electronics

Article

A Hyper-Parameter Optimizer Algorithm Based on Conditional Opposition Local-Based Learning Forbidden Redundant Indexes Adaptive Artificial Bee Colony Applied to Regularized Extreme Learning Machine

Philip Vasquez-Iglesias[1,†], Amelia E. Pizarro [2,†], David Zabala-Blanco [1,*], Juan Fuentes-Concha [2], Roberto Ahumada-Garcia [2], David Laroze [3] and Paulo Gonzalez [4]

[1] Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Avenida San Miguel 3605, Talca 3460000, Chile; fvasquez@ucm.cl
[2] Doctorado en Ingeniería, Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Avenida San Miguel 3605, Talca 3460000, Chile; ampizarro@ucm.cl (A.E.P.); juan.fuentes.01@alumnos.ucm.cl (J.F.-C.); rahumada@ucm.cl (R.A.-G.)
[3] Instituto de Alta Investigación, Universidad de Tarapacá, Casilla 7 D, Arica 1000000, Chile; dlarozen@uta.cl
[4] Facultad de Economía y Negocios, Universidad de Talca, Av. Lircay, Talca 3460000, Chile; paulo.gonzalezg@utalca.cl
* Correspondence: dzabala@ucm.cl
† These authors contributed equally to this work.

Abstract: Finding the best configuration of a neural network's hyper-parameters may take too long to be feasible using an exhaustive search, especially when the cardinality of the search space has a big combinatorial number of possible solutions with various hyper-parameters. This problem is aggravated when we also need to optimize the parameters of the neural network, such as the weight of the hidden neurons and biases. Extreme learning machines (ELMs) are part of the random weights neural network family, in which parameters are randomly initialized, and the solution, unlike gradient-descent-based algorithms, can be found analytically. This ability is especially useful for metaheuristic analysis due to its reduced training times allowing a faster optimization process, but the problem of finding the best hyper-parameter configuration is still remaining. In this paper, we propose a modification of the artificial bee colony (ABC) metaheuristic to act as parameterizers for a regularized ELM, incorporating three methods: an adaptive mechanism for ABC to balance exploration (global search) and exploitation (local search), an adaptation of the opposition-based learning technique called opposition local-based learning (OLBL) to strengthen exploitation, and a record of access to the search space called forbidden redundant indexes (FRI) that allow us to avoid redundant calculations and track the explored percentage of the search space. We set ten parameterizations applying different combinations of the proposed methods, limiting them to explore up to approximately 10% of the search space, with results over 98% compared to the maximum performance obtained in the exhaustive search in binary and multiclass datasets. The results demonstrate a promising use of these parameterizations to optimize the hyper-parameters of the R-ELM in datasets with different characteristics in cases where computational efficiency is required, with the possibility of extending its use to other problems with similar characteristics with minor modifications, such as the parameterization of support vector machines, digital image filters, and other neural networks, among others.

Keywords: artificial bee colony (ABC); metaheuristics; regularized extreme learning machine (R-ELM); hyper-parameter optimization (HPO); heuristic optimization; opposition-based learning (OBL); tabu search (TS); classification applications; artificial neural network (ANN)

Citation: Vasquez-Iglesias, P.; Pizarro, A.E.; Zabala-Blanco, D.; Fuentes-Concha, J.; Ahumada-Garcia, R.; Laroze, D.; Gonzalez, P. A Hyper-Parameter Optimizer Algorithm Based on Conditional Opposition Local-Based Learning Forbidden Redundant Indexes Adaptive Artificial Bee Colony Applied to Regularized Extreme Learning Machine. *Electronics* **2024**, *13*, 4652. https://doi.org/10.3390/electronics13234652

Academic Editor: Maciej Ławryńczuk

Received: 27 September 2024
Revised: 10 November 2024
Accepted: 19 November 2024
Published: 25 November 2024

1. Introduction

In neural networks, the training process usually implies the optimization of internal parameters, for instance, the weights, bias, or learning rate. This optimization process modifies those parameters to generate a model adapted to the data, with the aim to predict the behavior of unknown samples. However, models usually require a set of external parameters used in their initial configuration, fixed before training, that does not participate in the internal optimization process. Those external parameters are known as hyper-parameters. Some examples are the number of hidden networks in an artificial neural network model, *C* regularization parameter in logistic regression or support vector machine, or the number of considered neighbors in K-nearest neighbors.

Although the selection of an optimal set of hyper-parameters can directly affect the model performance, usually this process tends to be underestimated when approaching it with settings based on empirical knowledge, such as user defined grid search, random search, or even exhaustive search [1], which requires a lot of processing time (due to the need to train the model with all possible configurations of hyper-parameters to maximize metric performance) or gives us suboptimal results. Therefore, it is important to consider this process as a key component to build an effective machine learning model [1].

Random weight neural networks, such as extreme learning machines (ELM), simplify the training process by not requiring an iterative adjustment of all network weights. This allows for efficient and effective training [2], thereby facilitating a focus on hyper-parameter optimization. In this context, the regularized extreme learning machine (R-ELM) network is a variant of the original ELM model where a regularization coefficient is introduced. This coefficient is responsible for balancing between model complexity and the fit to the training data to improve model generalization performance. As an ELM model, R-ELM is a single hidden layer feed-forward network where the input weights and biases are initialized randomly. In this case, the training consists of searching for the output weights of the network, which can be calculated analytically. As a result, the training times are significantly reduced compared to traditional methods based on gradient descent.

However, as for any machine learning algorithm, the process of searching for optimal hyper-parameters is crucial for obtaining a robust and effective model. In a discretized search space, an exhaustive search ensures we obtain the optimal hyper-parameter configuration. However, it can involve a large number of unnecessary training runs that significantly increase the time associated with the hyper-parameter tuning process and, consequently, the overall experiment.

In this regard, the search for an optimal set of hyper-parameters can be seen as an optimization problem, called hyper-parameter optimization (HPO). Among the HPO methods, some are based on metaheuristics, which are general frameworks used to obtain approximate solutions for optimization problems that often require significant computational time to solve. These algorithms belong to the global random search algorithms, which have a set of typical limitations such as scalability problems in high-dimensionality problems and the lack of guarantee in finding the optimal value [3]. Even so, those approximate solutions are usually acceptable sub-optimal solutions, meaning they are not the optimal one but are still valid solutions. However, they are not necessarily close to the optimal solution, unlike near-optimal solutions. Metaheuristics apply different criteria to explore the search space, usually based on the behavior of living organisms to survive, resulting in the so-called bioinspired algorithms. An example is the swarm intelligence (SI) algorithm, which adopts the collective behavior of organized animal groups in search of survival [4]. In [5], SI algorithms are presented as particularly useful for non-deterministic polynomial-time hard problems where finding a global optimum becomes infeasible in a real-time scenario.

Several bioinspired algorithms can be found in the state of the art. Some examples are ant colony optimization (ACO) [6], derived from the behavior of ant colonies and their food-searching efficiency using pheromones; particle swarm optimization (PSO) [7], originated by the collective behavior of animal groups; Aquila optimizer (AO) [8], based on the four hunting methods used by Aquilas; pelican optimization algorithm (POA) [9],

inspired by how pelicans gather their food; and artificial bee colony (ABC) [10] algorithm, on which this work is based. This comes from the different roles performed by bees in a hive foraging for the best food source based on their waggle dance.

The use of these algorithms presents some advantages over classical optimization methods, such as the function to be optimized does not need to be differentiable. It is not necessary to know the geometry of the objective function in advance, and bioinspired algorithms tend to be more robust at optimizing multimodal functions by avoiding becoming stuck in local minima.

The concept of convergence is an important point within optimization algorithms. It indicates the process by which the algorithm approaches an optimal solution, depending on how some error measure or distance measure between the solution found and the expected one is reduced throughout iterations. Convergence can be classified by speed (sublinear, linear, superlinear, and quadratic) or according to the number of iterations required to reach the optimum (high convergence or low convergence) [11]. According to [11], SI algorithms are difficult to evaluate in terms of performance and convergence, so a graph of the best performance obtained throughout the iterations is usually presented. It should be noted that the convergence of these algorithms can be premature if the swarm stops exploring too soon and becomes stuck in local optima and can be global if the swarm converges to the optimal solution or close to it, which is usually the result of a balance between exploration (global search) and exploitation (local search). There are generic methods that help with the acceleration of convergence, such as opposition-based learning [12], which reinforces the exploration process by considering solutions opposite to the current candidates in order to improve the probability of finding better solutions, or tabu search [13], which establishes a tabu list to escape local optima.

In this paper, the ABC algorithm is formalized as parameterizer over an R-ELM network's hyper-parameters by incorporating three methods that adapt, strengthen, as well as diversify its search process and its exploration and exploitation mechanisms, which are common problems in global random search algorithms for large dimensions. The first incorporates an adaptive mechanism based on the number of iterations during the exploitation phases of ABC, which balances the search overall over the iterations. The second modifies the opposition-based learning technique to strengthen exploitation in promising areas and the latter keeps track of accesses to the search space elements in order to avoid redundant calculations and escape from local optima. The performance of these three methods was analyzed both independently and in combination, resulting in a total of ten parameterizers, which were applied to two datasets with class imbalance, by covering both binary and multiclass classification problems. At the same time, we propose to use a stopping criterion for the parameterizers based on a percentage of the cardinality of the discretized search space, comparing the results obtained with respect to an exhaustive search. This is used as a ground truth for the objective functions and as a contrast in execution times. For this, 100 complete executions (k-fold with k equal to 5, with 20 repetitions for each fold) of the exhaustive grid were averaged to reduce the effect of the stochastic behavior of the R-ELM on the metrics.

