour. By generating initial solutions based on neural models, the algorithm achieves efficient, effective, and robust search outcomes. To validate the effectiveness of SIRO, a set of numerical hybrid test functions from the CEC 2017 benchmark, each characterized by 30 dimensions were utilized. The experimental results were compared against various state-of-the-art algorithms, demonstrating that SIRO outperforms its competitors. Moreso, it delivers high-quality solutions while utilizing fewer control parameters. The incorporation of a learning process in SIRO leads to superior precision and computational efficiency compared to other optimization approaches in the existing literature.

Keywords: SIR-model · SIRO · optimization algorithms · bio-inspired computing · deep learning · machine learning

1 Introduction

In recent years, there has been a growing research interest in the development of efficient, effective, and robust global search techniques for solving numerical and combinatorial optimization problems [1]. However, the complexity of real-world optimization problems has significantly increased over the past decades due to advancements in industrial processes and societal evolution, posing a challenge for existing optimization techniques, particularly classical methods [2]. The literature suggests that these optimization techniques have limitations in performing intelligent and robust search operations within

M. Sevaux et al. (Eds.): MIC 2024, LNCS 14753, pp. 45–61, 2024.
https://doi.org/10.1007/978-3-031-62912-9_6

problem-solution search spaces [3]. Nevertheless, the optimization community continues to propose new design variants and implementations of optimization techniques to address these challenges [4].

Metaheuristic algorithms have gained popularity and acceptance as the preferred optimization technique, despite their inability to generate precise or exact solutions for candidate optimization problems [5]. While these algorithms provide approximate solutions, they suffer from the requirement of problem-specific information or techniques, lack of guaranteed optimality in terms of convergence, absence of a theoretical or mathematical basis, reliance on multiple search parameters, stochastic search processes resulting in different solutions for the same problem, and the need for stopping criteria declaration. However, metaheuristics offer a valuable alternative to exact or mathematical optimization methods due to their flexibility, global optimization capability, robustness to problem size and randomness, and practical applicability to challenging real-world problems. These algorithms have also found widespread applications in engineering design problems, deep learning parameter optimization, facility layout problems, medical image segmentation and classification, parallel machine scheduling, and many others [6].

Despite the existence of numerous state-of-the-art optimization algorithms in the literature, the pursuit of new optimization methods remains constant. This need for new metaheuristic algorithms is driven by the no-free lunch (NFL) theorem of optimization, as proposed by Wolpert and Macready [7]. The NFL theorem suggests that there is no single optimization algorithm that works best for all optimization problems. In other words, there is no "free lunch" or universally superior optimization strategy. Consequently, the design of a universal, general-purpose optimization strategy is deemed impossible. This realization has motivated optimization experts to develop new algorithms.

From an intelligent and learning system perspective, the focus has shifted towards the development of more intelligent metaheuristic algorithms capable of learning from historical datasets. Extensive research on classical metaheuristic algorithms has highlighted that these algorithms generate substantial volumes of datasets during the solution search process. These datasets may contain valuable knowledge regarding the properties of good and bad solutions, the performance of different operators at different stages of the search process, and the precedence of search operators [5]. Surprisingly, classical optimization algorithms have not effectively utilized the knowledge hidden within these generated datasets.

Recent studies have demonstrated that machine learning (ML) techniques can complement metaheuristics by extracting useful knowledge from the generated data throughout the search process [6]. Integrating such knowledge into the search process can guide metaheuristics to make more informed decisions, enhancing their intelligence and significantly improving solution quality, convergence rate, and robustness. Thus, the present study proposes a new metaheuristic optimization algorithm for solving both single-objective and multi-objective problems. Notably, no classical algorithm has employed ML techniques in its initial algorithmic design stage.

In this study, a novel next-generation metaheuristic optimization algorithm called SIRO is proposed. Inspired by the SIR (susceptible-infected-removed) epidemiological model, the algorithm mimics the propagation of disease to guide the search process. A

machine learning component is integrated into SIRO to facilitate the intelligent automation of the algorithm's initialization and parameter configuration settings. This learning component assists the algorithm in selecting optimal initial high-quality solutions, thereby guiding an intelligent search process within the solution search spaces. The SIRO algorithm was evaluated using various numerical test functions. To validate the superior performance of SIRO, its results were compared against other state-of-the-art metaheuristic techniques.

2 Model Description

This section presents an overview of the proposed SIRO algorithm, including its inspiration and the optimization modeling design steps. Moreover, the section is divided into two main subsections for a smooth presentation of the various aspects of SIRO: the modeling source of inspiration for the SIRO from the popular SIR model is first discussed, followed by a description of the SIRO's implementation.

2.1 SIRO Algorithm Modelling

In this section, the design and model formulation describing the complete procedure for the proposed SIRO algorithm are discussed. An optimization process of SIRO is first described using a SIR-based model comprising the three compartments. Secondly, the mathematical model of the SIRO method is outlined from the concept of population initialization to the update of the compartment. Furthermore, a pseudocode describing the algorithmic representation of SIRO and an analysis of the algorithm's complexity is also presented in this section. Also, the neural learning method for improving the population and optimization of the parameter combination is described.

The SIR model applied in this study follows the classical model for modeling disease propagation so that all redundant compartments are eliminated. In Fig. 1, the combined representation of the SIR model and its optimization process are illustrated. The three compartments captured by the model include susceptible (S), infected (I), and recovered (R) individuals. The initial population of the SIR model is assigned to the S compartment since all contagious diseases target the susceptible population to generate the infected population. Considering the generic nature of the classical SIR model, the propagation pattern of each disease determines the assignment of individuals to the I compartment. In this study, the transition of individuals from S to I is conditioned on exposure to disease at a given rate denoted by $S\pi$. Conditions, such as immunity, vaccination, hospitalization, treatment, and self-recovery, have often reassigned individuals from the I compartment to the R compartment. The movement of individuals to R follows the order of Ig, which signifies recovery from the disease. At some point, individuals in R transition to S, thereby making them susceptible individuals that might be recruited into I unless there is a high influence of immunity that mitigates contagion.

