biomimetics

Article

Training of Feed-Forward Neural Networks by Using Optimization Algorithms Based on Swarm-Intelligent for Maximum Power Point Tracking

Ebubekir Kaya [1,*], Ceren Başternur Kaya [2], Emre Bendeş [1], Sema Atasever [1], Başak Öztürk [1] and Bilgin Yazlık [1]

[1] Department of Computer Engineering, Engineering Architecture Faculty, Nevsehir Haci Bektas Veli University, Nevşehir 50300, Türkiye; emrebendes@nevsehir.edu.tr (E.B.); sema@nevsehir.edu.tr (S.A.); basakozturk@nevsehir.edu.tr (B.Ö.); bilginyazlik@nevsehir.edu.tr (B.Y.)
[2] Department of Computer Technologies, Nevsehir Vocational School, Nevsehir Haci Bektas Veli University, Nevşehir 50300, Türkiye; ceren@nevsehir.edu.tr
* Correspondence: ebubekir@nevsehir.edu.tr

Abstract: One of the most used artificial intelligence techniques for maximum power point tracking is artificial neural networks. In order to achieve successful results in maximum power point tracking, the training process of artificial neural networks is important. Metaheuristic algorithms are used extensively in the literature for neural network training. An important group of metaheuristic algorithms is swarm-intelligent-based optimization algorithms. In this study, feed-forward neural network training is carried out for maximum power point tracking by using 13 swarm-intelligent-based optimization algorithms. These algorithms are artificial bee colony, butterfly optimization, cuckoo search, chicken swarm optimization, dragonfly algorithm, firefly algorithm, grasshopper optimization algorithm, krill herd algorithm, particle swarm optimization, salp swarm algorithm, selfish herd optimizer, tunicate swarm algorithm, and tuna swarm optimization. Mean squared error is used as the error metric, and the performances of the algorithms in different network structures are evaluated. Considering the results, a success ranking score is obtained for each algorithm. The three most successful algorithms in both training and testing processes are the firefly algorithm, selfish herd optimizer, and grasshopper optimization algorithm, respectively. The training error values obtained with these algorithms are 4.5×10^{-4}, 1.6×10^{-3}, and 2.3×10^{-3}, respectively. The test error values are 4.6×10^{-4}, 1.6×10^{-3}, and 2.4×10^{-3}, respectively. With these algorithms, effective results have been achieved in a low number of evaluations. In addition to these three algorithms, other algorithms have also achieved mostly acceptable results. This shows that the related algorithms are generally successful ANFIS training algorithms for maximum power point tracking.

Keywords: swarm intelligence; feed-forward neural network; maximum power point tracking; metaheuristic algorithm

Citation: Kaya, E.; Başternur Kaya, C.; Bendeş, E.; Atasever, S.; Öztürk, B.; Yazlık, B. Training of Feed-Forward Neural Networks by Using Optimization Algorithms Based on Swarm-Intelligent for Maximum Power Point Tracking. *Biomimetics* **2023**, *8*, 402. https://doi.org/10.3390/biomimetics8050402

Academic Editors: Gang Hu, Weiguo Zhao and Zhenxing Zhang

Received: 30 June 2023
Revised: 26 August 2023
Accepted: 29 August 2023
Published: 1 September 2023

1. Introduction

Today, investments are being made in various energy production technologies to meet the increasing demand for energy. Among these technologies, the interest and demand for renewable energy sources, which are environmentally friendly and carbon-free, is increasing day by day. Therefore, renewable energy generation is of great importance for power generation. PV panels, one of the electric power generation systems that have these characteristics, can absorb solar energy and convert it into electrical energy. PV systems have the advantage of being easy to install and using a free energy source. However, the efficiency of PV systems depends on some external factors, such as solar radiation and ambient temperature.

MPPT methods are used to optimize nonlinear power generation due to the mentioned characteristics of photovoltaic. Thus, the objective is to generate the maximum amount of electrical energy using MPPT, using a model to control a DCDC converter. An ANN is one of the most commonly used models for MPPT [1,2]. On the other hand, training the ANN model for MPPT using traditional backpropagation is not efficient [3]. To solve this problem, metaheuristic algorithms could be used.

Metaheuristic algorithms are widely used due to their global search properties that provide efficient solutions to complex problems. With their population-based search properties, these algorithms can perform a detailed search in the search space that expresses the solution to the problem without getting stuck in local minima. Bio-inspired algorithms, which mimic the behavior of intelligent swarm intelligence in nature, construct a unique interaction model between search agents. Thus, they not only perform local searches but also contribute to global searches by communicating with other agents. Therefore, they are often preferred in solving difficult optimization problems.

In this study, the MPPT of PV systems is modeled using ANN. Determining the optimal parameters of this model is a challenging optimization problem for which bioinspired swarm intelligence-based metaheuristic optimization algorithms could be used. Many variants of swarm intelligent algorithms have been proposed, especially in the last 30 years. PSO [4], ABC [5], FA [6], KHA [7], CSA [8], DA [9], GOA [10], SHO [11], BOA [12], TSA [13], TSO [14], CS [15] and SSA [16] are some of them. A total of 13 of the well-known new and old optimization algorithms from the literature were selected for this study. A comparative analysis of the algorithms' performance in optimizing the given model was performed. The novelty of the study is that it is one of the most comprehensive studies using swarm intelligence-based algorithms for MPPT. This study makes important contributions to the literature. These contributions are presented below in substance:

- Metaheuristic algorithms are grouped according to how they occur. One of these groups is swarm intelligence-based algorithms. In this study, 13 swarm intelligence-based algorithms for FFNN training are compared. It is one of the first studies in the literature in this context.
- Metaheuristic algorithms are used to solve the MPPT problem. It is one of the most influential studies in the literature using thirteen metaheuristic algorithms for MPPT.
- The success of these algorithms in both FFNN training and MPPT will shed light on future studies.
- In this study, the effect of network structure and population size on performance is examined in detail.

The next sections of this study continue as follows: detailed related works on ANN and MPPT are given in Section 2. In Section 3, optimization algorithms based on swarm-intelligent used in this study and feed-forward neural network are introduced. Section 4 gives simulation results. The discussion is presented in Section 5. In the last section, the conclusion is given.

2. Related Work

Thus far, numerous methods have been suggested and put into practice for effectively supervising the MPP for PV systems. Yang et al. [17] performed four case studies, introducing the memetic SSA (MSSA), an advanced version of SSA, as a new optimization method for MPPT. It showed improved performance over other algorithms by generating more energy and reducing power fluctuations in changing weather conditions.

A Model Reference Adaptive Control (MRAC) approach was studied and compared with several established techniques, including INC, P&O, ANFIS, and Variable Step Perturb and Observe. According to the results, the MRAC approach outperformed other methods and achieved MPP in just 4 milliseconds. Its tracking ability and effectiveness were also found to be superior [18].

Kamarposhti et al. [19] employed the whale algorithm to optimize solar system parameters for enhanced MPP tracking accuracy and increased electrical power output.

Their study introduced an adaptive fuzzy controller to achieve this goal. The researchers performed multiple assessments under varying irradiation conditions, comparing the performance of their proposed controller against the practical and high-performance P&O algorithm. Upon analyzing simulation outcomes, they observed that their approach outperformed the P&O algorithm.

Akram et al. [20] explored the utilization of FA for MPPT in solar PV systems. The findings demonstrated that the FA technique outperformed other methodologies, such as the P&O approach, PID control, and PSO, in relation to monitoring efficiency and convergence velocity. In contrast to PSO, which required the tuning of three parameters, the FA method only required the tuning of two parameters.

Vadivel et al. [21] focused their attention on executing the SGO algorithm. To evaluate the effectiveness of the algorithm, the researchers made a comparison with other global search MPPT techniques, including PSO, DA, and ABC. The outcomes of the simulation demonstrated the superiority of SGO-MPPT, as evidenced by the ability to achieve rapid global power tracking in under 0.2 s, coupled with a decrease in oscillations.

Conventional MPPT methods work well under stable conditions, but their effectiveness decreases during rapid changes in irradiation; this requires a fast and accurate MPPT method [22]. A few studies have compared the proposed method with conventional MPPT techniques. Examples of these studies are given below.

Mirza et al. [23] conducted a study that introduced a novel MPPT technique utilizing SSA. The SSA technique utilized the confined exploitation property of salps and improved robustness and efficiency. Also, it reduced the oscillation and saved the computation time. Comparative testing against conventional MPPT techniques showed that the SSA technique successfully handled different weather conditions and achieved faster tracking and more stable output.

Jamshidi et al. [24] presented a new technique for MPPT in solar panels utilizing a Backstepping Sliding Mode Controller (BSMC) strategy. To guarantee the stability of the proposed approach, they employed Lyapunov criteria and a fuzzy inference system. Additionally, the parameters of the Fuzzy BSMC were optimized by means of a PSO technique. The findings of the study showed that the recommended controller surpassed conventional methodologies, indicating improved performance.

Pal et al. [25] conducted a comparative analysis of their study, which put forth a novel algorithm, MPPT, in PV systems with other contemporary findings. Their outcomes exhibited a reduction in iteration and tracking time, thereby augmenting the average tracking efficiency of the proposed approach.

Mirza et al. [26] applied two new techniques for MPPT of PV systems and compared the results of their research with the results of PSO, P&O, CS, and SSA. The results obtained from their study in which they examined six distinct cases while keeping the steady-state oscillation below a certain value, and they obtained the least tracking time.

Yan et al. [27] proposed an algorithm called the Adaptive PSO Back Propagation Neural Network-Fuzzy Control (APSO-BP-FLC). The proposed algorithm consists of three stages. The simulation results showed that the proposed algorithm surpasses the performance of the APSO-BP, FLC, and P&O algorithms in terms of tracking accuracy, steady-state oscillation rate, and efficiency. Specifically, the proposed algorithm yielded an improvement compared to other algorithms.

Rezk et al. [28] introduced two optimization techniques, namely PSO and CS, in their paper. The performance of both techniques was compared to that of a conventional INR-based tracker. The findings of the study revealed that PSO and CS-based trackers showed superior performance. Moreover, the CS-based tracker outperformed the PSO-based tracker in terms of tracking time and the ability to converge to the global MPP in all investigated cases.

In their study on solar PV MPPT controller optimization using gray wolf optimizer, Aguila-Leon et al. [29] compared their proposed method with other techniques such as

P&O and INC. Their results showed that the proposed controller works well under varying weather conditions and has low oscillations.