The rest of the paper has the following organization. Section 2 presents the related works found in the state of the art. Section 3 provides the background that supports the conducted research. Section 4 describes the applied methodology, namely, the adaptation of metaheuristics as parameterizers, each method proposed, the base ABC parameterizer, each method implementation to the ABC algorithm independently and in combination, the definition of performance metrics and objective functions, and the information related to the datasets used in this work. Section 5 details the results obtained during the experiments as well as presents the discussions related to the work. Finally, Section 6 presents the conclusions and future works.

2. Related Works

This section was created based on a literature review using the Web of Science database. The search was performed using the keywords: artificial bee colony, extreme learning machine, opposition-based learning, and tabu search. The inclusion criteria for the reviewed articles were based on the methodological relevance and practical applications of ABC-OBL, ABC-OBL-Tabu search, and ABC-Tabu search in the context of ELM. Studies that did not present significant advances or methodological innovations in the use of ABC and its variants were excluded.

Wang et al. [14] present a method called SADEABC-ELM, which combines ELM with optimizations using self-adaptive differential evolution (SADE) and ABC. Its goal is to improve the performance of blood concentration analysis via Raman spectroscopy. This approach focuses on optimizing the input and output weights, as well as the biases of ELM. By carrying this out, it improves the convergence of the algorithm and avoids sub-optimization.

The model introduced by Xu et al. [15] uses ABC to determine the best number of hidden neurons in the ELM based on the loss function, mean square error, and symmetric mean absolute percentage error. This is carried out to improve the prediction of annual GDP based on CO_2 emissions in the member countries of the Shanghai Cooperation Organization. The combined ELM-ABC model addresses issues related to parameter dependence and the limited prediction horizon of other traditional models, resulting in improved accuracy in long-term GDP prediction.

In [16], ABC is utilized to tune parameters and hyper-parameters of the ELM. The optimized parameters include connection weights between input and hidden layers, biases and activation functions in hidden neurons, and the regularization parameter. This method uses electricity price data from sources in Finland, Switzerland, and India. In combination with wavelet decomposition techniques, it improves forecasting capabilities in various electricity markets, particularly under high volatility conditions, leading to a substantial reduction in prediction errors.

In the study of [17], a hybrid model called ABC-DE-ELM is introduced. This model combines ABC with differential evolution to optimize the input weights and biases in an ELM, addressing the instability caused by random initialization in ELM. The proposed model improves the stability and accuracy in predictions for the context of traffic in intelligent transportation systems in big data environments.

In [18], a new version of ELM called TSE-ELM is exposed, which uses the ABC algorithm to optimize the input weights and biases of hidden neurons. It incorporates a hybrid strategy called GLABC to improve the algorithm's ability to explore the solution space. This approach provides more precise tuning of ELM parameters, improved generalization ability, and higher robustness to data variations. The approach uses the sigmoid function for activation of hidden neurons and dynamically adjusts the weights and biases during the optimization process using the Levy flight strategy.

In [19], the authors propose a hybrid model that includes an improved version of the GPS-EO-ABC algorithm, combining empirical wavelet transform (EWT), ARIMA, and an optimized ELM. The improved ABC algorithm, with good point sets (GPS) theory and elite opposition-based learning (EO) strategy, optimizes both global and local search capabilities, avoiding falling into suboptimal solutions. The hybrid model shows significant improvement in prediction accuracy compared to individual models and other model combinations. Furthermore, using EWT to decompose and denoise financial data significantly improves the prediction quality, and implementing EO increases the exploration capability of the solution space and helps avoid premature convergence. Other improvements include the use of GPS to generate a well-distributed initial population and adaptive inertia to automatically adjust the search step size during the iterative process.

In [20], a hybrid approach using ABC, ELM, ANFIS, and tabu search for early autism detection is presented. Tabu search was used for feature selection, ABC for gene selection, and ANFIS with ELM for classification. This integrated approach optimized the selection

of relevant features and improved the system's ability to model complex nonlinear relationships in the data. The combination of these techniques resulted in improved accuracy and speed in early autism detection.

In summary, most publications focus on creating new hybrid networks between the metaheuristic and the model used to optimize the weights and biases. As for the number of neurons, only [15] considers them in the optimization, while the rest of the publications use a fixed number or simple grid selected based on previous uses in the state of the art, arbitrarily or according to expert knowledge. Even so, no publication defines the selection range of the parameters or hyper-parameters based on a criterion. Similarly, the stopping criterion is not justified quantitatively based on some formula.

3. Background

3.1. Artificial Bee Colony

Artificial bee colony (ABC) [21] is a metaheuristic based on the behavior of honey bees and their dance communication system, which started as an idea in [10], by representing the food sources as solutions and their quality as the objective function. It works over iterations by randomly generating solutions in the first run. Dances represent the coordinates and quality of the food source. In order to perform global and local searches in the search space, metaheuristic algorithms use exploration and exploitation criteria, respectively. The ABC phases can be grouped as exploration phases (initialization and scout) and exploitation phases (employed and onlooker).

The ABC methodology begins with the generation of a population randomly distributed throughout the search space and of a size equal to the food source parameter. The approach utilizes three kind of bees that work in different phases:

$$x_i^j = x_{min}^j + \delta \cdot (x_{max}^j - x_{min}^j), \tag{1}$$

such that x_i^j corresponds to a new i-th food source in the j-th dimension; x_{min}^j and x_{max}^j correspond to the minimum and maximum assignable value of the j-th dimension, respectively; and δ is a random number in the range $[0, 1]$.

In the bee phase, each employed bee searches the periphery of its food source for a better location. Equation (2) is used to perform exploitation of the search space as a function of some other neighbor. It contains the impact parameter (ϕ), which is assigned a random value between -1 and 1 each time the equation is used.

$$v_{ij} = x_{ij} + \phi(x_{ij} - x_{kj}), \tag{2}$$

such that i corresponds to the i-th food source, j corresponds to the dimension selected to be modified, v_{ij} corresponds to the value of the i-th sample in the j-th dimension during the next iteration, x_{ij} corresponds to the value of the i-th sample in the current j-th dimension, ϕ is the random value previously explained, and x_{kj} corresponds to the value of the j-th dimension of a randomly selected k neighbor that must be different from i. Once an element of the periphery has been exploited, the employed bee returns to the hive to report on the quality and location of the food source based on the "waggle dance", whereas the onlooker bees evaluate the dances and select one of them to continue in the following phase.

In the onlooker bee phase, each $dance_j$ performed by the employed bees has a proportional probability of selection calculated by dividing its $fitness_j$ by the sum of the fitness, selecting one of them with the roulette wheel selection to exploit its periphery based on Equation (2).

Another form to calculate each probability is to normalize each $probability_j$ dividing $Fitness_j$ by the maximum fitness of the population [22,23]. Namely,

$$probability_j = a + b \cdot \frac{Fitness_j}{MAX(Fitness)}, \tag{3}$$

where a is a minimum probability of selection for all food sources; b is the impact factor of the normalized fitness with $1 = a + b$, so that the best food source has a selection probability of 100% independent of the values assigned to a and b; $Fitness_j$ represents the $Fitness$ of the j-th food source; and $MAX(Fitness)$ is the actual maximum $Fitness$ of all the population. For example, if the values of a and b are assigned as 0.1 and 0.9, respectively, all food sources have at least a 10% probability of selection independent from their fitness quality, and the remaining 90% are determined according to their performance.

The bees then observe the dances one by one based on the following procedure: bee_i compares the value of $probability_j$ with a random number r_0 between the range $[0, 1]$. If $r_0 \leq probability_j$, bee_i will use the j-th solution to exploit its periphery based on Equation (2). However, if $r_0 > probability_j$, another random value r_1 is generated, comparing it with $probability_{j+1}$. This process is repeated until there is a random number $r_n \leq probability_{j+n}$. Then, the following bee_{i+1} starts from $probability_{j+n+1}$. In the event that a bee does not select any of the remaining dances, this bee begins to observe again from $probability_1$.

Finally, the scout bee phase occurs whenever a food source is exhausted due to exploitation by employed or onlooker bees. The employed bee leaves the exhausted source and locates itself in a random position based on Equation (1). This phase contributes to the ABC exploration mechanism by reducing the cases where the algorithm becomes stuck in local optima and by allowing the discovery of new areas throughout the course of the algorithm's iterations.

As mentioned above, global random search algorithms have a set of typical limitations. In the case of ABC, its equations are often the targets of possible improvements to the algorithm [24] because these are its mechanisms of exploration and exploitation. In [25], the seminal ABC is described as a heuristic that converges slowly and struggles from the start to find the optimal value, which indicates problems in balancing the exploitation of promising areas and its ability to escape local optima. According to [26], the ABC algorithm has problems in optimizing multimodal functions by affecting its convergence rate. This suggests the need to implement modifications to the ABC algorithm or the incorporation of generic methods that improve its performance and make it robust against functions of unknown geometry, as occurs in parameterization. There are strategies that help improve the speed of algorithm convergence by enhancing or balancing exploration and exploitation mechanisms, such as methods with memory-based prohibitions and opposition-based learning.

3.2. Memory-Based Prohibitions

In general, evolutionary or swarm intelligence algorithms such as ABC have mechanisms that use previously generated solutions or information related to them, which can lead to repetition of these solutions, which is detrimental to exploration and to the computational efficiency when calculating the objective function impact on execution time. One of the first proposals to regulate this situation was [13], where it introduces the concept of tabu search. It comes to be a heuristic with a tabu list phase with a fixed size, where recent movements are recorded only for a certain number of iterations. The objective of this list is to avoid repeating solutions, aiming to escape local optima and encourage the exploration of new areas in the search space.