The total duration T for a disease outbreak can be denoted into some $t_1, t_2, \ldots t_n$, which is represented in the figure as $t = 0,\ t = 1,\ t = 2 \ldots t = n$ where t_0 or $t = 0$ is a notation for showing when the outbreak of the disease is reported in a given population P. For the SIR model in this study, it is assumed that the population size of P is kept at

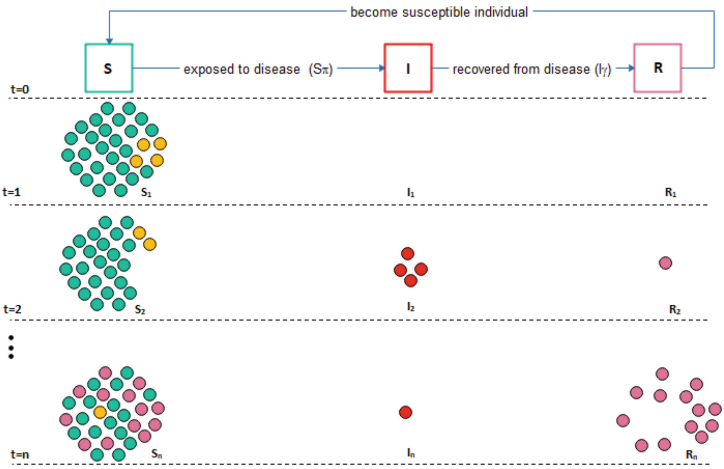

Fig. 1. Illustration of the SIR model in the process of solving optimization problems.

constant c with the implication that birth or death are variables or events that are frozen until duration T is completed. This keeps the size of the population constant, where the sum of individuals includes those in S, I, and R. The optimization process of the SIR models therefore follows that at t_0 or $t = 0$, all individuals in P are represented in S ($size(P) = size(S)$). At time $t = 1$, some individuals in S have been exposed to the disease and now can be infected by the virus, causing the disease after some duration, which is often peculiar to the nature of the disease. At $t = 2$, it can be observed that the exposed individuals in S have now been infected, leading to their reassignment to the I compartment. Meanwhile, at the same time, it is reported that one of the infected cases has recovered and is now assigned to the R compartment. As T increases, the reassignment of individuals across the compartment generates a dynamic process with each instance of the processes having a formation of allocations to S, I, and R with different patterns. The optimization process is resumed with the continuous anatomical changes of each individual whose mutation and displacement within the search space present a useful search pattern.

Table 1 presents an outline of all the parameters and their corresponding definition used to describe the SIR model. We take note of the infection rate π, recovery rate g, contact rate β with infected individuals, and natural death rate G of a population.

Considering the generic nature of the classical SIR model proposed in this study, setting specific values for each of the parameters is not applicable. Moreover, using stochastic models to assign values to the parameters assumes the popular approach used in literature. In this study, we proposed a novel machine learning-based model for the selection and combination of parameter values for all parameters used for the proposed SIRO algorithm. Meanwhile, the mathematical model of the algorithm is first presented and then a description of the learning-based method is also discussed in the following paragraphs.

Table 1. Symbols and definitions of the SIR model parameters

Symbols	Descriptions
π	Infection rate
γ	Recovery rate
Γ	Disease-induced death rate
τ	Natural death rate
β_1	Contact rate of infected
β_2	Contact rate of recovered

2.2 SIRO Model

The entire population of the SIR model is represented in Eq. (1), where at any time t_i, the summation of all individuals in the S, I, and R yields the total population:

$$P = S + I + R \tag{1}$$

We note that the generation of individuals into the S, I, and R compartments is dependent on a system differential equation consisting of three sub-models, as captured in Eqs. (2), (3), and (4) respectively:

$$S_t = \frac{\partial S(t)}{\partial t} = S\pi - \mathrm{I}\beta \tag{2}$$

$$I_t = \frac{\partial I(t)}{\partial t} = I + \mathrm{S}\beta - I\mathrm{g} \tag{3}$$

$$R_t = \frac{\partial R(t)}{\partial t} = R + \mathrm{g}I \tag{4}$$

where S_t, I_t, and R_t are the computed number of individuals allocated to S, I, and R at an arbitrary time t_i, respectively. Updates to these compartments are achieved during every iteration phase of the algorithm during the training or optimization process. Each time there is an update, it implies that individuals have been reassigned to different compartments, and therefore, there is a mutation of individuals according to the operation associated with a compartment. In the population initialization phase, all individuals are appropriated a certain composition as discussed in the following paragraph.

2.3 Basic and Neural Network-Based Initialization Methods

The population initialization and representation for population-based optimization algorithms often assume a stochastic approach or use some recent methods reported in the literature. In this study, a stochastic population initialization method is evolved to incorporate a deep learning approach. First, the population is initialized and represented by the matrix representation in Eq. (5) using a two-dimensional approach, where d is equivalent to the dimension of the optimization problem:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d-1} & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d-1} & x_{2,d} \\ & & \vdots \; \vdots \; x_{i,j} \; \vdots \; \vdots & & \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d-1} & x_{n,d} \end{bmatrix} \tag{5}$$

The computation for each x_i in X is achieved using the stochastic method represented in Eq. (6), where U and L respectively denote the upper and lower bounds typical in optimization problems with the optimization problem:

$$x_i = L + rand\,(0,1) * (U - L) \tag{6}$$

The resulting initial solution in X was applied to train a related biology-based and disease propagation optimization algorithm to obtain some final solution. The trajectory of solutions obtained was then curated and applied to train a deep learning model so that the model can learn a suitable solution space that can yield a potential initial solution for training the proposed SIRO algorithm. A long short-term memory (LSTM), which is a kind of recurrent neural network (RNN) and generally classified as deep learning (DL), was considered for this task of learning to generate a suitable initial search space. The architectural representation of LSTM is shown in Fig. 2.