Castaño et al. [30] compared their studies, which suggested using the ABC algorithm for MPPT, with the traditional P&O method and revealed that the proposed method outperforms the proposed method. The researchers stated that the proposed algorithm has several advantages, such as high efficiency, no need for parameter knowledge, increased flexibility and simplicity, and fast-tracking power.

Al-Majidi et al. [3] proposed an optimized FFNN technique based on real data for predicting the MPP of PV arrays. The ANN model's topology and initial weights were optimized using the PSO algorithm. The proposed method exhibited hourly average efficiencies of over 99.67% and 99.30% on sunny and cloudy days, respectively. The proposed method surpassed the conventional ANN, FLC, and P&O methods.

Corrêa et al. [31] proposed a new MPPT method for PV systems, which combines P&O with a modified version of the Fractional Open Circuit Voltage algorithm. This method enabled direct duty cycle control, reduced efficiency losses due to steady-state oscillations, and tracked the global MPP during PS. Validation via computer simulations and practical experiments showed that the proposed method outperformed commonly used MPPT algorithms.

Dagal et al. [32] investigated an enhanced version of the SSA that is based on the PSO approach. The outcomes of the study demonstrated impressive results, with the proposed algorithm achieving remarkable efficiencies of up to 99.99% and generating high power outputs of up to 316.32 W and 428.6 W under optimal operating conditions.

Ahmed et al. [33] introduced an enhanced MPPT technique that integrated the principles of the P&O method and the Fractional-Order Sliding-Mode Predictive Control (FS-MPC). This technique eliminated all the drift loops associated with traditional methods. Compared to the direct and FS-MPC methods, the proposed technique was more efficient, required fewer sensors, and had a lower computational time.

Ibrahim et al. [34] introduced a hybrid MPPT algorithm that utilized PSO for optimizing the output power of PV systems. The effectiveness of this algorithm was compared to conventional methods under various weather conditions using a grid-connected PV system. The results showed that the hybrid MPPT algorithm outperformed conventional methods with a fast-tracking time of 43.4 ms and a high efficiency of 99.07%.

Ibneloud et al. [35] proposed a hybrid approach called ANN-PSO that utilized ANN for the prediction of solar irradiation and cell temperature, along with PSO for power generation optimization and solar power tracking. Simulation results demonstrated that the ANN-PSO approach attained up to 97% efficiency, indicating its potential to extract optimal power and enhance the performance of PV systems.

Kumar et al. [36] proposed a new hybrid algorithm that PSO-trained ML and flying squirrel search optimization to achieve optimum efficiency. The proposed algorithm is compared with other well-known methods. The findings of the study outperformed well-known algorithms, improving efficiency and reducing settling time.

Mohebbi et al. [37] introduced a novel method that integrated PSO with variable coefficients and P&O techniques. The proposed method was used to converge to the global MPP, while the P&O method was used until significant changes occurred in the system. In the study, in which the MPPT structure of the PV system is introduced under partial shading conditions, it is stated that the proposed algorithm has a much better performance than other methods, such as PSO and P&O.

Ngo et al. [38] proposed a hybrid method that combines the improved CSO and the INC algorithm for a DC standalone PV energy conversion system. The proposed method achieved the global MPP under uniform solar irradiance and PS effects, and it was faster and simpler in calculation than other methods. Simulation and experimental results demonstrated that this control strategy had been successful in searching the global region.

Nancy Mary et al. [39] presented a hybrid genetic-particle swarm-based (GA-PSO) MPPT and optimized ZETA converter to maximize output power and reduce ripple current.

The GA-PSO-based ZETA converter provides regulated load power from renewable energy sources and reduces distortions in the output effectively. The total harmonic distortion (THD) was also effectively lowered by 20%.

Al-Muthanna et al. [40] discussed the challenges of using PSO for MPPT in PV energy systems, especially under PS circumstances. The paper proposes a hybrid PSO-PID algorithm that combines PSO with a PID controller to enhance tracking efficiency, reduce power ripples, and improve response time. The algorithm outperforms conventional PSO and bat algorithm MPPT strategies. The proposed hybrid PSO-PID MPPT shows significant improvement.

Kacimi et al. [22] implemented a new combined GOA-Model Predictive Controller (MPC) based GMPPT technique. Experimental simulations showed that this method outperformed traditional methods (PSO, PSO-MPC, and GOA). The integration of the MPC controller with the GOA technique reduced the steady-state oscillations and overall MPPT time.

Nisha and Nisha [41] proposed a new methodology for optimal tuning of PV systems under PS conditions by integrating hybrid MPPT algorithms. The proposed system addresses issues caused by clouds, trees, dirt, and dust and computes MPPT using an adaptive model-based approach. The hybrid optimization approach combined the CS-P&O and INC-PSO algorithms to achieve MPPT with 99.5% efficiency.

Gong et al. [42] proposed a bionic two-stage MPPT control strategy to improve the accuracy and rapidity of the MPPT controller for PV systems. The first stage used an improved ABC algorithm to quickly identify the rough search region around the global peak, while the second stage used the simultaneous heat transfer search algorithm to accurately acquire the global MPP. The proposed strategy showed excellent performance and outperformed all counterparts.

Babes et al. [43] developed an FFNN model, which is optimized with the ACO learning algorithm to evaluate the MPP of a PV system. The voltage and current of the PV array were set as the input layer of the model, and the duty cycle was set as the parameter of the output layer. Six topologies were created to find the best structure, and the best model was observed as a model with a single hidden layer and 20 neurons.

Avila et al. [44] proposed a deep reinforcement learning (DRL) algorithm to maximize the efficiency of MPPT control. With this proposed neural network, sensory information was taken as input, and the control signal was taken as output.

Saravanan et al. [45] proposed an RBFN-based MPPT algorithm, and the results were compared with the conventional P&O and INC methods using MATLAB/Simulink software. The results revealed that the RBFN algorithm is better than the conventional methods.

3. Materials and Methods

3.1. Optimization Algorithms Based on Swarm-Intelligent

3.1.1. Particle Swarm Algorithm

PSO is a metaheuristic swarm intelligence-based optimization algorithm that simulates the social behavior of swarms, such as flocks of birds or schools of fish [4]. A particle in a swarm searches for food using previous experiences and discoveries of all particles in the swarm. The particles are initially randomly distributed in the search space and move according to two values related to the fitness value. The first value is the local best solution that the particle has found so far. The second value is the global best solution found by the swarm. First, each particle orders its acceleration by distance as follows:

$$v_i = v_i + 2r_1\left(x^t_{iBest} - x^t_i\right) + 2r_2\left(x^t_{Best} - x^t_i\right) \quad (1)$$

Here, r_1 and r_2 are random values in [0, 1]. x^t_{iBest} is the best solution of the i-th particle and x^t_{Best} is the best solution for the swarm so far. After that, each particle moves to the next position by using the following equation.

$$x^{t+1}_i = x^t_i + v_i \quad (2)$$

3.1.2. Artificial Bee Colony Algorithm

ABC algorithm is a metaheuristic algorithm inspired by the swarming behavior of bees to find a food source [5]. Three types of artificial bees perform search operations in ABC. The first type of bee is the employed bee, which searches the food area independently. Each employed bee has a food source. Another type is onlooker bees, which select a food source from employed bees. In ABC, the colony consists of the same number of employed bees and onlooker bees, and a food source improved by only one employed bee and a random number of onlooker bees. In a predetermined iteration, an employed bee that cannot find a better food source transforms into the last type of bee, the scout bee, which performs random exploration. At the initialization of ABC, the employed bees randomly take their positions. After that, each type of bee performs exploration successively throughout the iterations. First, the employed bees search for a better position at the neighboring food source as a candidate solution as follows:

$$v_{i,j}^t = x_{i,j}^t + \phi_{i,j}\left(x_{i,j}^t - x_{k,j}^t\right) \tag{3}$$

where $j\epsilon\{1, 2, \cdots D\}$ for D dimensional search space, and k is randomly selected bee index. t is the iteration index. $\phi_{i,j}$ is a random number in $[-1, 1]$. If a new location is better, the bee moves there. In the second step, onlooker bees choose a location with a probability calculated depending on the fitness values of the employed bees and behave as an employed bee. In the last step, an employed bee whose position has not changed for a certain number of iterations moves randomly in the search space. This step is the scout bee step. These three steps continue until the stop criterion is met.

3.1.3. Firefly Algorithm

FA is a metaheuristic algorithm that simulates the flashing behavior of fireflies to find an optimal solution [6]. The objective function determines the brightness of a firefly with which it attracts other fireflies. The attraction also depends on the distance between the two fireflies. The greater the distance, the less the attraction. A firefly updates its position relative to the brightest firefly in proportion to the distance. If the brighter firefly is not found, it goes to a random position. The updating of the position of a firefly is formulated as follows:

$$x_i^{t+1} = x_i^t + e^{-\lambda r^2}\left(x_b^t - x_i^t\right) + \alpha\varepsilon \tag{4}$$

where x_i^t is the current location of the firefly and x_b^t is the location of the brightest firefly. λ is the light absorption coefficient, and r is the distance. α is a random value in [0, 1], and ε is a random vector. The algorithm starts with randomly generated fireflies. The brightness is calculated as a function of distance and fitness value as follows:

$$I(r) = I_0 e^{-\lambda r^2} \tag{5}$$

where I_0 is the fitness value of the other fireflies. A firefly selects the best $I(r)$ value as brightest and then flies toward it. If there is no brighter value, it flies randomly. These steps continue until the algorithm stops.