In other metaheuristics, concepts related to banning already explored locations were also applied. In [27], an adaptation of ABC is proposed to detect contours in grayscale images and applies a criterion called "Banned resource" to mark abandoned food sources and to use this information every time new solutions are generated by avoiding redundant computation. The way to decide if a resource is banned is to allow five executions of the same coordinate configuration, counted only in the scout bee phase, and then proceed to ban it. In this case, the objective function is the value of a pixel in the image and is solved with a direct query in memory, so the wasted computation is associated with the reiteration of the use of previous information.

In [28], a set of restrictions to the PSO algorithm is proposed. There is one restriction called "Checking for already visited locations", where a list is created by adding to it each visited position in the search space. This list is described as a permanent marking of the locations already visited, equivalent to a long-term memory mechanism. In the event that a new solution has already been visited, it will be replaced by a random adjacent neighbor that is not determined to be already visited. One aspect to consider is that each element added to it increases the query time. In conclusion, these kinds of proposals can reduce the amount of redundant computation when the objective function is evaluated with input value combinations that have already been previously tested, implying stagnation in convergence and wasted computation time.

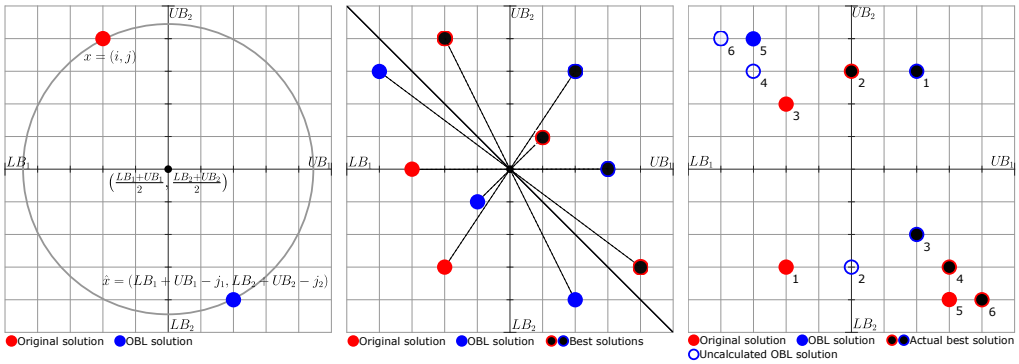
3.3. Opposition-Based Learning

Opposition-based learning (OBL) [12] is a strategy in which a solution found and its opposite solution compete to determine which one has a better performance. The opposite solution is one whose coordinates are reflected within the search space concerning the original solution, obtained according to Equation (4).

$$\hat{x_j} = LB_j + UB_j - x_j, \tag{4}$$

where $\hat{x_j}$ corresponds to the spatial opposite of x for the j-th dimension, UB_j corresponds to the maximum value or upper bound of the j-th dimension, and x_j corresponds to the value of x in the j-th dimension. The distance between x and UB is equal to the distance from LB to $\hat{x}$, namely, $\hat{x}$ is the reflection of x using the center of the search space $\frac{(LB+UB)}{2}$ as the reference point. In the case of two dimensions, if we have a point $x = (j_1, j_2)$, its spatial opposite is given by $\hat{x} = (LB_1 + UB_1 - j_1, LB_2 + UB_2 - j_2)$, which implies a reflection with respect to the central point of the search space $\left(\frac{LB_1+UB_1}{2}, \frac{LB_2+UB_2}{2}\right)$, see Figure 1a.

OBL can also be applied to a set of solutions where the performance of each solution of the original set and the opposite set is evaluated by keeping only the best half of all the solutions. In Figure 1b, an example of applying OBL to a set of six solutions in a 2D space is observed, where the red dots are the original solutions, the blue dots are the opposite solutions, and the dots with black centers are the selected solutions, by assuming that the best solutions are located over the main diagonal.



(a) OBL applied in a two dimensional search space

(b) OBL applied to a set of solutions

(c) Generation of solutions over iterations using COBL.

Figure 1. Different forms of OBL. The numbers next to the dots represent the iteration where a solution was generated, and the unfilled dots represent uncalculated OBL solutions in COBL.

OBL can be considered an auxiliary exploration mechanism because it uses the center of the search space as a reference. Consequently, the generated opposite solutions do not take into consideration whether the original solution is based on exploitation or exploration mechanisms. On the other hand, applying OBL to all solutions doubles the total number of solutions generated and the execution time. These problems can be partially reduced

by applying a conditional process in the application of OBL [26]. The way to apply it is to evaluate whether the generated solution is better or not than the current one, and only if it does not occur, the opposite solution is calculated, refer to Figure 1c. Because the authors do not give it a distinctive name, we call it conditional OBL (COBL) to refer to its conditional operation and differentiate it from the classic OBL in the rest of the paper.

Regarding neural networks, OBL, as well as ABC, offer an excellent alternative to tackle the optimization process compared to exhaustive search methods. For instance, unlike the ELM network, the regularized variant includes an additional parameter to be optimized related to the neural network's regularization. That difference can substantially impact the duration of the optimization process, especially when we consider that the search space grows exponentially when incorporating more hyper-parameters, combined with a large search space. In this case, if we observe the incorporation of the regularization parameter C and the large search space for the number of neurons in the hidden layer, where the step size is equal to 1 and the upper limit as the number of samples of each dataset, this problem offers the ideal conditions for the use of techniques such as memory-based prohibitions, OBL, and ABC. Those techniques could significantly reduce optimization process times with a minor impact on accuracy performance.

3.4. Regularized Extreme Learning Machine

The extreme learning machine (ELM) is a neural network marked by the use of a single hidden layer with randomly initialized input weights and biases, removing the need to adjust the weights of the hidden layer model iteratively [29,30]. As a result, the training times are significantly reduced compared to gradient descent approaches that usually have problems associated with local minima convergences or incorrect learning steps. It is part of the random weights neural network family [31], as radial basis function [32], random vector functional link [33], and feed-forward neural networks with random weights presented in [34], all of these characterized by fast training times and good generalization performance. Figure 2 provides a visual representation of the ELM algorithm, where $\widetilde{N}$ is the number of neurons in the hidden layer; $\mathbf{X}$ is the input data of n features and N samples; $\mathbf{T}$ is the expected output with m classes; $\mathbf{W}$ and b correspond to the input weights and the hidden layer bias, respectively, both of which are randomly generated; $\mathbf{H}(\mathbf{W}, \mathbf{X}, b)$ is the hidden layer output of size $N \times \widetilde{N}$; and $\boldsymbol{\beta}$ is the output weight.

The ELM training consists of the search for the least squares solution $\hat{\boldsymbol{\beta}}$ of the linear system given by the following equation:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}. \tag{5}$$

In the case when H is a square matrix, the solution is unique, and it is given by $\boldsymbol{\beta} = \mathbf{H}^{-1}\mathbf{T}$. However, this case rarely appears in practice because, usually, the number of training samples N does not necessarily have to match with the number of neurons in the hidden layer $\widetilde{N}$. According to [29], in the case when $\mathbf{H}$ is not a square matrix, one solution is given by $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore–Penrose pseudoinverse. This solution is optimal because it minimizes the network's error and norm weights [29]. This matrix is usually computed via orthogonal projection, orthogonalization, iterative, or singular value decomposition (SVD) methods [29]. In this work, the orthogonal projection computation $H^{\dagger} = (H^T H)^{-1} H^T$ is used due to efficient time performance [30].

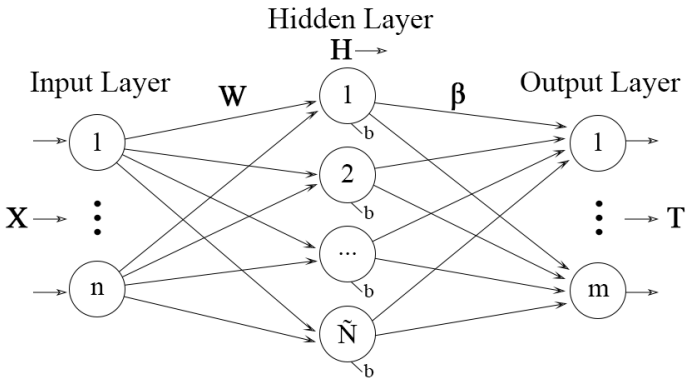


Figure 2. Representation of the ELM model.

According to [35], although the original ELM is fast in the learning phase and shows a good generalization performance, it has some drawbacks that should be addressed. In a prior instance, the real data distribution is unknown due to the ELM's training dataset representing only a subset of the whole population. This empirical risk minimization principle can lead to over-fitting models to the training data, reducing their ability to generalize over unseen samples. Furthermore, the training approach of ELM directly calculates the minimum norm least-squares solution that results in a loss of control over how ELM is adjusted to the training data. To overcome these drawbacks, a balance between the model complexity and the fit on training data is needed. The regularized extreme learning machine (R-ELM) [35] is based on the structural risk minimization principle. In order to achieve better generalization performance, it is essential to find the optimal balance between empirical risk (the sum of error squares, denoted as $\|\boldsymbol{\varepsilon}\|^2$) and structural risk (the sum of weight squares, denoted as $\|\boldsymbol{\beta}\|^2$). These factors are essential in determining the actual prediction risk in the cost function [35]. It can be represented in the following expression:

$$\min\left(\frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2}C\|\mathbf{D}\boldsymbol{\varepsilon}\|^2\right), \tag{6}$$

where C is the trade-off coefficient, and $\mathbf{D} = \text{diag}(\mathbf{v})$ with $\mathbf{v} = (v_1, v_2, \ldots, v_N)$ represents the weight factor vector of errors $\boldsymbol{\varepsilon}$. If $\mathbf{D} = \mathbf{I}$ is the unit matrix, $\boldsymbol{\beta}$ can be calculated as follows:

$$\boldsymbol{\beta} = \left(\frac{I}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{7}$$

Equation (7) is known as the unweighted regularized ELM. Note that this expression is valid in the case where the number of samples is greater than the number of hidden neurons. There are some variants of the ELM that are used under specific classification problems, such as weighted [36] or semi-supervised ELMs [37]. In this case, we select the regularized ELM since it is a simple network with sufficient good performance to test heuristic-based algorithms for hyper-parameter tuning as the employed in this work.