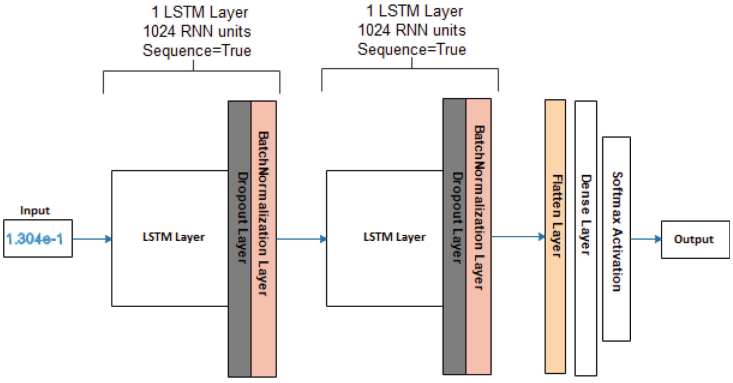

Fig. 2. The design of the proposed LSTM architecture used for generating suitable and optimal initial solutions for the proposed SIRO.

The deep learning architecture consists of two LSTM layers, each of which is followed by a dropout layer and batch-normalization layer, with each layer equivalent to

1024 RNN units. Each layer is followed by flattened and dense layers to produce outputs typical of a generator G whose sequence represents the initial solution space of SIRO. The output from the G is denoted by R, which is a sample raw solution that requires some processing or parsing to allow for it to be used as the search space. This generator and parser are represented in Eq. (7), where $\dim(P)$ denotes the size and dimension of the population as required by the SIRO algorithm:

$$R = parse(G(X, \dim(P))) \tag{7}$$

The parsed initial solution is then used to generate the individuals, which are populated into S in Eq. (8):

$$S = \left\{ s_i \in R \oint d \;\middle|\; 0 \leq i \leq \dim(P) \right\} \tag{8}$$

where $\oint$ is an N-ary operator with a clockwise integral operation on R; and d represents the delimiter, which indicates the end and beginning of the previous s_{i-1} from next s_i individual in the population. To ensure that each s_i is kept within the bounds of the problem space, Eq. (9) is applied to amend all s_i in S, where ub and lb are upper and lower bounds, respectively:

$$S = \left\{ s_i \in [amend(S, i)]_{lb}^{ub} | 0 \leq i \leq \dim(P) \right\} \tag{9}$$

During iteration for the optimization process, the global best s_{best} individual is obtained by using Eq. (10), which compares s_{cbest} with the previously obtained s_{cbest}:

$$s_{best} = \begin{cases} s_{best}, fits(s_{cbest}) < fits(s_{best}) \\ s_{cbest}, fits(s_{cbest}) \geq fits(s_{best}) \end{cases} \tag{10}$$

The position of every s_i in the search space is used to discover when the displacement of the individual has reached a disease-super-spreading point when such an individual is infected. As a result, at the beginning of every iteration, the current positions of the individuals in I are computed and used to determine when intensification and explorative mechanism of the algorithm occur. In Eq. (11), the computation of the positions for each individual in I is represented:

$$lpos_i^{t+1} = Ipos_i^t + rand(0, 1) * \rho \tag{11}$$

$$M(s) = lrate * rand(0, 1) + M(Ind_{best}) \tag{12}$$

where ρ represents the scale factor of displacement of an individual; $Ipos_i^{t+1}$ and $Ipos_i^t$ are the updated and original position at time t and t + 1, respectively; and *rand*(0, 1) randomly yields a value in the range 0 and 1.

The value obtained for $Ipos_i^{t+1}$ is assigned to *srate* when $Ipos_i^{t+1} < 0.5$; otherwise, it is assigned to *lrate*.

2.4 SIRO Neural Network-Based Parameter Selection

Compared with other related methods, SIRO uses a few parameters for the control and optimization process. To take this advantage further, this study proposes a novel deep learning approach that intelligently combines and selects the best parameter-value configuration required to yield a state-of-the-art performance. To achieve this, a pool of parameter-value configurations is generated and applied to optimize some benchmark functions so that the performance of the image of SIRO for each configuration is collected as a dataset for the deep learning model. Using a convolutional neural network (CNN), the datasets are supplied as input for training the model so that the trained model can be used to predict the best configuration appropriate for yielding the best performance of the best SIRO algorithm. In Eq. (13), the modeling of the mapping function, which generates the pool of configurations used in the experimentation, is described:

$$pv(\pi, g, \beta, G) = pv : \{p_1, p_2, p_3, p_4\} \rightarrow \{v_1, v_2, v, v_4\} \tag{13}$$

where *pv* represents a mapping function whose output is a set *C* of a unique combination of the SIRO parameters. The elements of *C* are described as $\{c \in pv(\pi, g, \beta, G)_i | 0 \leq i \leq N\}$. Each *c* is equivalent to a combination of parameters and their corresponding generated values so that such mapping of parameter-value is applied for partial training SIRO, and the convergence of the graph is generated and collected as a dataset for training the CNN. The architecture of the CNN used for this training is illustrated in Fig. 3, whereby the completely trained network is then used to predict which best c_i is suitable for fully training SIRO.