3.1.4. Krill Herd Algorithm

KHA is a metaheuristic algorithm that simulates the density-dependent attraction and foraging of krill populations [7]. Three operations determine the movement of a krill in search space. The first is the movement influenced by other krill individuals and is calculated as follows:

$$N_i^{t+1} = N^{max}\alpha_i + \omega_i N_i^t \tag{6}$$

Here, $\alpha_i = \alpha_i^{local} + \alpha_i^{target}$. α_i^{local} is the local effect caused by the neighbor and α_i^{target} is the direction determined by the best krill individual. N^{max} is the maximum induced speed, ω_i is the inertia weight of induced motion in [0, 1], and N_i^t is the last induced motion. The

second operation is the foraging motion, which is calculated based on the food location and previous experience. The foraging motion is expressed as follows:

$$F_i^{t+1} = V_f \beta_i + \omega_f F_i^t \tag{7}$$

where $\beta_i = \beta_i^{food} + \beta_i^{best}$ which β_i^{food} is the attractiveness of the food and β_i^{best} is the effect of best fitness. V_f is the speed of foraging and ω_f is the inertia weight for foraging motion in [0, 1]. The krill individuals move randomly in the search space at the last operation, which is called physical diffusion, as follows:

$$D_i^{t+1} = D^{max} \delta \tag{8}$$

Here, D^{max} is the maximum diffusion speed, and δ is a random vector. Using all three motion operations above, the final motion of the krill is calculated as follows:

$$\begin{aligned} x_i^{t+\Delta t} &= x_i^t + \Delta t \left(N_i^{t+1} + F_i^{t+1} + D_i^{t+1} \right) \\ \Delta t &= C_t \sum_{j=1}^{n} \left(U_j - L_j \right) \end{aligned} \tag{9}$$

where n is the dimension of the search space, U_j is the upper bound and L_j is the lower bound of the jth dimension. C_t is a constant in [0, 2]. In the last step, for the sake of improving the performance of the algorithm, genetic operators such as the crossover and the mutation are implemented. All these steps form a KHA iteration that continues until stopping conditions are met.

3.1.5. Chicken Swarm Optimization

CSO is a bio-inspired metaheuristic optimization algorithm that mimics the behaviors of a chicken swarm [8]. The swarm is divided into groups consisting of one roaster, a number of hens, and chicks. Chickens are ranked according to their fitness values. The best *RN* individuals are assigned as roosters, the worst *CN* individual chicks, and the rest as hens. The roster whose fitness value is better has a bigger probability of searching a wider range of areas. The new position of a roster is calculated as follows:

$$x_i^{t+1} = x_i^t \left(1 + rand\left(0, \sigma^2 \right) \right) \tag{10}$$

Here, $rand\left(0, \sigma^2\right)$ is a Gaussian distribution in which the mean is 0 and standard deviation is σ^2. A, hence, is moved by following the roster in the same group, and this is formulated as follows:

$$x_i^{t+1} = x_i^t + re^{\left(\frac{f_i - f_{r1}}{|f_i| + \varepsilon}\right)} \left(x_{r1}^t - x_i^t \right) + re^{(f_{r2} - f_i)} \left(x_{r2}^t - x_i^t \right) \tag{11}$$

where r is a random value in [0, 1]. x_{r1}^t is the position of the roster while x_{r2}^t is a position of any chicken in the group. The f is the fitness value of a chicken, and ε is smallest value a computer can produce. The chicks search around their mother with the following formula.

$$x_i^{t+1} = x_i^t + F\left(x_m^t - x_i^t \right) \tag{12}$$

where F is a constant in [0, 2] and x_m^t is the position of the chick's mother. After G iterations, the status of chickens is rearranged, and each chicken moves depending on its status at each iteration until CA stops because of stopping conditions.

3.1.6. The Dragonfly Algorithm

DA is one of the swarm intelligent optimization algorithms whose main inspiration is the swarm behavior of dragonflies during hunting and migration [9]. Individuals in

the swarm change their position based on five factors: separation, alignment, cohesion, attraction, and distraction. Separation is expressed by the sum of the distances of an individual location from other individual locations. Alignment is calculated by taking the average of neighboring locations, and cohesion is the difference between the alignment and the current individual location. Attraction is the distance to the food source, and distraction is the sum of the enemy and current individual positions. The food source is the best solution, while the enemy is the worst solution ever. The position is updated using the following formula.

$$x_i^{t+1} = x_i^t + \Delta x_i^{t+1} \tag{13}$$

where x_i^t current position of ith individual, t is iteration counter and Δx_i^{t+1} is step vector formulated as follows:

$$\Delta x_i^{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta x_i^t \tag{14}$$

where s, a, c, f, and e are, respectively, the separation coefficient, the alignment coefficient, the cohesion coefficient, the attraction coefficient, and the distraction coefficient. w is inertia weight. S_i, A_i, C_i, F_i end E_i are the separation, alignment, cohesion, attraction, and distraction values, respectively. A dragonfly has a neighborhood within a certain radius. During iterations, the radius is increased to explore a global optimum. If there is no other dragonfly in the neighborhood, the dragonfly uses a random walk as follows:

$$x_i^{t+1} = x_i^t + \text{Levy(d)} \times x_i^t \tag{15}$$

where d is the dimension of the search space and *levy(.)* is the Levy flight function. Low cohesion and high alignment weights are used with a small radius. During iteration, the larger the radius, the smaller the alignment, the greater the cohesion. Accordingly, at each iteration, parameters should be updated until the stopping criterion is met.

3.1.7. Grasshopper Optimization Algorithm

GOA is an optimization algorithm that simulates the behavior of grasshopper swarms [10]. First, the parameters of GOA, such as maximum iteration, population size, *cmax*, and *cmin* are defined, and the fitness of each grasshopper is calculated. At each iteration, the c coefficient is updated as follows:

$$c = cmax - l\frac{cmax - cmin}{L} \tag{16}$$

where l is the current iteration, and L is the maximum iteration. *cmax* and *cmin* are the maximum and minimum values for c, respectively. After implementing this formula, for each grasshopper, the distances between grasshoppers are normalized into [1, 4], and the new positions are calculated as follows:

$$x_{i,d}^{t+1} = c\left(\sum_{\substack{j=1 \\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2}\left(fe^{\frac{-|x_{j,d}^t - x_{i,d}^t|}{l}} - e^{-|x_{j,d}^t - x_{i,d}^t|}\right)\frac{x_j^t - x_i^t}{\left|x_j^t - x_i^t\right|}\right) + \widehat{T}_d \tag{17}$$

Here, N is the total number of grasshoppers in the swarm, f is the intensity of attraction, l is the attractive length scale. ub_d and lb_d are the upper and lower bounds of the dth dimension of the search space. $\widehat{T}_d$ is the best position till now. The iterations continue until the maximum iteration is reached, and finally, the best position is returned as the solution of GHA.

3.1.8. Selfish Herd Optimizer

SHO is a swarm optimization algorithm [11]. The population consists of two groups: Pack predators (P) and Perd of Prey (H). Each member of the population has a survival value computed by the following equation.

$$SV_i = \frac{f_i - f_{best}}{f_{best} - f_{worst}} \tag{18}$$

where f_i is the fitness value of the position of the ith member. f_{best} and f_{worst} are the best and worst fitness values found so far by SHO. Through the SHO iterations, the herd movement operator is first applied for each member in H, which consists of two types of movements: herd's leader movement and herd's following and desertion movement. The position of the leader of a herd, which is the best in H so far, is updated as follows:

$$x_L^{t+1} = \begin{cases} x_L^t + c^t & if\ SV_L = 1 \\ x_L^t + s^t & if\ SV_L < 1 \end{cases} \tag{19}$$

where c^t is the motion vector that depends on the selfish repulsion experiment and s^t is the motion vector that depends on the selfish attraction experiment. Within H, the members except the leader are divided into groups of herd followers (H_F) and herd deserters (H_D), and each member moves according to the equation below.

$$x_i^{t+1} = \begin{cases} x_i^t + f_i^t & if\ i \in H_F^t \\ x_i^t + d_i^t & if\ i \in H_D^t \end{cases} \tag{20}$$

Here f_i^t is the following movement vector and d_i^t is the herd desertion vector in the iteration t. Then, the predator movement operator is applied to each member in P.

$$x_i^{t+1} = x_i^t + 2\rho\left(x_h^t - x_i^t\right) \tag{21}$$

Here x_h^t is the position of a randomly selected member in H, ρ is a random number in [0, 1]. The iteration of SHO ends after predation and restoration phases are performed. In the predation phase, members of the herd are killed by predators with the probability of being hunted. These members are removed from the population. In the restoration phase, new herd members are generated from the remaining herd members with the mating probability. After these phases, if the stop criterion is satisfied, the iteration is finished.

3.1.9. The Butterfly Optimization Algorithm

BOA is inspired by the food-foraging activities of butterflies [12]. The optimization process consists of three phases: The initial phase, the iteration phase, and the final phase. Each phase is executed sequentially. In the initial phase, randomly generated agents, called butterflies, are created to represent a candidate solution. Also, in this phase, the solution space and the fitness function are defined, and the algorithm parameters are initialized. The butterflies use the fragrance of other butterflies to find nectar, and each butterfly generates the fragrance according to its fitness value. The fragrance is calculated as follows:

$$f = cI^a \tag{22}$$

where c and a are the sensory modality and power exponent, respectively, and are in the range [0, 1]. I is the fitness value of the solution and is referred to as the stimulus intensity. In the second phase, iterations are performed until the stopping criteria are met. A new position of each butterfly is computed by one of the two main operations: global search and local search. A switch probability p determines whether the global search or the local

search is used. In global search, the direction of movement to find the new position is determined by using the best position g^* as follows:

$$x_i^{t+1} = x_i^t + \left(r^2 \times g^* - x_i^t\right) \times f_i \tag{23}$$

where x_i^t is the solution of the ith butterfly and r is a random number in [0, 1]. If r is greater than the switch probability, then the local search is performed instead, as follows:

$$x_i^{t+1} = x_i^t + \left(r^2 \times x_j^t - x_k^t\right) \times f_i \tag{24}$$

where x_j^t and d x_k^t are the solution of the ith and jth butterflies. The final phase of BOA indicates the achievement of the stopping criterion. In this phase, the best butterfly position is determined as the solution.

3.1.10. Tunicate Swarm Algorithm

TSA is a bio-inspired metaheuristic optimization algorithm [13]. Jet propulsion and swarm intelligence are two behaviors of tunicates used by TSA. Jet propulsion is modeled by three conditions. Avoiding conflicts between search agents, the first condition uses a vector to determine the new position of a tunicate as follows:

$$\vec{A} = \frac{c_2 + c_3 - 2c_1}{\lfloor P_{min} + c_1 P_{max} - P_{min} \rfloor} \tag{25}$$

Here c_1, c_2, and c_3 are random numbers in [0, 1]. P_{min} and P_{max} give the speed limits for social interaction. The second condition is the movement towards the best neighbor. For this purpose, the distance between the tunicate and the food source is calculated as follows:

$$\overrightarrow{PD} = \left|\overrightarrow{FS} - r\overrightarrow{P_p(x)}\right| \tag{26}$$

where $\overrightarrow{FS}$ represents the optimal location, x indicates the current iteration, and $\overrightarrow{P_p(x)}$ indicates the position of the agent. r is a random value in [0, 1]. Then, the last condition for modeling the jet propulsion is performed by converging to the best search agent as follows:

$$\overrightarrow{P_p(x)} = \begin{cases} \overrightarrow{FS} + \vec{A} \cdot \overrightarrow{PD} & if\ r \geq 0.5 \\ \overrightarrow{FS} - \vec{A} \cdot \overrightarrow{PD} & if\ r < 0.5 \end{cases} \tag{27}$$

When modeling the swarm behavior of tunicates, the best solution is used to update the positions of the agents in the swarm using the following equation.

$$\overrightarrow{P_p(x+1)} = \frac{\overrightarrow{P_p(x)} + \overrightarrow{P_p(x+1)}}{2c_1} \tag{28}$$

At each iteration, the TSA algorithm updates the search accents and their position with respect to the jet propulsion and swarm behavior and returns the best optimal position so far as the solution.