Although the R-ELM model provides a better balance between model complexity and fit to the training data compared to traditional ELM networks, the introduction of an additional parameter, which may not seem significant at first glance, can exponentially increase the hyper-parameter optimization process time. This process, both in ELM and in neural networks in general, can require significant computational resources when using exhaustive search algorithms. In this context, the use of a more efficient optimization algorithm that can accelerate convergence to an optimal set of hyper-parameters is needed, saving time, energy, and resource availability. In other words, we notice that the R-ELM has two hyper-parameters (the number of hidden neurons and the regularization parameter). These must be optimized in order to maximize the performance without computational cost. This task is performed by exhaustive search in most researches or fixed arbitrarily.

Nevertheless, as the number of samples increases, which is the last goal in the artificial intelligence context for successful learning, the method based on brute force (exhaustive search) is unfeasible, see Section 5.2. Solutions based on swarm intelligence (AB metaheuristics and its improvements, previous sections) become relevant here.

4. Methodology

The main objective is to propose a parameterizer based on the ABC algorithm, which is capable of obtaining a sub-optimal configuration of the hyper-parameters of an R-ELM with a low error margin compared to the optimal value obtained in an exhaustive search.

To this end, we define and propose a set of methods that seek to improve the convergence rate of obtaining results in the parameterizers. In Section 4.1, we establish the necessary settings to implement a metaheuristic optimization algorithm as a parameterizer. In Section 4.2, the proposed methods adaptive phi ABC, forbidden redundant indexes, and opposition local-based learning are carefully presented. In Section 4.3, the base ABC, OBL ABC, as well as three parameterizers that implement one of the proposed methods, are presented. In Section 4.4, the ABC parameterizers that implement two or more of the proposals are illustrated. Finally, in Section 4.5, the objective functions and performance metrics used during this research are revealed.

4.1. Adaptation of Metaheuristics as Parameterizers

To carry out the parameterization from a metaheuristic optimization algorithm, it is necessary to determine the following components according to the problem to be solved: objective function to be optimized, dimensionality of the problem, restrictions, and cardinality of the search space.

The objective function to be optimized depends on the algorithm to be parameterized. It has the dimensions of the problem as input values and some metric as the output value to objectively measure the performance. In this work, it is applied to the (R-ELM) network and the performance metrics accuracy and g-mean.

The dimensionality of the problem refers to the number of input variables in the objective function of the problem to be parameterized. In the case of this work, each dimension corresponds to a hyper-parameter in the R-ELM, which are the number of hidden neurons and the regularization parameter C, together representing the search space.

The constraints are given by the different domains of each of the dimensions. In general, four constraints can be identified for each dimension: the type of domain to which the elements belong, their lower limit, their upper limit, and the step size between elements of discretized domains. The search space refers to the set of all possible valid solutions to be evaluated. Discretizing the search space allows for standardizing access to the elements of each dimension. This is achieved by assigning the values incrementally in an array, which implies defining a standardized step size for the elements of each domain. In the case of an R-ELM, for the regularization parameter C, a restriction is established in the domain in the range of $2^{[-25,25]}$ with a step size of 1 in the exponent and for the number of neurons corresponding to the interval $[1, MaxNeu]$, with a step size of 1, where $MaxNeu$ is the rounded value of calculating 80% of the number of samples in the dataset worked.

The cardinality of the search space is given by all possible combinations of the valid elements of each dimension. It is determined by the production of the cardinal of each dimension, as presented in the following:

$$\#\Omega = \prod_{i=1}^{N} \#axis(i), \tag{8}$$

where $\#axis(i)$ denotes the i-axis cardinality. Here, the cardinality $\#\Omega$ for the parameterization of the R-ELM is given by expression (9):

$$\#\Omega_{R-ELM} = size(C) \cdot size(NeuralNumber), \tag{9}$$

where $\#\Omega_{R-ELM}$ represents the number of different solutions when the R-ELM is parameterized in the classification of a dataset, $size(C)$ is the number of elements in the dimension generated by parameter C, and $size(NeuralNumber)$ denotes the number of elements in the dimension generated by the parameter $NeuralNumber$.

Finally, the stopping criterion is given by a number of executions of the objective function equivalent to a percentage of the cardinality, defined in Equation (10),

$$Totaltravels : T = \left\lceil \frac{P \cdot \#\Omega_{R-ELM}}{N} \right\rceil, \tag{10}$$

where P represents a percentage of $\#\Omega_{R-ELM}$ and N the number of agents in the population. This implies that the total number of allowed executions of the objective function is divided equally among the N individuals in the population. Using the percentage defined, the relationship between the proximity of the best solution found to the optimal value and the amount of time spent searching for it can be decided, which is essential in search spaces with high cardinality.

4.2. Proposed Methods

4.2.1. Adaptive Phi ABC

Adaptive phi ABC (AP-ABC) is a modification of the ABC algorithm. The objective of this method is to generate a balance between exploration and exploitation through the execution of the algorithm, so that all solutions created in each iteration are affected globally and equivalently. Part of the inspiration is based on [38], where a dynamic strategy based on the number of iterations executed is proposed for the PSO algorithm to dynamically change from exploitation to exploration.

In the case of our approach, it started favoring exploration, changing to exploitation and changing to exploration again due to endowing the parameter ϕ seen in Equation (2) with an adaptive feature. It is based on the percentage of the number of iterations elapsed during the execution using a monotonically descending lineal function $y = mx + b$, with $y = \phi_{it}$, $x = it$, $b = 1$ and $m = -\frac{2}{T}$, according to Equation (11).

$$\phi_{it} = \begin{cases} 1 - \frac{2 \cdot it}{T}, & \text{if } it < \frac{T}{2}, \\ \frac{2 \cdot it}{T} - 1, & \text{otherwise .} \end{cases} \tag{11}$$

thus, ϕ_{it} is a vector of size T that has mapped values in the range [0,1] starting in 1 when $it = 1$, decreasing to 0 when $it = \frac{T}{2}$, and increasing again to 1 when $it = T$. In order to represent the possibility of moving towards or away from the previous solution based on the selected neighbor, Equation (2) is replaced by Equation (12).

$$v_{ij} = x_{ij} + \alpha \cdot \phi_{it}(x_{ij} - x_{ik}), \tag{12}$$

where α is a random number belonging to the set $\{-1, 1\}$. For a better understanding, the behavior of ϕ_{it} is seen in Figure 3. On average, ϕ_{it} is equal to 0.5, which is distributed in such a way that in the first 25%, exploration is heavily encouraged, then further exploitation is forced to occur in the middle 50% of runs, and finally, exploration is reinforced again in the final 25%.

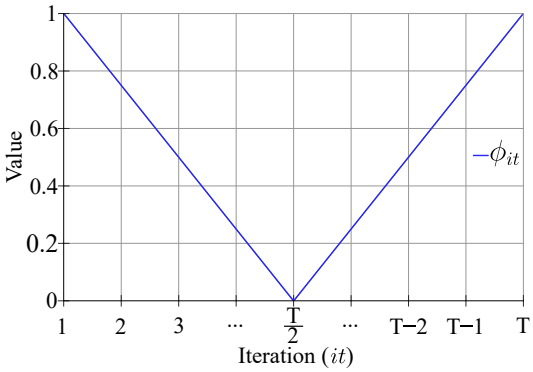


Figure 3. Behavior of ϕ_{it} in the AP-ABC approach.

4.2.2. Forbidden Redundant Indexes

The forbidden redundant indexes (FRI) proposal aims to avoid possible redundant calculations performed during the optimization process by marking all the positions already explored in a multidimensional array of the same size and dimension as the total search space. These forbidden redundant indexes allow for checking its availability each time a new solution is created. The multidimensional array starts with "False" values. During the algorithm, each time a solution is created, the combination of its indexes is checked in the array. This improves computational efficiency by ensuring that each generated solution is unique, avoiding wasting time executing the objective function on previously explored configurations, effectively promoting exploration.

This criterion becomes stronger as the search space becomes more discrete, since it can guarantee that each new solution generated is unique, enabling measurement of the traveled percentage of the search space throughout the iterations of the algorithm. Similarly, it is a mechanism that ensures escape from local optima by forcing the exploration of new places when all its nearby area has already been marked. All this can be seen in Figure 4, a 1D example where it can be seen how the search space is marked throughout the iterations. By a single travel for each iteration, this method establishes the cardinality of the problem as the maximum possible value for the number of iterations of the parameterizers, because at that moment, the entire search space will have already been explored.

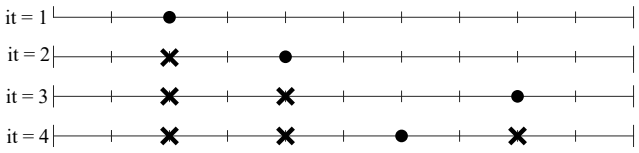


Figure 4. Example of forbidden redundant indexes applied to a 1D search space. The dots represent an available index in the search space, while the Xs represent forbidden indexes.

The detailed operation of FRI is presented in the Algorithm 1. For its implementation, a binary array with a cardinality equal to that of the search space is initialized with zero values. Then, each time the coordinates are assigned to calculate a solution, the array is checked to see if it has already been marked. If not, it is marked with a 1 and a TRUE is returned, which means that it is an unexplored configuration, proceeding to perform the calculation of the objective function only in this case. On the contrary, if, at the time of the query, it was already marked with a value of 1, a FALSE is returned, preventing computation from being wasted in an already explored configuration.