The CNN architecture consists of six blocks of convolution-pooling units, where each block contains two convolution layers, one zero-padding layer, and one max pooling layer. A 3 × 3 filter size is used for all layers of convolution in the architecture, though the filter count for each block of convolution-pooling follows 2^k where *k* ranges between [5, 10]. The architecture is completed using a flattened layer, Dense layer, and Dropout layer, and the SoftMax activation function is used for classification purposes.

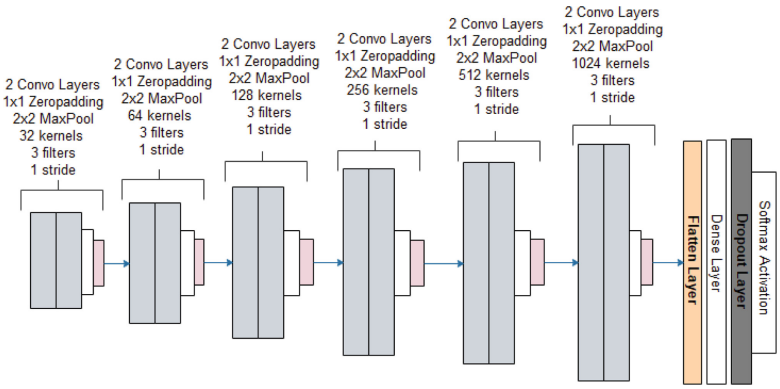

Fig. 3. The design of the convolutional neural network (CNN) architecture applied for learning solution pattern of SIRO to obtain the best configuration and combination of parameter values

2.5 SIRO Algorithms and Computational Complexity

In Algorithm 1, the pseudo-code of the proposed SIRO metaheuristic algorithm is presented. The algorithm lists all the procedures described in previous paragraphs and outlines how the optimization process is achieved during the iterative process. Input to the algorithm includes the number of iterations required for training the model, population size N, and dataset required to train the CNN model for obtaining the best c_i to be used for training SIRO. The expected output is the combination of all the best solutions obtained for each iteration and also the final global best. Line 1 of the algorithm shows the initialization of the S, I, and R compartments and also the container for storing the best solutions. In order for the LSTM model to generate the best initial solution for training SIRO, a stochastically generated initial solution X is supplied as input for training the LSTM model. Then, the trained LSTM model generates the learned-solution space for SIRO for full training. Meanwhile, Lines 5–7 display the application of the CNN model to generate the best configuration of the parameter-value set suitable for obtaining the values to use on all parameters of SIRO. The fitness values of all individuals in S are obtained and the global best solution is assigned in Lines 8–9. The iterative training of the SIRO algorithm is listed in Lines 10–31, and the return values are in Line 32. The generation and mutation of individuals into the I compartment are seen in Lines 12–21, and updates of individuals in all three compartments are exhibited in Lines 22–25. Lastly, the update of the current and global best solution for each iteration is reported in Lines 26–30.

Algorithm 1: SIRO metaheuristic algorithm

```
   Result: global Best, solutions
   Input: iter, N, dataset
   Output: sols, gbest
1  S, I, R, sols ← ∅;
2  X ← createPopulation(N, S) using Eq.6;
3  lstm = buildLSTM();
4  S = lstm.trainGenS(X, N) using Eq.9;
5  cnn = buildCNN();
6  model = cnn;
7  params = Δ ∘ {π, Γ, γ, β};
8  π, Γ, γ, β ← apply(cnn, params);
9  S = fits(S, asc);
10 gbest, cbest, I[0] ← S[0];
11 while e ≤ iter do
12 |  lpos ← position(I) using Eq.11;
13 |  for i ← 1 to len(I) do
14 |  |  newI ← ∅;
15 |  |  if lpos_i < 0.5 then
16 |  |  |  tmp ← rand(0, Eq.3 × I × srate, γ, β);
17 |  |  end
18 |  |  else
19 |  |  |  tmp ← rand(0, Eq.3 × I × lrate, γ, β);
20 |  |  end
21 |  |  newI+ ← tmp;
22 |  end
23 |  I+ ← newI;
24 |  r ← rand(0, Eq.4 × I, γ), R+ ← r;
25 |  I+ ← I − r;
26 |  S+ ← r;
27 |  cbest = fits(S);
28 |  if cbest > gbest then
29 |  |  gbest = cbest;
30 |  |  sols ← gbest;
31 |  end
32 end
33 return gbest, sols;
```

The complexity of the proposed SIRO algorithm can be computed by partitioning the algorithm into three segments, namely the initialization, optimization, and result return stages. The first stage, i.e. the initialization stage, has Lines 5, 7, and 8, which will yield a computational analysis of $O(n)$ and all other lines showing evidence of basic operations evaluated to $O(1)$. Computationally speaking, it is clear that $O(n)+ O(n)+ O(n) = O(n)$ by the rule of summation algorithm analysis, where n is the population size of the search space. The nested loop seen in Lines 11–31 has a computational analysis equivalent to $O(m * n)$ for the optimization stage, where m in this case is the number of iterations during the optimization process. The third stage which returns the result of the optimization process can be equated to $O(1)$. Therefore, the complexity of the entire algorithm may be evaluated by summing $O(n)+ O(m * n)+ O(1)$, which in turn yields $O(m * n)$. Note that for simplification, may be defined as $O(n^2)$.

In the next section, detailed experimentation of the proposed SIRO algorithm is presented along with a thorough discussion of the results. For the implementation of SIRO,

we combined the solution-space initialization, parameter fine-tuning, and optimization process of the SIR model, as described in Algorithm 1.