3.1.11. Tuna Swarm Optimization

TSO is a swarm-based metaheuristic optimization algorithm based on the cooperative foraging behavior of tuna schools [14]. The spiral and parabolic tuna foraging strategies are adapted to the TSO algorithm. Which strategy is used in an iteration is determined randomly. In the spiral strategy, the next position of the tunas is calculated as follows:

$$x_i^{t+1} = (\alpha + (1-\alpha)t/t_{max})(x_{best}^t + \beta|x_{best}^t - x_i^t|) + ((1-\alpha) - (1-\alpha)t/t_{max})x_{i-1}^t \tag{29}$$

where $\beta = e^{bl}\cos(2\pi b)$ and $l = e^{3\cos(((t_{max}+1/t)-1)\pi)}$. a is a constant, t is the number of iterations, t_{max} is the maximum number of iterations, i is tuna, and b is a random number in [0, 1]. To improve global exploration capability, a randomly generated position is swapped with x_{best}^t in the equation mentioned above, provided that a randomly generated value from the interval [0, 1] is greater than the ratio of t/t_{max}. In the case of using parabolic foraging to determine the next position in an iteration, the equation given below is used.

$$x_i^{t+1} = \begin{cases} x_{best}^t + r\left(x_{best}^t - x_i^t\right) + TF(1 - t/t_{max})^{2t/t_{max}}\left(x_{best}^t - x_i^t\right) & if\ r < 0.5 \\ TF(1 - t/t_{max})^{2t/t_{max}} x_i^t & if\ r \geq 0.5 \end{cases} \quad (30)$$

where TF is a random value in $[-1, 1]$. The TSO iterations continue to search for optimum results until the stopping criteria are met.

3.1.12. Cuckoo Search

CS is a metaheuristic optimization algorithm inspired by the obligate brood parasitic behavior of some cuckoo species [15]. An egg is a solution produced by a cuckoo and stored in a nest in CS. The initial population is created by randomly generated eggs for each host nest. Only one cuckoo is placed in a nest, and a new solution is generated from a randomly selected egg by the cuckoo as follows:

$$x_i^{t+1} = x_i^t + \alpha \times Levy(\lambda) \quad (31)$$

Here, α is the step size. In most cases α = 1 and $1 < \lambda \leq 3$. Each cuckoo places its egg in a randomly selected nest at a time if the new egg is better than the older one. The best nests, containing high-quality eggs, are retained to pass on to the next generation, while the p_a potion of the remainder is reproduced. These steps are repeated until the stopping criterion is met.

3.1.13. Salp Swarm Algorithm

SSA was proposed to solve engineering optimization problems using the navigation and foraging behavior of salps as a model [16]. The salp that has the best position is called the leading salp. The other members of the population are called follower salps, which form a salp chain to chase the position of the leading salp. First, an SSA population is randomly created, and the leading salp is determined. Then, the position of the leading salp is updated as follows:

$$x_1^{t+1} = \begin{cases} x_1^t + c_1((u-l)c_2 + l) & c_3 \geq 0 \\ x_1^t - c_1((u-l)c_2 + l) & c_3 < 0 \end{cases} \quad (32)$$

where x_1^t is the position of the leading salp at the t-th iteration. u and l are the upper and lower bounds of the search space, respectively. c_2 and c_3 are random numbers in [0, 1]. The value of c_1 is updated at each iteration as follows:

$$c_1 = 2e^{-\left(\frac{4t}{t_{max}}\right)^2} \quad (33)$$

Here, t_{max} is the maximum number of iterations, and t is the current iteration. After the position of the leader has been determined, the position of the followers is determined as follows:

$$x_i^{t+1} = 1/2\left(x_i^t + x_{i-1}^t\right) \quad (34)$$

Here, x_i^t is the i-th position of the follower salp $i \geq 2$. Until reaching the global optimum or stopping criteria, these processes are repeated.

3.2. Feed Forward Neural Network

While developing ANNs, as the name suggests, it was inspired by the nervous system of humans. ANN is successfully used to solve many problems, such as pattern classification, signal processing, image processing, and prediction. The most important feature that distinguishes ANNs from classical computer programs is their ability to learn. ANNs, or in short, neural networks, are classified under two main headings: FFNNs and RNNs. The main difference between the two models is that there is no feedback in FFNNs [46]. FFNNs can approximate any function assigned to it with targeted accuracy [47].

There are no closed paths in FFNN models. Input and output nodes are not interconnected among themselves; all other nodes are hidden nodes. Once the input nodes are set, the remaining points adjust their values by forward propagation. In FFNN, the output node values are determined by optimizing the model according to the values given at the input [48].

Artificial neurons are the heart of ANNs. We use many artificial neurons to form an ANN. In a conventional FFNN, there are three layers: the input layer, the hidden layer, and the output layer. As shown in Figure 1, a neuron produces an output by performing a series of operations on the data applied to its input. A FFNN can be formed by connecting the output of one neuron to the input of another neuron. The neuron model in Figure 1 has many inputs but only one output. Activation and transfer functions have a key role in the output [49].

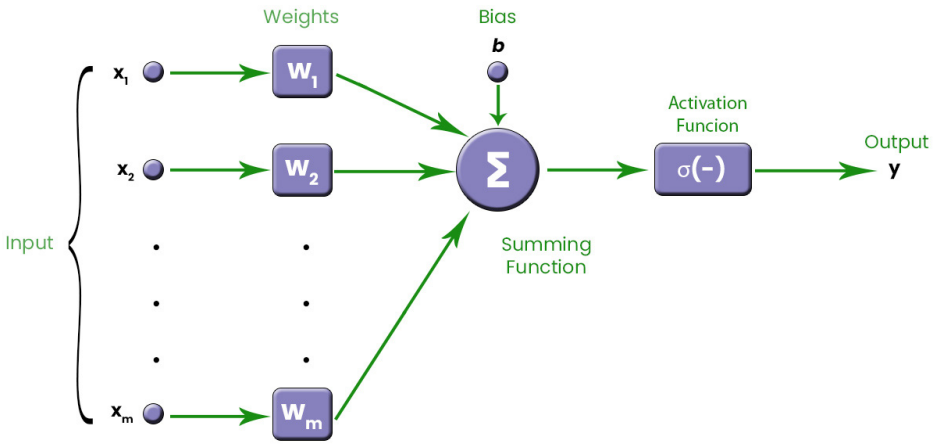

Figure 1. Conventional model of an artificial neuron.

Input values are weighted using w values. The bias value is denoted by b. The activation function is denoted by f. The output of the artificial neuron is denoted by y. There are different activation functions used in ANNs. One of the commonly used functions among these functions is the sigmoid function. In this study, we use the sigmoid function. The calculations performed in an artificial neuron are as follows:

$$y = f\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{35}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{36}$$

Weights and bias values have a significant effect on the training of the network. The total number of parameters to be optimized depends on the number of inputs, the total number of neurons, and their weights.

4. Simulation Results

In this study, FFNN was trained by using swarm-intelligent-based metaheuristic algorithms for MPPT, and their performances were analyzed. 13 algorithms were used for

FFNN training. These algorithms are ABC, BOA, CS, CSO, DA, FA, GOA, KHA, PSO, SHO, SSA, TSA, and TSO.

One of the most important techniques used to reach the maximum power of alternative energy sources is MPPT. Although it is used in different energy sources, its most intensive use is on wind turbines and solar panels [50]. This study is on the MPPT of the solar PV system. The simplified electrical equivalent circuit of a PV cell is given in Figure 2. It is possible to calculate the output current and output voltage of a PV cell via (37) and (38). Here, *pvcell* is PV cell current depending on temperature and solar radiation. *Id* is diode current. Ip is parallel resistor current. Vd is diode voltage drop. Ipv is the PV cell output current. *Rs* is serial resistor [50]. In this study, the data of 250 W solar panels was used. The inputs of the system are temperature (t) and solar radiation (s). The power value (p) is the output of the system. As here, the data set is scaled in the range of [0, 1] due to the large values. All analyses were performed according to the scaled data set.

$$Ipv = Ipvcell - Id - Ip \tag{37}$$

$$Vpv = Vd - Ip * Rs \tag{38}$$

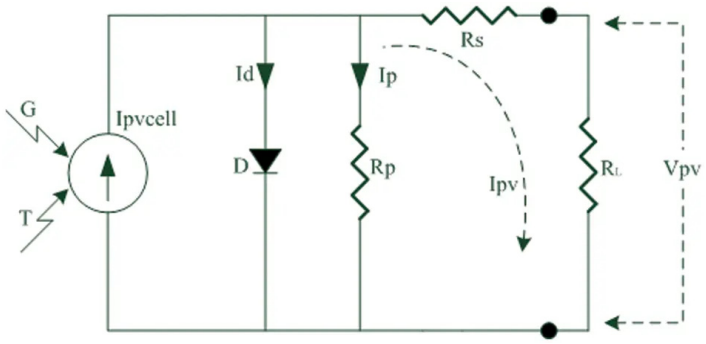

Figure 2. The simplified electrical equivalent circuit of a PV cell [50].

Approximately 80% of the data set was used for the training process. The rest is devoted to the testing process. The block diagram of the training process to be carried out for MPPT is presented in Figure 3. As seen here, a training process is separately carried out with ABC, BOA, CS, CSO, DA, FA, GOA, KHA, PSO, SHO, SSA, TSA, and TSO algorithms. The output p is obtained corresponding to the inputs t and s. The difference between the estimated output and the real output gives the error. It is aimed to minimize this error at the end of the training process. A zero error indicates that the real system is optimally modeled.