In conclusion, for the application of this method in the hyper-optimization of the R-ELM, it is guaranteed that each training performed during the search for the optimal configuration is unique and the method remains constantly exploring the space of options.

Algorithm 1 Forbidden Redundant Indexes

```
Inputs: Coordinates, FRI_Array
Output: TRUE or FALSE
valid ← FRI_Array[Coordinates]
if valid == 0 then
    FRI_Array[Coordinates] ← 1
    Return TRUE
else
    Return FALSE
end if
```

4.2.3. Opposition Local-Based Learning

Opposition local-based learning (OLBL) is a modification of OBL focused on enhancing exploitation. In standard OBL, to generate the opposite solution $\hat{x}_j$, where j represents the j-th dimension, each coordinate of the candidate solution x_j is reflected with respect to the center of the search space of each dimension according to Equation (4). The problem with the classical form of OBL is that it does not take into account whether the candidate solution was generated based on exploration or exploitation mechanisms. This goes against the exploitation mechanisms if the opposite solution is generated in an area far from the current one, increasing the possibility of wasting computation in unpromising areas.

In the case of OLBL, the current solution i is used as a reference center to reflect the candidate solution $i+1$ and generate an opposite local solution based on Equation (13), namely,

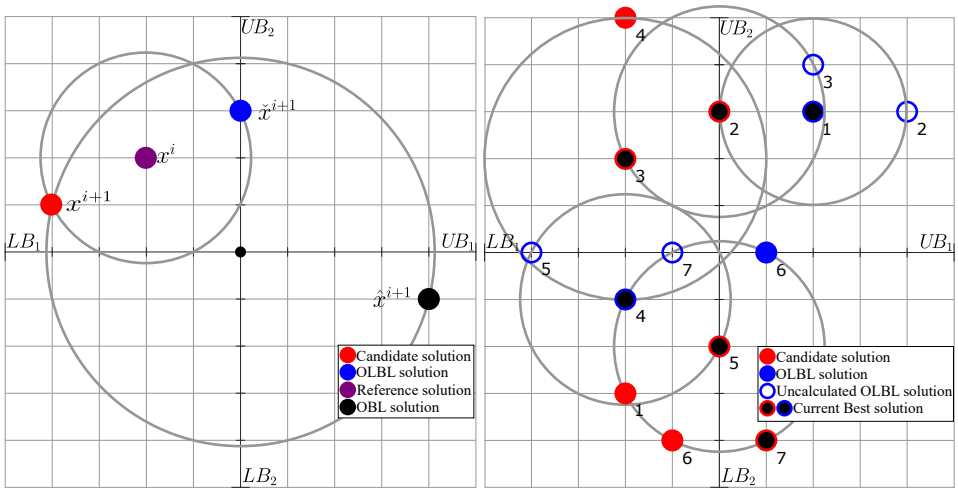
$$\check{x}_j^{i+1} = 2 \cdot x_j^{i+1} - \gamma_j \cdot x_j^i, \tag{13}$$

where $\check{x}_j^{i+1}$ represents the opposite local solution, γ_j is a scaling parameter, x_j^i represents the current solution, and x_j^{i+1} represents the candidate solution. This method continues to use the concept of opposite-based learning but effectively enhances the exploitation mechanisms by focusing the computation on the areas surrounding the reference solution. In this way, OBL and OLBL complement each other, in the sense that OBL reinforces the exploration mechanisms and OLBL reinforces the exploitation mechanisms.

It is worth noting that it is possible to generate solutions that go outside the edges. One way to correct this is to relocate all the coordinates that go outside their allowed limits to the nearest edge. To define the interval of values of γ_j that allow generating a valid solution within the search space, it must be considered that $LB < 2 \cdot x_j^{i+1} - \gamma_j \cdot x_j^i < UB$, by producing

$$\frac{2 \cdot x_j^{i+1} - UB}{x_j^i} < \gamma_j < \frac{2 \cdot x_j^{i+1} - LB}{x_j^i}. \tag{14}$$

As with OBL, this approach can be applied directly (OLBL) or conditionally (COLBL), which allows for reducing unnecessary computation. A bi-dimensional example of OLBL is shown in Figure 5a,b, which shows an example of the evolution process of a solution using conditional OLBL (COLBL).



(**a**) OLBL applied to a two dimensional search space

(**b**) Evolution process of a solution using COLBL

Figure 5. Examples of OLBL. Each reference solution is the center of each gray circle, which, in its radius, has each candidate solution represented with a red dot, and each opposite local solution is identified with a blue dot. In the right figure, the numbers next to the dots represent the iteration where a solution was generated.

It should be noted that OLBL is different from [39], where they use the centroid of the population as a reference, or [40], where the best solution of the population is used as a reference center. In the case of OLBL, the current solution of each agent is used independently, such that their opposite local solutions are their own. This implies that the best solution of each agent is not used due to depending on the metaheuristic; there may be some mechanism of diversification of solutions to avoid stagnating in local optima, and it could replace the best solution of an agent with a worst one, even if it is the best value found so far.

The operation of COLBL is carefully presented below in Algorithm 2, which allows it to be implemented as a stand-alone function whose input parameters are x^i, x^{i+1}, γ, LB, and UB, while the output is the solution with the best fitness between x^i, x^{i+1}, and $\check{x}^{i+1}$, chosen according to the algorithm's criteria. The algorithm starts by checking whether $fitness(x^{i+1}) > fitness(x^i)$. If TRUE, x^{i+1} is returned. Otherwise, the values of γ are assigned, the opposite local solution $\check{x}^{i+1}$ is obtained, and an empty list is initialized, to which all the coordinates of $\check{x}^{i+1}$ that fall outside the allowed limits are added. If all of them are within the limits, it is checked whether $\check{x}^{i+1} > fitness(x^i)$, returning $\check{x}^{i+1}$ if true or $fitness(x^i)$ if false. On the other hand, if at least one coordinate of $\check{x}^{i+1}$ is invalid, it is corrected to its nearest limit and the query is made for $\check{x}^{i+1} > fitness(x^i)$, returning $\check{x}^{i+1}$ if true or x^i if FALSE.

Algorithm 2 Conditional Opposition Local-based Learning

Inputs: x^i, x^{i+1}, γ, LB and UB
Output: Best solution between x^i, x^{i+1} and $\breve{x}^{i+1}$
if $fitness(x^{i+1}) > fitness(x^i)$ **then**
  Return x^{i+1}
else
  for $k \leftarrow 1$ **to** j **do**
    $\gamma_k \leftarrow 1$
  end for
  Get $\breve{x}^{i+1}$ with Equation (13)
  valid $\leftarrow TRUE$
  invalid_dimensions $\leftarrow$ **empty list**
  for $k \leftarrow 1$ **to** j **do**
    if $\breve{x}_k^{i+1} < LB_k$ **or** $\breve{x}_k^{i+1} > UB_k$ **then**
      valid $\leftarrow FALSE$
      add k to **invalid_dimensions**
    end if
  end for
  if valid then
    if $fitness(\breve{x}^{i+1}) > fitness(x^i)$ **then**
      Return $\breve{x}^{i+1}$
    else
      Return x^i
    end if
  else
    for each k in **invalid_dimensions do**
      if $\breve{x}_k^{i+1} < LB_k$ **then**
        $\breve{x}_k^{i+1} \leftarrow LB_k$
      else
        $\breve{x}_k^{i+1} \leftarrow UB_k$
      end if
    end for
    if $fitness(\breve{x}^{i+1}) > fitness(x^i)$ **then**
      Return $\breve{x}^{i+1}$
    else
      Return x^i
    end if
  end if
end if

4.3. Baseline parameterizer Algorithms

4.3.1. Base ABC parameterizer

The parameterizer based on the base ABC does not have changes in the algorithm's structure itself, and the majority of its modifications are focused on the necessary adjustments to parameterize other algorithms, as described in Section 4.1. Furthermore, Equation (2) does not consider the limits of the search space in generating solutions. Hence, it is possible to generate a candidate solution v_{ij} that is outside the valid range. In this situation, the out of range coordinate is corrected to LB_j if $distance(v_{ij}, LB_j) < distance(v_{ij}, UB_j)$ or UB_j otherwise. In Figure 6, a diagram that includes the structure of the parameterizer based on the base ABC is presented step by step.

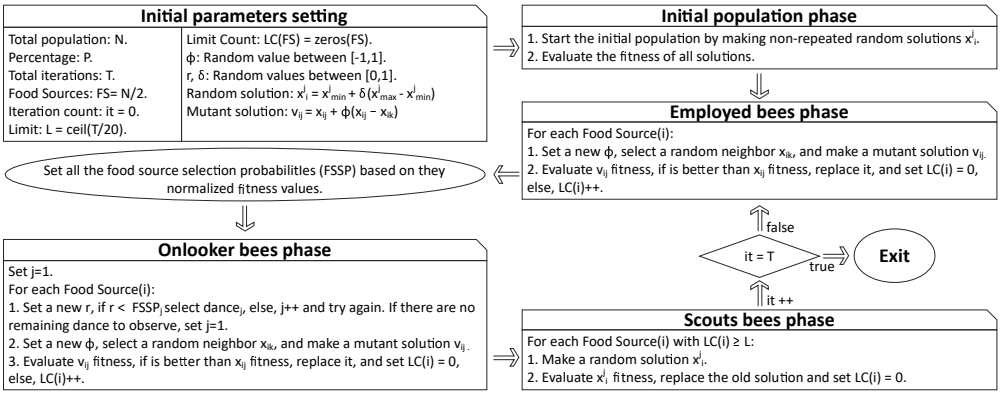


Figure 6. Algorithm diagram of the Base ABC Parameterizer approach.