3 System and Parameter Configuration

The experimentation was conducted using the Google Colab platform, which has 12 GB memory, 100 GB disk size, Python 3 backend, and a GPU. Additional experiments were performed on the Google Cloud compute engine with specific configurations. The parameter values for the SIRO algorithm were derived from ranges and step variables, with certain parameters computed from randomly generated models. Each parameter had a grid of ten candidate values for the deep learning model to select from. The details of the parameter configurations can be found in Table 2. The parameters and candidate values were used to train the deep learning model for SIRO. In the next subsection, we discuss the application of IEEE CEC 2017 functions to rigorously test the resulting SIRO configuration. Moreover, various metaheuristic algorithms were considered, spanning human-based, evolutionary-based, swarm-based, biology-based, physics-based, and math-based approaches. This includes the Arithmetic Optimization Algorithm (AOA), Artificial Bee Colony (ABC), Aquila Optimizer (AO), Archimedes Optimization Algorithm (ArchOA), Ebola Optimization Search Algorithm (EOSA), Firefly Algorithm (FA), Invasive Weed Optimization (IWO), and Memetic Algorithm (MA).

Table 2. List of SIRO parameters with corresponding notations and range of values

Variable	recruitment_rate	disease_induced_death_rate	contact_rate_infectious	contact_rate_recovered	recovery_rate	natural_death_rate
Notation	π	G	β_1	β_2	g	t
Parameter range	[0.1, 0.9]	*randf* (0, 1)	[0.1, 0.9]	[0.1, 0.9]	*randf* (0, 1)	*randf* (0, 1)
Step	0.1	N/A	0.1	0.1	N/A	N/A
Size	10	10	10	10	10	10

Training SIRO for each parameter-value configuration was achieved using 50 iterations since the aim was to perform partial training for plotting the convergence curve. Therefore, the scatter plots representing the convergence of solutions in the solution space were obtained as an image input for training the CNN model. Figure 4 displays sample scatter plots for the parameter-value configurations listed in Table 3. It can be seen that each plot has a unique representation of the distribution of points in the scatter plot to show how the solutions converge after some training time.

Scatter plot-generated image samples were employed to train the CNN to learn input convergence patterns. The goal was to predict optimal image representations for fully training the SIRO method. After the completion of CNN training, a correctly predicted image with superior convergence was selected, and its corresponding parameter values were extracted. The optimal parameter-value combination for training SIRO was identified as π: 0.3, β1: 0.0783, β2: 0.3, Γ: 0.3, γ: 0.1914, and τ: 0.1272.

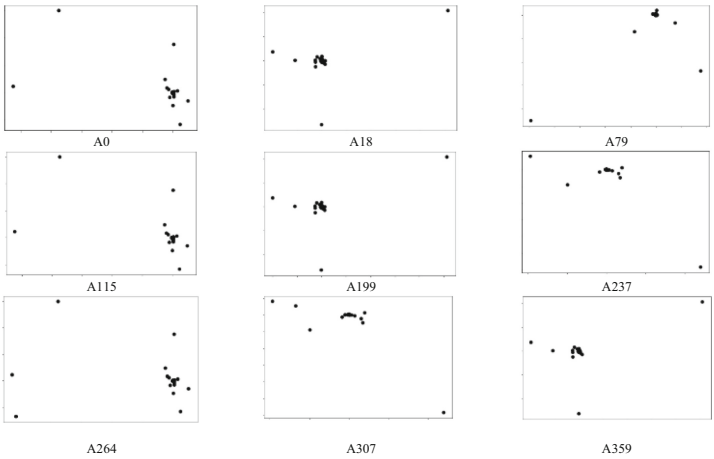

Fig. 4. Scatter plot illustrating parameter-value configurations used as input for training a CNN in an image-label solution.

3.1 Results and Discussion

Benchmark functions are traditionally used for evaluating metaheuristic algorithms to compare their performances. The standard benchmark functions and IEEE CEC functions have been widely accepted and applied for testing the performance of optimization algorithms. Since the IEEE CEC functions demonstrate some uniqueness and complexity, which are comparable with real-life optimization problems, we focused on the evaluation of the SIRO algorithm using these functions. Ideally, the CEC functions consist of fourteen (14) functions, offering a comprehensive range of computational capabilities. We experimented with 28 hybrid combinations derived from these 14 fundamental CEC functions. These hybrids include variants such as C1–C8 and C10, which undergo shifts (S) relative to their original CEC counterparts. The C9 and C11–C16 represent the shift-rotate (SR) adaptation of the corresponding CEC functions. The functions C17–C22 represent shift variants of a combination of some CEC functions, whereas C23–C28 are hybrids of predefined CEC hybrids. Each of the 28 derived hybrids has 30 dimensions.

Results presented in this section are based on the pre-training of SIRO using deep learning methods and the training and testing of SIRO using the hybrid CEC functions. The section is divided into three subsections so that three major issues are addressed: 1) evaluation of the SIRO parameter selection capability as obtained using the CNN; 2) comparative analysis of SIRO with other similar optimization methods to show if the deep learning aided method is preferable; and 3) a collection of observations into a discussion of findings as revealed from the study. The selected parameter-value configuration was used to fully train SIRO under 500 iterations. In addition to SIRO, other metaheuristic algorithms such as ABC, AO, AOA, ArchOA, EOSA, FFA, IWO, and MA were trained for the purpose of comparative analysis.