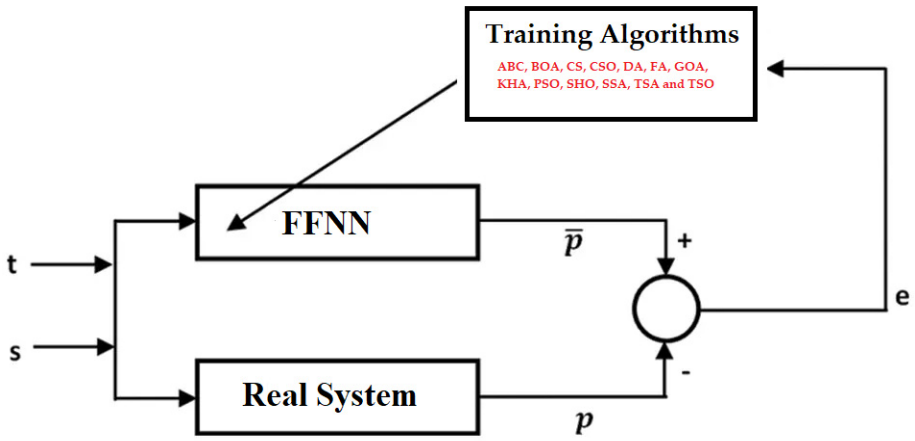

Figure 3. Block diagram of FFNN-based MPPT modeling.

Each application was run at least 30 times in order to obtain statistical results. The mean squared error was used as the error metric. In the training process, results for three different population sizes were obtained for each algorithm. These values are 10, 20, and 50. The numbers of maximum generation corresponding to these population sizes are 100, 50, and 20, respectively. Other control parameters used for algorithms are as follows. The "limit" value of the ABC algorithm is calculated as (NP×D)/2. NP is the number of population size. D is the number of parameters of the problem to be optimized. The probability switch, power exponent, and sensory modality values of BOA are 0.8, 0.1, and 0.01, respectively. The discovery rate of alien eggs/solutions of CS is 0.25. The light absorption coefficient, attraction coefficient base value, mutation coefficient, and mutation coefficient damping ratio of FA are 1, 2, 0.2, and 0.98, respectively. The inertia weights of PSO are 0.9 and 0.6. The constant (a) of TSO is 0.7.

The FFNN model, consisting of two inputs and one output, was created to solve the related problem. The results obtained for models with 5, 10, and 15 neurons in the hidden layer are examined. In other words, 2-5-1, 2-10-1, and 2-15-1 network structures were used in the applications. Only three different network structures were taken into consideration, especially due to the high number of algorithms, the fact that each application was run 30 times, and results in different population sizes were obtained. In other words, high processing times are avoided. This is one of the limitations of this study. These results in this study should be considered within limitations. The summing function is used as the transfer function in FFNN. Sigmoid was chosen as the activation function. The optimized parameter numbers for 2-5-1, 2-10-1, and 2-15-1 network structures are 21, 41, and 61, respectively.

The results obtained with 13 metaheuristic algorithms for n = 10 are given in Table 1. The most effective training and test error values of the ABC algorithm were found to be 2-5-1. The training and test error values obtained with the ABC algorithm are 5.0×10^{-3} and 5.1×10^{-3}, respectively. The worst training and test results were achieved with the BOA. Increasing the number of neurons in CS worsened the results. The best results in both training and testing were obtained with the 2-5-1 network structure. These results are 3.9×10^{-3} and 3.9×10^{-3}, respectively. The best results in CSO were obtained with the 2-10-1 network structure. The training error value of CSO is 3.7×10^{-3}. The test error value found is 3.8×10^{-3}. Increasing the number of neurons in DA worsened the performance. The best training and test error values were obtained in the 2-5-1 network structure. Effective results were achieved in all network structures in FA. The best training and test error values were found as 1.8×10^{-3} and 1.8×10^{-3}, respectively, via a 2-10-1 network structure. Increasing the number of neurons in GOA improved performance. The best training and test error values were obtained with 2-15-1. As in the GOA, the best results were achieved with 2-15-1 in KHA. The training result of KHA is 5.5×10^{-3}. The test is 5.6×10^{-3}. Network structure with low neuron count was more effective in PSO. Better results were found with 2-5-1. Effective results were achieved with all network structures in SHO. The best training and test results are 3.5×10^{-3} and 3.6×10^{-3}, respectively. The 2-15-1 network structure has been effective in SSA. The best training error value of SSA is 4.0×10^{-3}. The test error value is 4.1×10^{-3}. The most effective network structure in TSA is 2-10-1. The training and test error values are 2.5×10^{-3} and 2.6×10^{-3}, respectively. As in TSA, the 2-10-1 network structure is more effective in TSO.

Table 1. Comparison of the results obtained for n = 10.

System	Network Structure	The Results			
		Train		Test	
		Mean	Std.	Mean	Std.
ABC	2-5-1	5.0×10^{-3}	2.4×10^{-3}	5.1×10^{-3}	2.4×10^{-3}
	2-10-1	1.2×10^{-2}	4.3×10^{-2}	1.8×10^{-2}	7.6×10^{-2}
	2-15-1	6.0×10^{-3}	3.2×10^{-3}	6.0×10^{-3}	3.1×10^{-3}
BOA	2-5-1	5.2×10^{-2}	4.1×10^{-2}	5.2×10^{-2}	4.1×10^{-2}
	2-10-1	3.8×10^{-2}	3.5×10^{-2}	3.8×10^{-2}	3.5×10^{-2}
	2-15-1	4.1×10^{-2}	3.8×10^{-2}	4.1×10^{-2}	3.8×10^{-2}
CS	2-5-1	3.9×10^{-3}	1.1×10^{-3}	3.9×10^{-3}	1.1×10^{-3}
	2-10-1	4.0×10^{-3}	1.2×10^{-3}	4.1×10^{-3}	1.2×10^{-3}
	2-15-1	4.3×10^{-3}	1.6×10^{-3}	4.4×10^{-3}	1.6×10^{-3}
CSO	2-5-1	5.3×10^{-3}	2.9×10^{-3}	5.3×10^{-3}	2.9×10^{-3}
	2-10-1	3.7×10^{-3}	2.6×10^{-3}	3.8×10^{-3}	2.6×10^{-3}
	2-15-1	4.6×10^{-3}	2.6×10^{-3}	4.7×10^{-3}	2.6×10^{-3}
DA	2-5-1	5.0×10^{-3}	2.9×10^{-3}	5.0×10^{-3}	2.9×10^{-3}
	2-10-1	5.3×10^{-3}	3.4×10^{-3}	5.4×10^{-3}	3.5×10^{-3}
	2-15-1	5.7×10^{-3}	3.5×10^{-3}	5.7×10^{-3}	3.4×10^{-3}
FA	2-5-1	2.0×10^{-3}	2.0×10^{-3}	2.0×10^{-3}	2.0×10^{-3}
	2-10-1	1.8×10^{-3}	1.9×10^{-3}	1.8×10^{-3}	1.9×10^{-3}
	2-15-1	1.9×10^{-3}	1.7×10^{-3}	1.9×10^{-3}	1.8×10^{-3}
GOA	2-5-1	1.3×10^{-2}	3.4×10^{-2}	1.3×10^{-2}	3.5×10^{-2}
	2-10-1	1.2×10^{-2}	4.0×10^{-2}	1.2×10^{-2}	4.0×10^{-2}
	2-15-1	4.9×10^{-3}	3.1×10^{-3}	4.9×10^{-3}	3.2×10^{-3}
KHA	2-5-1	5.6×10^{-3}	7.3×10^{-3}	5.6×10^{-3}	7.3×10^{-3}
	2-10-1	6.9×10^{-3}	8.7×10^{-3}	7.0×10^{-3}	9.0×10^{-3}
	2-15-1	5.5×10^{-3}	5.8×10^{-3}	5.6×10^{-3}	6.0×10^{-3}
PSO	2-5-1	6.5×10^{-3}	2.6×10^{-3}	6.5×10^{-3}	2.6×10^{-3}
	2-10-1	9.0×10^{-3}	3.0×10^{-3}	9.1×10^{-3}	3.1×10^{-3}
	2-15-1	8.6×10^{-3}	3.0×10^{-3}	8.6×10^{-3}	3.0×10^{-3}
SHO	2-5-1	3.7×10^{-3}	2.5×10^{-3}	3.7×10^{-3}	2.6×10^{-3}
	2-10-1	3.5×10^{-3}	2.1×10^{-3}	3.6×10^{-3}	2.1×10^{-3}
	2-15-1	3.6×10^{-3}	3.3×10^{-3}	3.7×10^{-3}	3.3×10^{-3}
SSA	2-5-1	4.7×10^{-3}	3.2×10^{-3}	4.7×10^{-3}	3.2×10^{-3}
	2-10-1	4.1×10^{-3}	2.3×10^{-3}	4.1×10^{-3}	2.2×10^{-3}
	2-15-1	4.0×10^{-3}	3.1×10^{-3}	4.1×10^{-3}	3.1×10^{-3}
TSA	2-5-1	1.5×10^{-2}	3.4×10^{-2}	1.5×10^{-2}	3.4×10^{-2}
	2-10-1	2.5×10^{-3}	1.6×10^{-3}	2.6×10^{-3}	1.7×10^{-3}
	2-15-1	2.9×10^{-3}	1.4×10^{-3}	2.9×10^{-3}	1.4×10^{-3}
TSO	2-5-1	3.3×10^{-3}	1.9×10^{-3}	3.3×10^{-3}	1.8×10^{-3}
	2-10-1	2.7×10^{-3}	1.5×10^{-3}	2.8×10^{-3}	1.5×10^{-3}
	2-15-1	3.0×10^{-3}	1.7×10^{-3}	3.1×10^{-3}	1.7×10^{-3}

The results obtained with 13 metaheuristic algorithms for n = 20 are given in Table 2. 2-5-1 is the most effective network structure in the ABC algorithm. As with n = 10, the results of BOA are ineffective at n = 20. 2-5-1 is more effective in CS. The training and test results of CS are 5.5×10^{-3} and 5.6×10^{-3}, respectively. The training error value found in the CSO is 3.9×10^{-3}. The test error value is 4.0×10^{-3}. An increase in the number of neurons in DA improved performance. The most effective results were achieved with the 2-5-1 network structure in FA. The training and test results of the FA are 5.0×10^{-4} and 5.2×10^{-4}, respectively. Effective and similar results were achieved with all network structures in GOA. The increasing population size in KHA worsened performance. Unsuccessful results were found with all network structures. The most effective results in PSO were achieved with 2-10-1. For n = 20, SHO is successful. Increasing the number of neurons in SHO improved performance. Effective results were achieved with 2-15-1 in SHO. The training and test error values found are 1.6×10^{-3} and 1.6×10^{-3}, respectively. In SSA, 2-15-1 is more effective than other network structures. Effective results were achieved with 2-10-1 in TSA. Training and test results of TSA are 2.5×10^{-3} and 2.5×10^{-3}, respectively. 2-10-1 and 2-15-1 network structures are successful in TSO.