4.3.2. Opposition-Based Learning ABC Parameterizer

Opposition-based learning ABC parameterizer (OBL ABC) is inspired by the base ABC parameterizer with the inclusion of OBL. In the initialization phase, OBL is applied to all the sets of solutions, selecting the best 50% of them, as shown in Figure 1b. In the employed and onlooker bee phases, OBL is applied in each generated solution, selecting the best one between the reference solution, candidate solution, and OBL solution. Finally, in the scout bee phase, the selected solution is the best one between the candidate solution and its OBL solution. The diagram of this approach is seen in Figure 7.

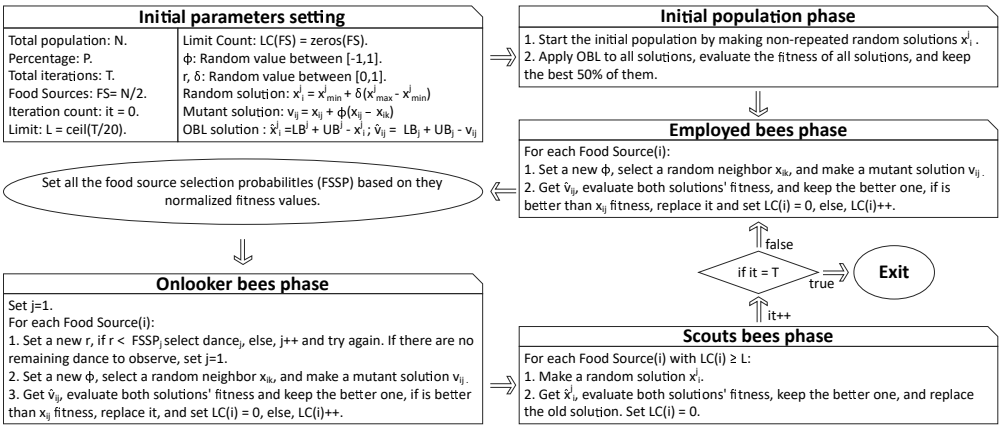


Figure 7. Algorithm diagram of the OBL ABC parameterizer approach.

4.3.3. Adaptive Phi ABC Parameterizer

Adaptive phi ABC parameterizer (AP-ABC) is the direct implementation of the one proposed in Section 4.2.1 to the base ABC parameterizer. It acts exactly as the base ABC parameterizer but using Equations (11) and (12) instead of Equation (2). For a better understanding, the diagram of this approach is seen in Figure 8.

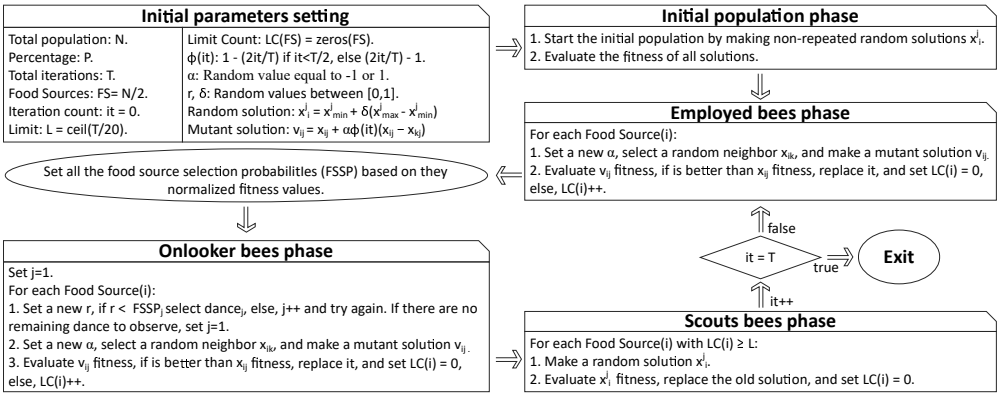


Figure 8. Algorithm diagram of the AP-ABC parameterizer approach.

4.3.4. Forbidden Redundant Indexes ABC Parameterizer

The forbidden redundant indexes ABC parameterizer (FABC) is the inclusion of the FRI method on the Base ABC. Its operation is based on the principles presented in Section 4.2.2. Each time a candidate hyper-parameter configuration is generated, Algorithm 1 is called. If it returns FALSE, another hyper-parameter configuration is generated. This process is iterative until it returns TRUE, at which point the hyper-parameter configuration is marked as indicated by the algorithm and the objective function is calculated.

To avoid wasting time consecutively consulting highly explored sectors, the following considerations should be taken into account during the search space exploitation phases (employed and onlooker phases). Each time a new mutant solution is generated from Equation (2), its status is checked from the array of forbidden positions. If it is available, the evaluation of the new solution continues, and it is marked as visited. If it is not available, Equation (2) is executed up to five times, modifying only the random component of the formula. If it is not effective in finding an available solution, it is executed up to another five times, also modifying the dimension and the chosen neighbor with new random values. If it is also not effective, it implies that a large part of the periphery of the explored area is already forbidden; so, using Equation (1), the bee becomes an explorer, locating itself in a random position that is not forbidden. The diagram of this approach is seen in Figure 9.

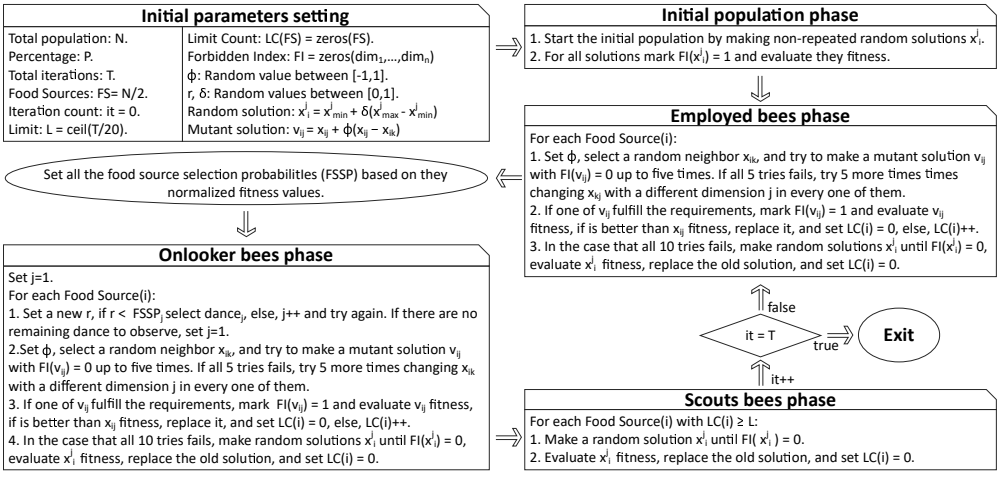


Figure 9. Algorithm diagram of the FABC parameterizer approach.

4.3.5. Conditional Opposition Local-Based Learning ABC Parameterizer

The conditional opposition local-based learning ABC parameterizer (COLBL ABC) is inspired by the base ABC parameterizer with the inclusion of COLBL presented in Section 4.2.3. In the initialization phase and scout bee phase, (Exploration) acts as the OBL ABC. In the case of the employed bee phase and onlooker bee phase (exploitation), COLBL is applied as shown in Algorithm 2. The diagram of this approach is seen in Figure 10.

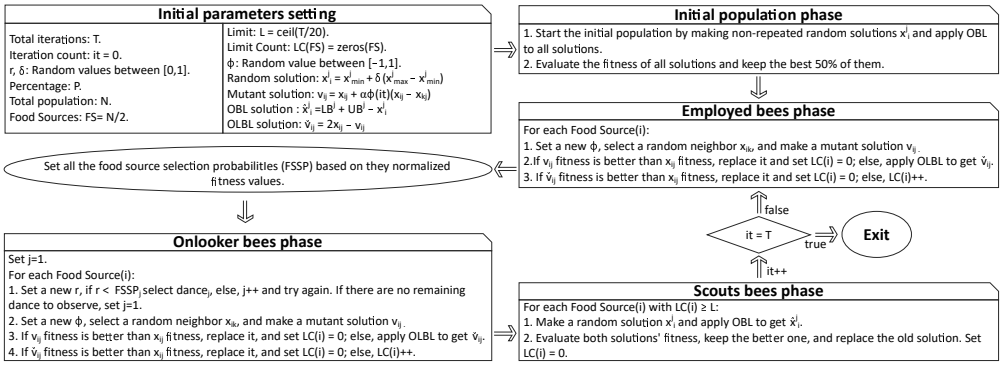


Figure 10. Algorithm diagram of the COLBL ABC parameterizer approach.

4.4. Mixed Parameterizer Algorithms

As mentioned, the ABC parameterizers that implement two or more of the proposals are illustrated by exploiting the information given in Section 4.3.

4.4.1. Opposition-Based Learning Forbidden Redundant Indexes ABC Parameterizer

The opposition-based learning forbidden redundant indexes ABC parameterizer (OBL FABC) integrates OBL, FRI, and ABC, considering that every time a new solution and its opposite are generated, both are marked in the forbidden index, which implies that if a generated solution is marked as visited, its opposite is also identified, being necessary to consult only for the generated solution (not for its opposite). This parameterizer is created to observe the effect of FRI on the OBL ABC parameterizer.

4.4.2. Adaptive Phi Forbidden Redundant Indexes ABC Parameterizer

The adaptive phi forbidden redundant indexes ABC parameterizer (AP-FABC) is the implementation of FRI and AP at the same time. It acts as the FABC parameterizer, but with the inclusion of the dynamic mechanism of ϕ_{it} following Equation (11).