Based on the experimental results presented in Table 3, the SIRO algorithm demonstrates exceptional performance across the various hybrid benchmark functions, particularly surpassing in handling complex real-life problems involving shift and rotate

operations. Its competitiveness is evident in outperforming other compared methods across different hybrid function types, indicating its suitability for addressing a wide range of optimization challenges. Moreover, SIRO maintains its effectiveness even when confronted with higher levels of complexity, as observed in functions involving combinations of CEC benchmarks. The method's consistent performance across different function categories underlines its utility and confirms its status among top optimization algorithms for tackling complex real-world problems. In summary, it was found that SIRO returned the best performance among all methods for the C1–C8 hybrid categories, while the performance is competitively tied among AO, AOA, EOSA, and ArchOA. A further evaluation of SIRO on C9 and C11–C16 functions also revealed good performance. Recall that these functions present a higher level of complexity than the C1–C8, notwithstanding, the performance of SIRO with this increasing level of complexity is highly encouraging.

Figure 5 depicts the analysis of SIRO's exploration and exploitation patterns. The ideal curves exhibit a smoothly drawn slanted U-shape, with the exploration curve descending from the top-left to bottom-right, and the exploitation curve ascending from bottom-left to top-right. SIRO displayed distinctive and nearly perfect curve patterns, showcasing its exceptional performance in navigating the solution space. In comparison to other algorithms, SIRO demonstrated superior abilities in efficiently finding solutions within both exploration and exploitation phases, affirming its capacity to intelligently avoid local optima. An effective optimization algorithm is characterized by convergence curves aligning into a straight line over iteration or training time. In contrast, if these curves scatter without evident convergence as iterations increase, the optimization method is deemed underperforming. SIRO's categorization among top-performing metaheuristic algorithms, demonstrated by its convergence graph, underscores its suitability for challenges demanding smooth and desirable convergence in the solution space.

Figure 6 illustrates the computational time needed by the optimization methods, including the SIRO algorithm, to execute the distinctive C10 benchmark function. Notably, SIRO outperformed other high-performing algorithms like EOSA and AOA, completing the task in just 0.05 secs, while EOSA and AOA required an average of 0.30 secs, respectively. This highlights SIRO's exceptional efficiency compared to similar methods. The exhaustive experimentation with the SIRO method indicates its competitiveness with state-of-the-art methods in addressing real-life optimization problems. Notably, the use of deep learning for parameter-value configuration improved SIRO's performance, ensuring consistent configuration across instances. This is a significant contribution to the field of optimization domain. SIRO's performance on benchmark CEC functions highlights its utility for real-life optimization problems, especially those akin to C1–C28 functions. In summary, SIRO competes favorably, demonstrating outstanding performance and effective balance between exploration and exploitation processes. Its convergence graphs affirm its relevance for problems demanding smooth convergence of the solution space.

3.2 Analysis of Statistical Results

The Anderson-darling test states that if the p-value > 0.05, then the dataset is normally distributed and the parametric test should be considered, otherwise a non-parametric

Table 3. A comparative analysis of SIRO against other optimization techniques using the hybrid functions of CEC 2017, each characterized by 30 dimensions.