Table 2. Comparison of the results obtained for n = 20.

System	Network Structure	The Results			
		Train		Test	
		Mean	Std.	Mean	Std.
ABC	2-5-1	7.0×10^{-3}	2.3×10^{-3}	7.1×10^{-3}	2.4×10^{-3}
	2-10-1	8.9×10^{-3}	8.2×10^{-3}	8.9×10^{-3}	8.5×10^{-3}
	2-15-1	1.3×10^{-2}	1.4×10^{-2}	5.6×10^{-2}	1.2×10^{-1}
BOA	2-5-1	3.7×10^{-2}	3.1×10^{-2}	3.7×10^{-2}	3.1×10^{-2}
	2-10-1	2.1×10^{-2}	1.8×10^{-2}	2.2×10^{-2}	1.8×10^{-2}
	2-15-1	2.6×10^{-2}	2.5×10^{-2}	2.6×10^{-2}	2.5×10^{-2}
CS	2-5-1	5.5×10^{-3}	1.6×10^{-3}	5.6×10^{-3}	1.6×10^{-3}
	2-10-1	5.8×10^{-3}	1.7×10^{-3}	5.8×10^{-3}	1.7×10^{-3}
	2-15-1	6.5×10^{-3}	2.1×10^{-3}	6.6×10^{-3}	2.2×10^{-3}
CSO	2-5-1	3.9×10^{-3}	2.3×10^{-3}	4.0×10^{-3}	2.4×10^{-3}
	2-10-1	4.7×10^{-3}	1.4×10^{-3}	4.8×10^{-3}	1.5×10^{-3}
	2-15-1	6.9×10^{-3}	2.3×10^{-3}	7.1×10^{-3}	2.5×10^{-3}
DA	2-5-1	4.3×10^{-3}	3.3×10^{-3}	4.4×10^{-3}	3.4×10^{-3}
	2-10-1	4.0×10^{-3}	3.2×10^{-3}	4.1×10^{-3}	3.3×10^{-3}
	2-15-1	3.5×10^{-3}	2.1×10^{-3}	3.5×10^{-3}	2.1×10^{-3}
FA	2-5-1	5.0×10^{-4}	3.9×10^{-4}	5.2×10^{-4}	4.0×10^{-4}
	2-10-1	8.0×10^{-4}	9.6×10^{-4}	8.3×10^{-4}	1.0×10^{-3}
	2-15-1	5.4×10^{-4}	4.6×10^{-4}	5.4×10^{-4}	4.7×10^{-4}
GOA	2-5-1	3.5×10^{-3}	2.3×10^{-3}	3.6×10^{-3}	2.4×10^{-3}
	2-10-1	3.4×10^{-3}	2.5×10^{-3}	3.5×10^{-3}	2.5×10^{-3}
	2-15-1	3.7×10^{-3}	3.1×10^{-3}	3.7×10^{-3}	3.1×10^{-3}

Table 2. *Cont.*

System	Network Structure	The Results			
		Train		Test	
		Mean	Std.	Mean	Std.
KHA	2-5-1	1.5×10^{-2}	1.2×10^{-2}	1.5×10^{-2}	1.2×10^{-2}
	2-10-1	1.5×10^{-2}	7.4×10^{-3}	1.5×10^{-2}	7.3×10^{-3}
	2-15-1	1.9×10^{-2}	1.4×10^{-2}	1.9×10^{-2}	1.4×10^{-2}
PSO	2-5-1	1.0×10^{-2}	4.4×10^{-3}	1.0×10^{-2}	4.5×10^{-3}
	2-10-1	9.0×10^{-3}	3.7×10^{-3}	9.0×10^{-3}	3.7×10^{-3}
	2-15-1	9.4×10^{-3}	3.2×10^{-3}	9.5×10^{-3}	3.2×10^{-3}
SHO	2-5-1	2.7×10^{-3}	1.7×10^{-3}	2.8×10^{-3}	1.7×10^{-3}
	2-10-1	1.9×10^{-3}	9.9×10^{-4}	2.0×10^{-3}	1.0×10^{-3}
	2-15-1	1.6×10^{-3}	6.3×10^{-4}	1.6×10^{-3}	6.5×10^{-4}
SSA	2-5-1	5.0×10^{-3}	2.9×10^{-3}	5.0×10^{-3}	3.0×10^{-3}
	2-10-1	5.6×10^{-3}	3.6×10^{-3}	5.7×10^{-3}	3.7×10^{-3}
	2-15-1	4.4×10^{-3}	2.7×10^{-3}	4.5×10^{-3}	2.7×10^{-3}
TSA	2-5-1	5.1×10^{-3}	8.5×10^{-3}	5.1×10^{-3}	7.9×10^{-3}
	2-10-1	2.5×10^{-3}	1.3×10^{-3}	2.6×10^{-3}	1.3×10^{-3}
	2-15-1	2.9×10^{-3}	1.5×10^{-3}	3.0×10^{-3}	1.9×10^{-3}
TSO	2-5-1	3.6×10^{-3}	2.4×10^{-3}	3.6×10^{-3}	2.3×10^{-3}
	2-10-1	3.2×10^{-3}	1.8×10^{-3}	3.3×10^{-3}	1.8×10^{-3}
	2-15-1	3.3×10^{-3}	1.5×10^{-3}	3.4×10^{-3}	1.6×10^{-3}

The results obtained with 13 metaheuristic algorithms for $n = 50$ are given in Table 3. The results obtained for $n = 50$ in the ABC algorithm are unsuccessful. The training and test error values found in the 2-10-1 network structure with BOA are 1.5×10^{-2} and 1.5×10^{-2}, respectively. In CS and CSO, 2-10-1 network structures are more effective than other network structures. The training error value found with the 2-5-1 network structure in DA is 3.8×10^{-3}. Its test error value is 3.9×10^{-3}. Effective results have been achieved with all network structures in FA. The most effective network structure is 2-5-1. The training and test error values found are 4.5×10^{-4} and 4.6×10^{-4}, respectively. Successful training and test results were achieved with 2-10-1 in GOA. The results found for $n = 50$ in KHA and PSO are unsuccessful. Successful results were achieved in SHO. The 2-10-1 network structure is effective. The best training and test error values found are 1.9×10^{-3} and 2.0×10^{-3}, respectively. The training and test error values found with 2-10-1 in the SSA are 5.3×10^{-3} and 5.4×10^{-3}, respectively. Effective results were found with the 2-15-1 network structure in TSA. The training and test error values found with the 2-5-1 network structure in TSO are 4.1×10^{-3}.

Table 3. Comparison of the results obtained for $n = 50$.

System	Network Structure	The Results			
		Train		Test	
		Mean	Std.	Mean	Std.
ABC	2-5-1	1.2×10^{-2}	1.0×10^{-2}	1.3×10^{-2}	1.0×10^{-2}
	2-10-1	1.1×10^{-2}	5.0×10^{-3}	1.1×10^{-2}	5.0×10^{-3}
	2-15-1	1.3×10^{-2}	6.2×10^{-3}	1.3×10^{-2}	6.0×10^{-3}
BOA	2-5-1	2.5×10^{-2}	1.6×10^{-2}	2.5×10^{-2}	1.6×10^{-2}
	2-10-1	1.5×10^{-2}	1.2×10^{-2}	1.5×10^{-2}	1.2×10^{-2}
	2-15-1	1.8×10^{-2}	1.2×10^{-2}	1.8×10^{-2}	1.2×10^{-2}
CS	2-5-1	8.2×10^{-3}	2.3×10^{-3}	8.3×10^{-3}	2.3×10^{-3}
	2-10-1	7.5×10^{-3}	2.4×10^{-3}	7.6×10^{-3}	2.4×10^{-3}
	2-15-1	8.1×10^{-3}	3.0×10^{-3}	8.2×10^{-3}	2.9×10^{-3}
CSO	2-5-1	7.3×10^{-3}	2.9×10^{-3}	7.3×10^{-3}	2.9×10^{-3}
	2-10-1	7.0×10^{-3}	2.6×10^{-3}	7.1×10^{-3}	2.6×10^{-3}
	2-15-1	7.6×10^{-3}	2.8×10^{-3}	7.8×10^{-3}	2.8×10^{-3}
DA	2-5-1	3.8×10^{-3}	3.0×10^{-3}	3.9×10^{-3}	3.0×10^{-3}
	2-10-1	4.6×10^{-3}	3.5×10^{-3}	4.7×10^{-3}	3.5×10^{-3}
	2-15-1	4.1×10^{-3}	2.5×10^{-3}	4.2×10^{-3}	2.6×10^{-3}
FA	2-5-1	4.5×10^{-4}	2.3×10^{-4}	4.6×10^{-4}	2.3×10^{-4}
	2-10-1	5.5×10^{-4}	2.8×10^{-4}	5.7×10^{-4}	2.8×10^{-4}
	2-15-1	8.6×10^{-4}	5.5×10^{-4}	8.9×10^{-4}	5.8×10^{-4}
GOA	2-5-1	2.7×10^{-3}	1.9×10^{-3}	2.8×10^{-3}	1.9×10^{-3}
	2-10-1	2.3×10^{-3}	1.6×10^{-3}	2.4×10^{-3}	1.7×10^{-3}
	2-15-1	3.3×10^{-3}	2.2×10^{-3}	3.4×10^{-3}	2.3×10^{-3}
KHA	2-5-1	1.6×10^{-2}	6.9×10^{-3}	1.6×10^{-2}	6.8×10^{-3}
	2-10-1	1.8×10^{-2}	7.2×10^{-3}	1.8×10^{-2}	7.2×10^{-3}
	2-15-1	1.6×10^{-2}	5.6×10^{-3}	1.6×10^{-2}	5.6×10^{-3}
PSO	2-5-1	1.1×10^{-2}	4.0×10^{-3}	1.1×10^{-2}	4.1×10^{-3}
	2-10-1	1.0×10^{-2}	3.9×10^{-3}	1.0×10^{-2}	3.9×10^{-3}
	2-15-1	1.1×10^{-2}	3.8×10^{-3}	1.1×10^{-2}	3.7×10^{-3}
SHO	2-5-1	2.3×10^{-3}	1.4×10^{-3}	2.3×10^{-3}	1.5×10^{-3}
	2-10-1	1.9×10^{-3}	8.4×10^{-4}	2.0×10^{-3}	8.9×10^{-4}
	2-15-1	3.3×10^{-3}	4.2×10^{-3}	3.3×10^{-3}	4.1×10^{-3}
SSA	2-5-1	7.4×10^{-3}	3.8×10^{-3}	7.6×10^{-3}	3.8×10^{-3}
	2-10-1	5.3×10^{-3}	2.8×10^{-3}	5.4×10^{-3}	2.8×10^{-3}
	2-15-1	5.7×10^{-3}	3.8×10^{-3}	5.8×10^{-3}	3.9×10^{-3}
TSA	2-5-1	3.7×10^{-3}	2.6×10^{-3}	3.7×10^{-3}	2.6×10^{-3}
	2-10-1	3.4×10^{-3}	2.3×10^{-3}	3.5×10^{-3}	2.3×10^{-3}
	2-15-1	3.1×10^{-3}	1.3×10^{-3}	3.2×10^{-3}	1.4×10^{-3}
TSO	2-5-1	4.1×10^{-3}	2.5×10^{-3}	4.1×10^{-3}	2.6×10^{-3}
	2-10-1	5.3×10^{-3}	2.8×10^{-3}	5.3×10^{-3}	2.8×10^{-3}
	2-15-1	5.7×10^{-3}	2.6×10^{-3}	5.9×10^{-3}	2.6×10^{-3}