4.4.3. Conditional Opposition Local-Based Learning Forbidden Redundant Indexes ABC Parameterizer

The conditional opposition local-based learning forbidden redundant indexes ABC parameterizer (COLBL FABC) integrates COLBL, FRI, and ABC. The following consideration should be taken into account because OLBL is applied conditionally, affecting the implementation of the proposed method in Section 4.2.2. During the scout bee phase, once the new unvisited random solution has been created using Equation (1), a check is made to see if the spatial inverse with respect to the axes is available. On the one hand, if the opposite has already been visited previously, its fitness is not calculated and the created random solution directly replaces the solution where the resources were exhausted. On the other hand, if the opposite has not been visited before, its fitness is also calculated. The new solution will be the one with the highest fitness between the newly created random and its spatial opposite.

4.4.4. Conditional Opposition Local-Based Learning Adaptive Phi ABC Parameterizer

The conditional opposition local-based learning adaptive phi ABC parameterizer (COLBL AP-ABC) is the application of the COLBL and AP methods simultaneously. In general, its operation is the same as the COLBL ABC parameterizer, but with the inclusion of the dynamic mechanism of ϕ_{it} following Equation (11).

4.4.5. Conditional Opposition Local-Based Learning Adaptive Phi Forbidden Redundant Indexes ABC Parameterizer

The conditional opposition local-based learning adaptive phi forbidden redundant indexes ABC parameterizer (COLBL AP-FABC) is equivalent to applying all the methods proposed in Section 4.2 to the base ABC parameterizer. Its operation is similar to that of

COLBL FABC, described in Section 4.4.3, but with the inclusion of the dynamic mechanism of ϕ_{it} following Equation (11).

To conclude, the diagram of the COLBL AP-FABC parameterizer approach is seen in Figure 11.

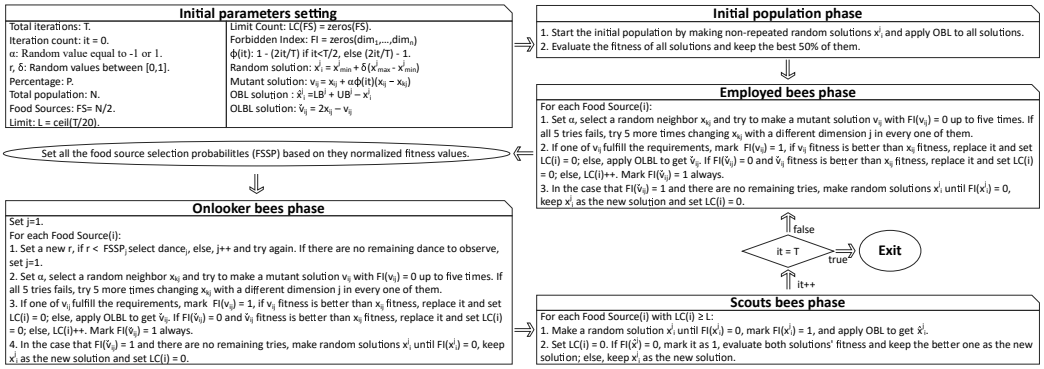


Figure 11. Algorithm diagram of the COLBL AP-FABC parameterizer approach.

4.5. Objective Functions and Performance Metrics

The proposed approaches require an objective function that represents the performance of the R-ELM. In this case, we will maximize the metrics accuracy and g-mean. Accuracy represents the percentage of correct classifications in general, and g-mean allows the model to be evaluated more robustly when there is an imbalance in the dataset.

The expression Equation (15) represents the accuracy, where N is the number of samples, t_i denotes the real class of the sample i, and $\bar{t}_i$ is the predicted class of the sample i [41,42].

$$\text{Accuracy} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - \bar{t}_i)^2}, \tag{15}$$

Regarding the g-mean, it is calculated according to what is established in Equation (16), where L corresponds to the number of classes, and β_i to the accuracy of class i [41,42].

$$\text{G-Mean} = \sqrt[L]{\beta_1\beta_2...\beta_L}, \tag{16}$$

The results will be evaluated in terms of the maximum value achieved in the metrics, and the percentage of proximity and of each approach to the best value achieved by the exhaustive approach will be calculated. Similarly, the percentage of the time of the approaches with respect to the exhaustive time will be calculated by indicating the standard deviation of these.

To complement the previous metrics, the standard deviation is presented. It is a statistical measure that measures how much the values of a set of data vary from their arithmetic mean. It is calculated from the square root of the variance, which, in turn, measures the dispersion of the data. The advantage of using the standard deviation is that it is expressed in the same unit as the original values. For this reason, it is a standard measure of dispersion and quantitative analysis. The equation of the standard deviation of a population σ is given by Equation (17).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \tag{17}$$

where N corresponds to the total number of data in the population, x_i are the individual values in the population, and μ corresponds to the arithmetic mean of the population. In this work, the average results will be presented according to the syntax $mean \pm \sigma$ using the standard deviation.

5. Results and Discussions

In all experiments, the dataset has been normalized into the range $[-1, 1]$ by following recommendations of the ELM creator, and a stratified k-fold cross-validation was performed to prevent overfitting, which allows having a proportion of each class in each fold equal to that of the complete set, with k equal to 5. It should be noted that in order to obtain representative results, each fold will be executed 20 times, making a total of 100 tests of each type for each dataset. In each experiment, the results of two metrics were reported, regardless of which one was used as the objective function.

The section is divided as follows. In Section 5.1, the datasets used during the investigation are presented. In Section 5.2, the parameters with which the algorithms used were configured are presented. In Section 5.3, the results obtained from the exhaustive tests are presented. In Section 5.4, the convergence results obtained by the ABC parameterizers are presented. In Section 5.5, the results as a function of time obtained by the ABC parameterizers and the exhaustive search are presented. In Section 5.6, a comparison in percentages of the results obtained by the ABC parameterizers with respect to the exhaustive search is presented as a function of performance and time. Finally, in Section 5.7, the analysis of the ABC parameterizers is presented based on the concurrence of the locations visited and the redundant computation that is performed.

5.1. Dataset Information

The datasets used in this work are presented below. The area of application of these is health, with the objective of demonstrating the usefulness of the proposals in environments where time is a valuable resource and performing a search in all combinations is infeasible.

5.1.1. Pima Indians Diabetes Database

Pima Indians Diabetes database, hereinafter referred to as the "Diabetes" dataset, is originally from the National Institute of Diabetes and Digestive and Kidney Diseases [43]. The study population consist of individuals of Pima Indian heritage near Phoenix, Arizona, who were selected due to the high incidence rates of diabetes. All patients are female and aged 21 years or older. The goal is to diagnose whether a person has diabetes or not based on various medical measurements. Diabetes was diagnosed according to the World Health Organization criteria [43].

The database correspond to a binary imbalance classification task, composed by 768 instances, 8 features, and 1 binary output (class). Class 0 and 1 represent the diabetes diagnosis, where class 0 indicates non-diabetic patients with 500 samples, and class 1 indicates diabetic patients with 268 samples. The following features are presented:

- Number of pregnancies per patient.
- Plasma glucose concentration at 2 h in an oral glucose tolerance test.
- Diastolic blood pressure (mm Hg).
- Triceps skin fold thickness (mm).
- 2-hour serum insulin (μU/mL).
- Body mass index (weight in kg / (height in m)^2).
- Diabetes pedigree function (probability of diabetes based on family history).
- Age (in years).

5.1.2. UCI Cardiotocography

The UCI Cardiotocography database contains measurements of fetal heart rate (FHR) and uterine contraction (UC) features from cardiotocograms, classified by expert obstetricians at the University of Porto [44]. This dataset has two output labels: morphologic pattern (10 classes) and fetal state (3 classes). In this work, the focus is on addressing the 3-class problem.

Therefore, the dataset corresponds to a multi-class imbalance classification task, composed of 2126 instances, 21 features, and a 3-class output. Class N, S, and P indicate the fetal state where N is normal with 1655 samples, S is suspect with 295 samples, and P is pathologic with 176 samples. The following features are presented:

- Baseline value, accelerations, and light, severe, and prolonged decelerations of FHR (5).
- Uterine contractions (1).
- Fetal movements (1).
- Percentage of time with abnormal short-term variability (STV) and mean of STV (2).
- Percentage of time with abnormal long-term variability (LTV) and mean of LTV (2).
- Minimum and maximum frequency of signal (2).
- Number of zeros and peaks, width, mode, mean, median, variance, and tendency of histogram (8).

5.2. Parameters

As can be seen from the diagrams in the previous section, the ABC-based parameterizers have a common set of parameters that were homogeneously assigned.

- $N = 10$.
- $FoodSources : FS = \frac{N}{2} = 5$.
- $Percentage : P = 0.05$.
- $C = 2^{[-25,25]}$.
- $NeuralNumber = [1, MaxNeurals]$.
- $Totaltravels : T = \left\lceil \frac{P \cdot \#\Omega_{R-ELMDataset}}{N} \right\rceil$.
- $Limit : L = \left\lceil \frac{T}{20} \right\rceil$.

Note that the variable $MaxNeurals$ is obtained by calculating 80% of the total samples of the dataset, being equal to 615 for the Diabetes Dataset and 1701 for the UCI Cardiotocography 3-Class Dataset; the limit parameter (L), which defines the number of times an attempt is made to improve a position in the employed and onlooker phases, is defined according to the number of iterations to be performed; $\#\Omega_{R-ELMDataset}$ is the total size of the search space of the used dataset; and the variable T allows for the guarantee of a path of at least 5% of the total search space, generated by the space given by C and $NeuralNumber$. By solving expression (9) for each dataset, it is obtained in Equations (18) and (19).