Func.	Metric	ABC	AO	AOA	ArchOA	EOSA	FFA	IWO	MA	SIRO
C1	best	3.6E+03	**1.0E+02**	6.7E+04	**1.1E+02**	1.0E+03	7.9E+03	3.8E+05	2.2E+06	**1.1E+02**
	mean	2.9E+04	5.3E+04	6.7E+04	1.8E+04	4.2E+03	1.4E+04	3.9E+05	2.2E+06	2.1E+03
	std	1.4E+05	1.8E+05	1.5E−11	6.4E+04	2.7E+04	2.6E+04	4.5E+04	6.1E+03	1.6E+04
	worst	1.8E+06	2.1E+06	6.7E+04	6.4E+05	5.5E+05	5.7E+05	6.2E+05	2.3E+06	2.6E+05
C2	best	2.9E+05	2.7E+02	4.0E+06	**2.0E+02**	7.5E+05	5.1E+05	3.6E+08	1.3E+08	**2.0E+02**
	mean	1.8E+07	4.5E+07	4.0E+06	1.1E+07	1.7E+06	1.9E+07	3.9E+08	1.3E+08	1.0E+06
	std	1.1E+08	1.6E+08	0.0E+00	1.1E+08	5.5E+06	1.3E+08	1.1E+08	4.5E+06	5.3E+06
	worst	1.2E+09	1.6E+09	4.0E+06	1.6E+09	7.2E+07	1.4E+09	1.1E+09	1.6E+08	1.0E+08
C3	best	3.0E+02	3.0E+02	1.0E+06	3.0E+02	3.1E+02	1.9E+03	1.7E+04	1.9E+04	4.6E+02
	mean	9.6E+02	6.9E+03	1.0E+06	7.0E+02	2.0E+03	2.7E+03	1.7E+04	1.9E+04	8.0E+03
	std	4.5E+03	6.4E+03	0.0E+00	2.8E+03	6.9E+03	5.1E+02	1.5E+03	0.0E+00	1.4E+05
	worst	5.1E+04	2.6E+04	1.0E+06	2.4E+04	5.9E+04	5.2E+03	2.4E+04	1.9E+04	3.1E+06
C4	best	4.0E+02	4.0E+02	4.0E+02	4.0E+02	4.0E+02	4.0E+02	4.8E+02	4.1E+02	4.0E+02
	mean	**4.0E+02**	**4.0E+02**	**4.0E+02**	**4.0E+02**	**4.0E+02**	**4.0E+02**	4.8E+02	4.1E+02	**4.0E+02**
	std	5.4E+00	1.5E+01	5.7E−14	6.2E+00	5.6E+00	5.8E+00	5.7E−14	2.0E+00	7.5E+00
	worst	4.6E+02	5.5E+02	**4.0E+02**	4.9E+02	5.2E+02	4.7E+02	4.8E+02	4.4E+02	5.6E+02
C5	best	**5.0E+02**	**5.0E+02**	**5.0E+02**	**5.0E+02**	**5.0E+02**	**5.2E+02**	**5.2E+02**	**5.2E+02**	**5.0E+02**
	mean	5.0E+02	5.1E+02	5.0E+02	5.1E+02	5.0E+02	5.2E+02	5.2E+02	5.2E+02	5.0E+02
	std	2.9E+00	9.4E+00	1.1E−13	8.8E+00	5.2E−01	3.8E−02	5.9E−04	6.8E−01	2.5E+00
	worst	5.2E+02	5.2E+02	5.0E+02	5.2E+02	5.1E+02	5.2E+02	5.2E+02	5.2E+02	5.2E+02
C6	best	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**	**6.0E+02**
	mean	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02
	std	2.8E−01	5.1E−01	1.1E−13	2.8E−01	7.8E−02	1.8E−01	1.1E−13	3.7E−02	1.1E−01
	worst	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02	6.0E+02
C7	best	**7.0E+02**	**7.0E+02**	**7.0E+02**	**7.0E+02**	**7.0E+02**	**7.0E+02**	7.1E+02	**7.0E+02**	**7.0E+02**
	mean	7.0E+02	7.0E+02	7.0E+02	7.0E+02	7.0E+02	7.0E+02	7.1E+02	7.0E+02	7.0E+02
	std	2.7E+00	3.1E+00	1.1E−13	1.2E+00	1.6E+00	1.5E+00	5.8E−01	7.8E−02	1.4E+00
	worst	7.3E+02	7.3E+02	**7.0E+02**	7.1E+02	7.2E+02	7.2E+02	7.2E+02	**7.0E+02**	7.3E+02
C8	best	**8.0E+02**	**8.0E+02**	**8.0E+02**	**8.0E+02**	**8.0E+02**	8.1E+02	8.3E+02	8.1E+02	**8.0E+02**
	mean	8.0E+02	8.1E+02	8.0E+02	8.1E+02	8.0E+02	8.2E+02	8.3E+02	8.1E+02	8.0E+02
	std	3.2E+00	3.7E+00	0.0E+00	7.8E+00	2.7E+00	2.8E+00	8.3E−02	6.4E−01	2.1E+00
	worst	8.3E+02	8.2E+02	**8.0E+02**	8.4E+02	8.2E+02	8.4E+02	8.3E+02	8.1E+02	8.4E+02
C9	best	**9.1E+02**	**9.0E+02**	**9.0E+02**	**9.0E+02**	**9.0E+02**	9.1E+02	9.4E+02	9.1E+02	**9.0E+02**
	mean	9.1E+02	9.0E+02	9.0E+02	9.0E+02	9.0E+02	9.1E+02	9.4E+02	9.1E+02	9.0E+02
	std	1.8E+00	3.9E+00	1.1E−13	6.4E+00	2.2E+00	1.8E+00	1.3E−01	9.4E−01	2.5E+00
	worst	9.3E+02	9.2E+02	**9.0E+02**	9.3E+02	9.5E+02	9.3E+02	9.4E+02	9.2E+02	9.4E+02
C10	best	1.3E+03	**1.0E+03**	1.1E+03	**1.0E+03**	1.1E+03	1.3E+03	2.0E+03	1.6E+03	**1.0E+03**
	mean	1.3E+03	1.2E+03	1.1E+03	1.1E+03	1.1E+03	1.3E+03	2.0E+03	1.6E+03	1.0E+03

(*continued*)

Table 3. (*continued*)

Func.	Metric	ABC	AO	AOA	ArchOA	EOSA	FFA	IWO	MA	SIRO
	std	6.8E+01	1.6E+02	4.5E−13	1.4E+02	4.2E+01	7.6E+01	4.5E−13	3.5E+01	9.6E+01
	worst	2.1E+03	1.7E+03	1.1E+03	1.7E+03	2.1E+03	1.7E+03	2.0E+03	1.8E+03	1.9E+03
C11	best	1.2E+03	**1.1E+03**	1.2E+03	**1.1E+03**	**1.1E+03**	1.2E+03	2.0E+03	1.8E+03	**1.1E+03**
	mean	1.3E+03	1.3E+03	1.2E+03	1.3E+03	1.1E+03	1.3E+03	2.0E+03	1.8E+03	1.1E+03
	std	4.4E+01	1.7E+02	2.3E−13	2.5E+02	3.8E+01	7.2E+01	4.6E+00	1.7E+01	6.8E+01
	worst	1.8E+03	2.0E+03	1.2E+03	2.1E+03	1.9E+03	1.6E+03	2.1E+03	1.8E+03	1.9E+03
C12	best	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**	**1.2E+03**
	mean	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03
	std	3.2E−01	8.1E−01	2.3E−13	4.6E−01	6.8E−01	1.4E−01	5.2E−01	0.0E+00	7.2E−01
	worst	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03	1.2E+03
C13	best	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**	**1.3E+03**
	mean	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03
	std	1.5E−01	2.6E−01	4.5E−13	1.5E−01	3.6E−01	1.9E−01	2.3E−13	5.2E−02	3.8E−01
	worst	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03	1.3E+03
C14	best	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**	**1.4E+03**
	mean	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03
	std	6.8E−02	6.3E−01	2.3E−13	3.9E−01	6.3E−01	6.7E−01	2.3E−13	1.8E−02	1.3E−01
	worst	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03	1.4E+03
C15	best	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**	**1.5E+03**
	mean	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03
	std	2.9E−01	6.1E+00	0.0E+00	6.4E−01	1.7E+00	3.4E−01	4.2E−02	2.6E−01	5.3E−01
	worst	1.5E+03	1.6E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03	1.5E+03
C16	best	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**	**1.6E+03**
	mean	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03
	std	7.4E−02	9.9E−02	0.0E+00	1.5E−01	6.3E−02	4.1E−02	2.3E−13	9.3E−02	1.6E−02
	worst	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03	1.6E+03