5. Discussion

Network structure and population size significantly affect the performance of 13 algorithms. Some algorithms are more effective at low population sizes, while others are more effective at higher population sizes. The same is true for the network structure. The change in the number of neurons in the hidden layer affects the performance. Information on the best train mean error values obtained using related metaheuristic algorithms is given in Table 4. The best results with ABC, CS, CSO, KHA, PSO, SSA, TSA, and TSO algorithms were obtained when n = 10. The most effective results with DA and SHO were found when n = 20. BOA, FA, and GOA achieved more effective results with n = 50. In other words, the most successful results in 8 of 13 algorithms were obtained when n = 10. This number was 2 for n = 20, while this value was 3 for n = 50. The most effective results with ABC, CS, FA, and PSO were obtained with the 2-5-1 network structure. On the other hand, BOA, CSO, GOA, TSA, and TSO achieved better results in the 2-10-1 network structure. Other algorithms found their best results with a 2-15-1 network structure. In other words, the most effective results of the five algorithms were achieved with the 2-10-1 network structure. For other network structures, this value is four. Information on the best test mean error values obtained using related metaheuristic algorithms is given in Table 5. All of the evaluations made for Table 4 are valid for Table 5. In other words, the parameters in the training and testing processes showed parallelism with each other.

Table 4. Information on the best train mean error values obtained using related metaheuristic algorithms.

Algorithm	Train			
	Network Structure	Population Size	Mean	Std.
ABC	2-5-1	10	5.0×10^{-3}	2.4×10^{-3}
BOA	2-10-1	50	1.5×10^{-2}	1.2×10^{-2}
CS	2-5-1	10	3.9×10^{-3}	1.1×10^{-3}
CSO	2-10-1	10	3.7×10^{-3}	2.6×10^{-3}
DA	2-15-1	20	3.5×10^{-3}	2.1×10^{-3}
FA	2-5-1	50	4.5×10^{-4}	2.3×10^{-4}
GOA	2-10-1	50	2.3×10^{-3}	1.6×10^{-3}
KHA	2-15-1	10	5.5×10^{-3}	5.8×10^{-3}
PSO	2-5-1	10	6.5×10^{-3}	2.6×10^{-3}
SHO	2-15-1	20	1.6×10^{-3}	6.3×10^{-4}
SSA	2-15-1	10	4.0×10^{-3}	3.1×10^{-3}
TSA	2-10-1	10	2.5×10^{-3}	1.6×10^{-3}
TSO	2-10-1	10	2.7×10^{-3}	1.5×10^{-3}

According to the results in Tables 4 and 5, the success rankings of 13 swarm intelligent-based training algorithms in the training and testing process were compared for the solution of the MPPT problem in Table 6. The most effective result was found with FA in both training and testing processes. The training and test error values obtained are 4.5×10^{-4} and 4.6×10^{-4}, respectively. After FA, the most successful algorithm is SHO. The best training and testing values found with SHO are 1.6×10^{-3}. The third most successful algorithm is GOA. The best training error achieved with GOA is 2.3×10^{-3}. The test error value of GOA is 2.4×10^{-3}. The fourth successful algorithm is TSA. The fifth algorithm is TSO. The next ranks according to performance are as follows: DA, CSO, CS, SSA, ABC, KHA, PSO, and BOA.

Table 5. Information on the best test mean error values obtained using related metaheuristic algorithms.

Algorithm	Test			
	Network Structure	Population Size	Mean	Std.
ABC	2-5-1	10	5.1×10^{-3}	2.4×10^{-3}
BOA	2-10-1	50	1.5×10^{-2}	1.2×10^{-2}
CS	2-5-1	10	3.9×10^{-3}	1.1×10^{-3}
CSO	2-10-1	10	3.8×10^{-3}	2.6×10^{-3}
DA	2-15-1	20	3.5×10^{-3}	2.1×10^{-3}
FA	2-5-1	50	4.6×10^{-4}	2.3×10^{-4}
GOA	2-10-1	50	2.4×10^{-3}	1.7×10^{-3}
KHA	2-15-1	10	5.6×10^{-3}	6.0×10^{-3}
PSO	2-5-1	10	6.5×10^{-3}	2.6×10^{-3}
SHO	2-15-1	20	1.6×10^{-3}	6.5×10^{-4}
SSA	2-15-1	10	4.1×10^{-3}	3.1×10^{-3}
TSA	2-10-1	10	2.6×10^{-3}	1.7×10^{-3}
TSO	2-10-1	10	2.8×10^{-3}	1.5×10^{-3}

Table 6. General success scores according to the best results of related metaheuristic algorithms.

Order	Algorithm	Train Ranking Score	Test Ranking Score	Total Score
1	FA	1	1	2
2	SHO	2	2	4
3	GOA	3	3	6
4	TSA	4	4	8
5	TSO	5	5	10
6	DA	6	6	12
7	CSO	7	7	14
8	CS	8	9	17
9	SSA	9	8	17
10	ABC	10	10	20
11	KHA	11	11	22
12	PSO	12	12	24
13	BOA	13	13	26

6. Conclusions

In this study, the performance of thirteen swarms of intelligence-based algorithms in FFNN training for MPPT was evaluated. These algorithms are ABC, BOA, CS, CSO, DA, FA, GOA, KHA, PSO, SHO, SSA, TSA, and TSO. Three different network structures are used to analyze the effect of the network structure of the ANN on the results. They are 2-5-1, 2-10-1 and 2-15-1. At the same time, the effect of population size, which is one of the important control parameters, on the results was investigated. The results are obtained for values of 10, 20 and 50. The general results of this study are as follows:

- In general, all algorithms were found to be effective for MPPT. The three most effective algorithms are FA, SHO, and GOA.

- Network structure affects the performance of training algorithms. The network structure in which each algorithm is more successful may be different from each other.
- As with the network structure, the population size affects the performance of the training algorithms in solving the related problem. The population size in which each algorithm is more successful may differ.
- In general, the training and test results for each algorithm were close to each other. This shows that the learning process is successful.

This study is one of the most comprehensive studies based on swarm intelligence, and the performance of thirteen algorithms is compared specifically to MPPT. However, there are many problems in the real world, and the results of this study will shed light on future studies. We will evaluate the performances of metaheuristic algorithms based on swarm intelligence in the future in solving problems in many fields, such as engineering, economics, social sciences, educational sciences, and medicine.

Author Contributions: E.K.: conceptualization, methodology, validation, software, review and editing, original draft preparation, supervision; C.B.K.: software, writing, data curation, example analysis, visualization, original draft preparation; E.B.: conceptualization, methodology, original draft preparation, writing; S.A.: methodology, original draft preparation, writing; B.Ö.: methodology, original draft preparation, writing; B.Y.: methodology, original draft preparation, writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BOA	Butterfly optimization algorithm
SSA	Salp swarm algorithm
FA	Firefly algorithm
ABC	Artificial bee colony
PSO	Particle swarm optimization
KHA	Krill herd algorithm
CS	Cuckoo search
FFNN	Feed-forward neural network
ANN	Artificial neural network
GOA	Grasshopper optimization algorithm
ANFIS	Adaptive Network Fuzzy Inference System
DA	Dragonfly algorithm
SHO	Selfish herd optimizer
TSA	Tunicate swarm algorithm
TSO	Tuna swarm optimization
CSO	Chicken swarm optimization
MPP	Maximum power point
MPPT	Maximum power point tracking
PV	Photovoltaic
P&O	Perturb and observe
INC	Incremental Conductance
PID	Proportional Integral Derivative
FLC	Fuzzy Logic Control
PS	Photovoltaic System
SGO	Social group optimization
SMO	Slime mould optimization

RNNs	Recurrent neural networks
ACO	Ant colony optimization
RBFN	Radial basis function network