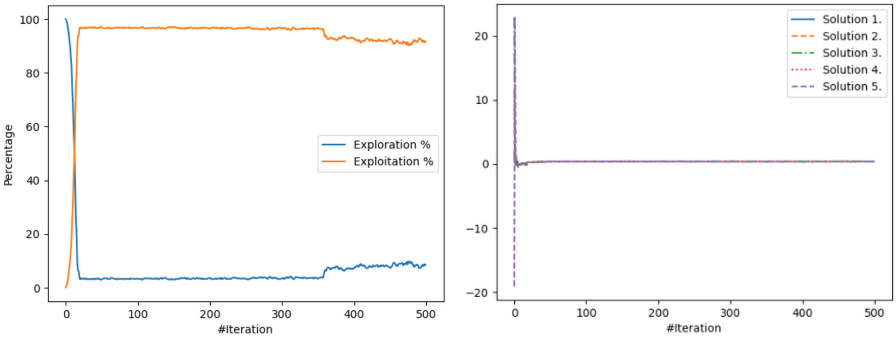

Fig. 5. The left plot illustrates the exploration and exploitation of SIRO, while the right plot shows the convergence trajectory of the first five (5) solutions in the solution space of SIRO.

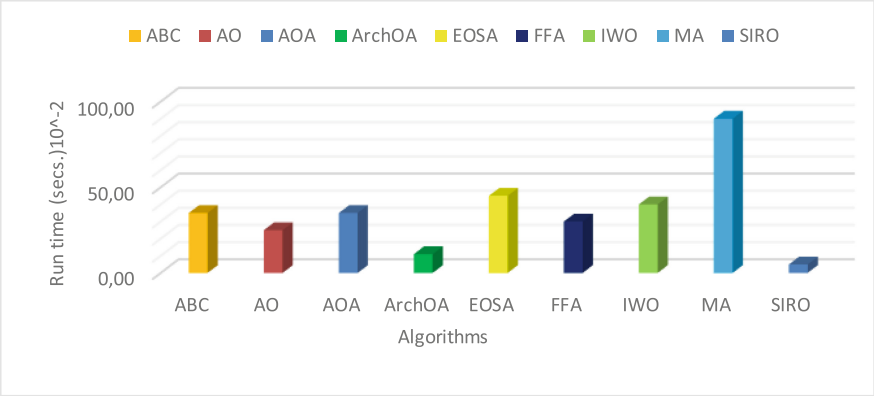

Fig. 6. Computational time comparison of SIRO and other optimization methods on benchmark function C10.

test should be employed. However, the initial test result obtained, with a p-value of 3.53e−25 shows that the data do not follow normal distribution. Therefore, Friedman's test which is a non-parametric statistical test was selected for the comparative study of SIRO with other algorithms. The non-parametric Friedman's test was conducted to obtain the rank SIRO in comparison with the other algorithms. According to Friedman's test, the algorithms are not equally effective. The proposed algorithm SIRO ranked 1 as the best algorithm, while other well-performing algorithms such as EOSA, ArchOA, and AOA are ranked, 2, 3, and 4 respectively. In Table 4, the ranks of the compared algorithms based on Friedman's rank test with a 95% level of confidence are illustrated.

Table 4. Friedman's rank test of proposed SIRO with considered algorithms Null hypothesis (H_0): All the algorithms are equally effective.

Method	Mean rank	Rank	p-value	Conclusion
ABC	4.84	5	1.318e−17	H_0 may be rejected for $\alpha = 1\%$ since p-value = 1.318e−17 < 0.01. At 1% level of significance, effectiveness of several algorithms are not equal considering type – I error of 1%
AO	5.02	6		
AOA	4.48	4		
ArchOA	4.23	3		
EOSA	3.50	2		
FFA	5.13	7		
IWO	7.45	9		
MA	7.13	8		
SIRO	3.23	1		

4 Conclusion and Future Work

This paper introduces a machine-learning approach to optimize parameter configurations for the SIRO algorithm. It utilizes an LSTM model to broaden the search space and explores different SIRO-parameter schemes through a parameter grid. SIRO is trained using permutations of these values, generating image representations for a CNN to identify optimal parameter values. Fully trained SIRO excels in IEEE CEC 2017 benchmark test functions, outperforming similar methods. Analysis of exploration-exploitation and convergence demonstrates SIRO's competitiveness. The approach enables consistent parameter values, challenging stochastic methods' dependency. Future research could extend benchmarks and assess SIRO's robustness against challenging functions and real-world problems. Moreover, SIRO demonstrates versatility in addressing challenging computer vision problems, including medical image classification, along with complex machine learning tasks like deep-leaning parameter optimization and feature selection.

References

1. Seyyedabbasi, A.: A reinforcement learning-based metaheuristic algorithm for solving global optimization problems. Adv. Eng. Softw. **178**, 103411 (2023)
2. Talbi, E.G.: Machine learning into metaheuristics: a survey and taxonomy. ACM Comput. Surv. (CSUR) **54**(6), 1–32 (2021)
3. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Comput. Ind. Eng. **137**, 106040 (2019)
4. Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. Artif. Intell. Rev. **52**(4), 2191–2233 (2019)
5. Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., Zhao, W.: Artificial rabbits optimization: a new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. **114**, 105082 (2022)
6. Zhao, S., Zhang, T., Ma, S., Chen, M.: Dandelion Optimizer: a nature-inspired metaheuristic algorithm for engineering applications. Eng. Appl. Artif. Intell. **114**, 105075 (2022)
7. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)