References

1. Mellit, A.; Kalogirou, S.A. MPPT-based artificial intelligence techniques for photovoltaic systems and its implementation into field programmable gate array chips: Review of current status and future perspectives. *Energy* **2014**, *70*, 1–21. [CrossRef]
2. Villegas-Mier, C.G.; Rodriguez-Resendiz, J.; Álvarez-Alvarado, J.M.; Rodriguez-Resendiz, H.; Herrera-Navarro, A.M.; Rodríguez-Abreo, O. Artificial neural networks in MPPT algorithms for optimization of photovoltaic power systems: A review. *Micromachines* **2021**, *12*, 1260. [CrossRef]
3. Al-Majidi, S.D.; Abbod, M.F.; Al-Raweshidy, H.S. A particle swarm optimisation-trained feedforward neural network for predicting the maximum power point of a photovoltaic array. *Eng. Appl. Artif. Intell.* **2020**, *92*, 103688. [CrossRef]
4. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
5. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
6. Yang, X.-S. Firefly algorithms for multimodal optimization. In Proceedings of the Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, 26–28 October 2009; pp. 169–178.
7. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
8. Meng, X.; Liu, Y.; Gao, X.; Zhang, H. A new bio-inspired algorithm: Chicken swarm optimization. In Proceedings of the Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, 17–20 October 2014; pp. 86–94.
9. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
10. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
11. Fausto, F.; Cuevas, E.; Valdivia, A.; González, A. A global optimization algorithm inspired in the behavior of selfish herds. *Biosystems* **2017**, *160*, 39–55. [CrossRef] [PubMed]
12. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
13. Kaur, S.; Awasthi, L.K.; Sangal, A.; Dhiman, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541. [CrossRef]
14. Xie, L.; Han, T.; Zhou, H.; Zhang, Z.-R.; Han, B.; Tang, A. Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization. *Comput. Intell. Neurosci.* **2021**, *2021*, 9210050. [CrossRef] [PubMed]
15. Yang, X.-S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
16. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
17. Yang, B.; Zhong, L.; Zhang, X.; Shu, H.; Yu, T.; Li, H.; Jiang, L.; Sun, L. Novel bio-inspired memetic salp swarm algorithm and application to MPPT for PV systems considering partial shading condition. *J. Clean. Prod.* **2019**, *215*, 1203–1222. [CrossRef]
18. Manna, S.; Akella, A.K.; Singh, D.K. Implementation of a novel robust model reference adaptive controller-based MPPT for stand-alone and grid-connected photovoltaic system. *Energy Sources Part A Recovery Util. Environ. Eff.* **2023**, *45*, 1321–1345. [CrossRef]
19. Kamarposhti, M.A.; Shokouhandeh, H.; Colak, I.; Eguchi, K. Optimization of Adaptive Fuzzy Controller for Maximum Power Point Tracking Using Whale Algorithm. *CMC-Comput. Mater. Contin.* **2022**, *73*, 5041–5061. [CrossRef]
20. Akram, N.; Khan, L.; Agha, S.; Hafeez, K. Global Maximum Power Point Tracking of Partially Shaded PV System Using Advanced Optimization Techniques. *Energies* **2022**, *15*, 4055. [CrossRef]
21. Vadivel, S.; Sengodan, B.C.; Ramasamy, S.; Ahsan, M.; Haider, J.; Rodrigues, E.M. Social Grouping Algorithm Aided Maximum Power Point Tracking Scheme for Partial Shaded Photovoltaic Array. *Energies* **2022**, *15*, 2105. [CrossRef]
22. Kacimi, N.; Idir, A.; Grouni, S.; Boucherit, M.S. A new combined method for tracking the global maximum power point of photovoltaic systems. *Rev. Roum. Des Sci. Tech. Série Électrotechnique Énergétique* **2022**, *67*, 349–354.
23. Mirza, A.F.; Mansoor, M.; Ling, Q.; Yin, B.; Javed, M.Y. A Salp-Swarm Optimization based MPPT technique for harvesting maximum energy from PV systems under partial shading conditions. *Energy Convers. Manag.* **2020**, *209*, 112625. [CrossRef]
24. Jamshidi, F.; Salehizadeh, M.R.; Yazdani, R.; Azzopardi, B.; Jately, V. An Improved Sliding Mode Controller for MPP Tracking of Photovoltaics. *Energies* **2023**, *16*, 2473. [CrossRef]
25. Pal, R.S.; Mukherjee, V. Metaheuristic based comparative MPPT methods for photovoltaic technology under partial shading condition. *Energy* **2020**, *212*, 118592. [CrossRef]
26. Mirza, A.F.; Mansoor, M.; Zhan, K.; Ling, Q. High-efficiency swarm intelligent maximum power point tracking control techniques for varying temperature and irradiance. *Energy* **2021**, *228*, 120602. [CrossRef]

27. Yan, Z.; Miyuan, Z.; Yajun, W.; Xibiao, C.; Yanjun, L. Photovoltaic MPPT algorithm based on adaptive particle swarm optimization neural-fuzzy control. *J. Intell. Fuzzy Syst.* **2023**, *44*, 341–351. [CrossRef]
28. Rezk, H.; Fathy, A.; Abdelaziz, A.Y. A comparison of different global MPPT techniques based on meta-heuristic algorithms for photovoltaic system subjected to partial shading conditions. *Renew. Sustain. Energy Rev.* **2017**, *74*, 377–386. [CrossRef]
29. Aguila-Leon, J.; Vargas-Salgado, C.; Chiñas-Palacios, C.; Díaz-Bello, D. Solar photovoltaic Maximum Power Point Tracking controller optimization using Grey Wolf Optimizer: A performance comparison between bio-inspired and traditional algorithms. *Expert Syst. Appl.* **2023**, *211*, 118700. [CrossRef]
30. González-Castaño, C.; Restrepo, C.; Kouro, S.; Rodriguez, J. MPPT algorithm based on artificial bee colony for PV system. *IEEE Access* **2021**, *9*, 43121–43133. [CrossRef]
31. Corrêa, H.P.; Vieira, F.H.T. Hybrid sensor-aided direct duty cycle control approach for maximum power point tracking in two-stage photovoltaic systems. *Int. J. Electr. Power Energy Syst.* **2023**, *145*, 108690. [CrossRef]
32. Dagal, I.; Akın, B.; Akboy, E. Improved salp swarm algorithm based on particle swarm optimization for maximum power point tracking of optimal photovoltaic systems. *Int. J. Energy Res.* **2022**, *46*, 8742–8759. [CrossRef]
33. Ahmed, M.; Harbi, I.; Kennel, R.; Rodriguez, J.; Abdelrahem, M. An improved photovoltaic maximum power point tracking technique-based model predictive control for fast atmospheric conditions. *Alex. Eng. J.* **2023**, *63*, 613–624. [CrossRef]
34. Ibrahim, M.H.; Ang, S.P.; Dani, M.N.; Rahman, M.I.; Petra, R.; Sulthan, S.M. Optimizing Step-Size of Perturb & Observe and Incremental Conductance MPPT Techniques Using PSO for Grid-Tied PV System. *IEEE Access* **2023**, *11*, 13079–13090.
35. Ibnelouad, A.; El Kari, A.; Ayad, H.; Mjahed, M. Improved cooperative artificial neural network-particle swarm optimization approach for solar photovoltaic systems using maximum power point tracking. *Int. Trans. Electr. Energy Syst.* **2020**, *30*, e12439. [CrossRef]
36. Kumar, D.; Chauhan, Y.K.; Pandey, A.S.; Srivastava, A.K.; Kumar, V.; Alsaif, F.; Elavarasan, R.M.; Islam, M.R.; Kannadasan, R.; Alsharif, M.H. A Novel Hybrid MPPT Approach for Solar PV Systems Using Particle-Swarm-Optimization-Trained Machine Learning and Flying Squirrel Search Optimization. *Sustainability* **2023**, *15*, 5575. [CrossRef]
37. Mohebbi, P.; Aazami, R.; Moradkhani, A.; Danyali, S. A Novel Intelligent Hybrid Algorithm for Maximum Power Point Tracking in PV System. *Int. J. Electron.* **2023**. [CrossRef]
38. Ngo, S.; Chiu, C.-S.; Ngo, T.-D.; Nguyen, C.-T. New Approach-based MPP Tracking Design for Standalone PV Energy Conversion Systems. *Elektron. Ir Elektrotechnika* **2023**, *29*, 49–58. [CrossRef]
39. Nancy Mary, J.; Mala, K. Optimized PV Fed Zeta Converter Integrated with MPPT Algorithm for Islanding Mode Operation. *Electr. Power Compon. Syst.* **2023**, *51*, 1240–1250. [CrossRef]
40. Al-Muthanna, G.; Fang, S.; AL-Wesabi, I.; Ameur, K.; Kotb, H.; AboRas, K.M.; Garni, H.Z.A.; Mas' ud, A.A. A High Speed MPPT Control Utilizing a Hybrid PSO-PID Controller under Partially Shaded Photovoltaic Battery Chargers. *Sustainability* **2023**, *15*, 3578. [CrossRef]
41. Nisha, M.; Nisha, M. Optimum Tuning of Photovoltaic System Via Hybrid Maximum Power Point Tracking Technique. *Intell. Autom. Soft Comput.* **2022**, *34*, 1399–1413. [CrossRef]
42. Gong, L.; Hou, G.; Huang, C. A two-stage MPPT controller for PV system based on the improved artificial bee colony and simultaneous heat transfer search algorithm. *ISA Trans.* **2023**, *132*, 428–443. [CrossRef] [PubMed]
43. Babes, B.; Boutaghane, A.; Hamouda, N. A novel nature-inspired maximum power point tracking (MPPT) controller based on ACO-ANN algorithm for photovoltaic (PV) system fed arc welding machines. *Neural Comput. Appl.* **2022**, *34*, 299–317. [CrossRef]
44. Avila, L.; De Paula, M.; Trimboli, M.; Carlucho, I. Deep reinforcement learning approach for MPPT control of partially shaded PV systems in Smart Grids. *Appl. Soft Comput.* **2020**, *97*, 106711. [CrossRef]
45. Saravanan, S.; Babu, N.R. RBFN based MPPT algorithm for PV system with high step up converter. *Energy Convers. Manag.* **2016**, *122*, 239–251. [CrossRef]
46. Sazli, M.H. A brief review of feed-forward neural networks. *Commun. Fac. Sci. Univ. Ank. Ser. A2–A3 Phys. Sci. Eng.* **2006**, *50*, 11–17. [CrossRef]
47. Razavi, S.; Tolson, B.A. A new formulation for feedforward neural networks. *IEEE Trans. Neural Netw.* **2011**, *22*, 1588–1598. [CrossRef] [PubMed]
48. Montana, D.J.; Davis, L. Training feedforward neural networks using genetic algorithms. In Proceedings of the 11th International Joint Conference on Artificial Intelligence, San Mateo, CA, USA, 20–25 August 1989; pp. 762–767.
49. Kaya, E. A comprehensive comparison of the performance of metaheuristic algorithms in neural network training for nonlinear system identification. *Mathematics* **2022**, *10*, 1611. [CrossRef]
50. Kaya, C.B.; Kaya, E.; Gokkus, G. Training Neuro-Fuzzy by Using Meta-Heuristic Algorithms for MPPT. *Comput. Syst. Sci. Eng.* **2023**, *45*, 69–84. [CrossRef]