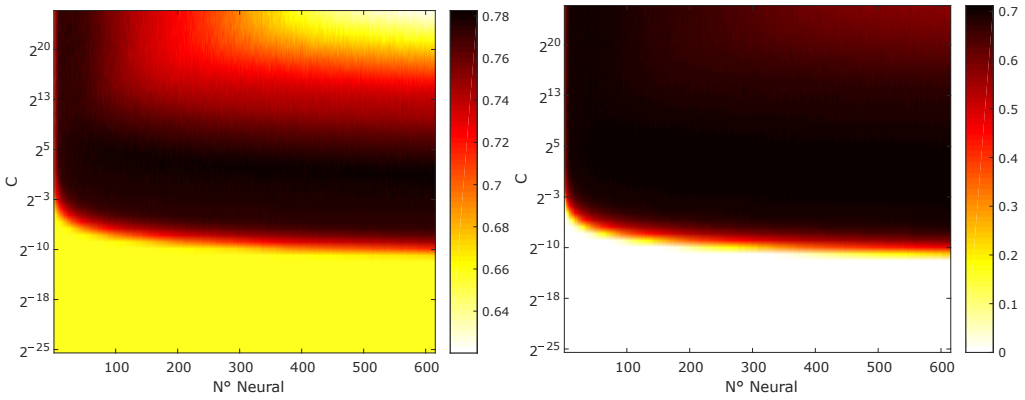
$$\#\Omega_{R-ELMDiabetes} = 51 \cdot 615 = 31365 \tag{18}$$

$$\#\Omega_{R-ELMCardio} = 51 \cdot 1701 = 86751 \tag{19}$$

where the result indicates that the total size of the search space when parameterizing the R-ELM for Diabetes Dataset is 31,365 and UCI Cardiotocography 3-Class Dataset is 86,751 different combinations. The total iterations performed for the datasets are 157 for the Diabetes dataset and 434 for the UCI Cardiotocography 3-Class dataset. As can be seen, the difference in size of the search space is given by a factor of 2.7659.

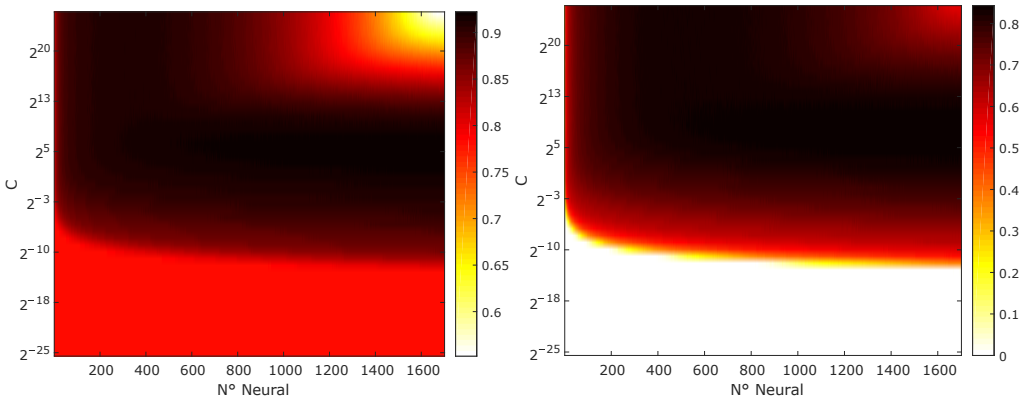
5.3. Exhaustive Search Results

The exhaustive results are shown below in heatmaps that show the geometry of the objective functions in both datasets. Figure 12 corresponds to the Diabetes dataset, where the average of the best position of the accuracy metric is 0.7825 ± 0.02840 with Neural No. = 507 and $C = 2^2$, while that of the g-mean is 0.7123 ± 0.03970 with Neural No. = 158 and $C = 2^4$. The estimated average time (expressed in seconds) is 4492.5096 ± 33.3767. Figure 13 corresponds to the UCI Cardiotocography 3-Class dataset. The average of the best position of the accuracy metric is 0.9222 ± 0.0104 with Neural No. = 1547 and $C = 2^7$, while that of g-mean is 0.8429 ± 0.03250 with Neural No. = 1670 and $C = 2^9$. The estimated average time (expressed in seconds) is 4492.5096 ± 33.3767.



(**a**) Accuracy Heatmap (**b**) G-mean Heatmap

Figure 12. Heatmap of the Exhaustive Search Diabetes Dataset.



(**a**) Accuracy Heatmap (**b**) G-mean Heatmap

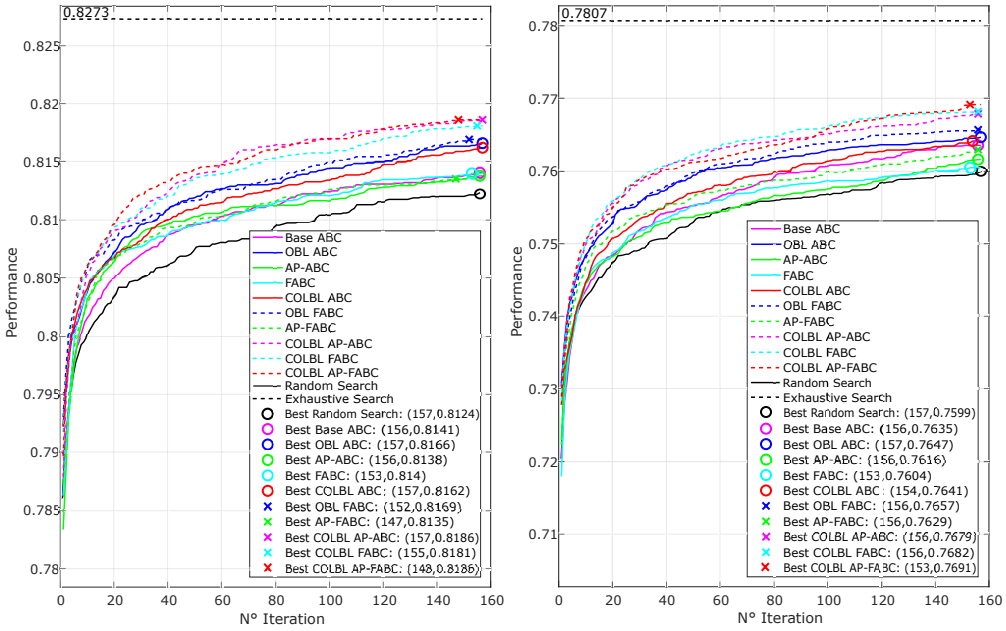
Figure 13. Heatmap of the Exhaustive Search UCI Cardiotocography 3-Class Dataset.

Regarding the performance, the darker areas represent the best performances, while the lighter ones the worst performances. It can be seen that the areas of the performance metrics reach their superior results following an L form and that g-mean is more relaxed than the accuracy in terms of the optimized zone, being overlapped at the same time. In all cases, this happens when the parameter C value is over 2^{-10}. Finally, there is a drop in the performance in the upper right corner in all cases.

5.4. Parameterizer Convergence Performance Results

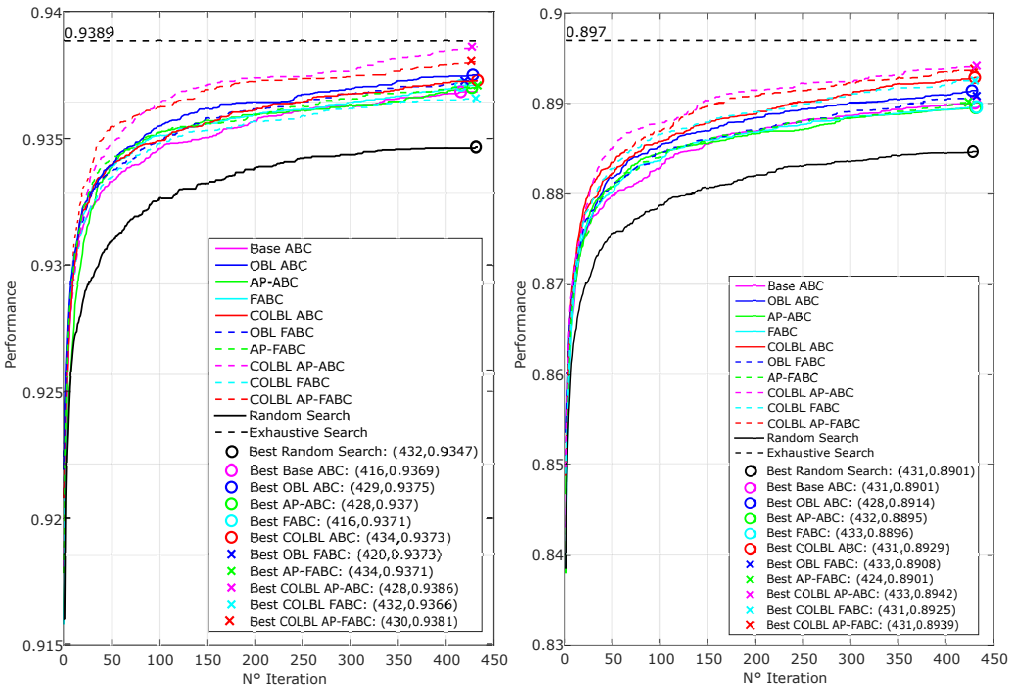
In the following, the results obtained from the tests carried out by the parameterizers are shown. These are delimited at the top by the average of the best value of each iteration of the exhaustive search (100 tests) and at the bottom by the convergence curve of the random search, encapsulating the convergence curves of the parameterizers in the area of improvement. For the random search, groups of random locations of the same size N of the ABC parameterizers were evaluated, storing in each iteration the highest performance in each metric of the study.

The number of iterations T is calculated for each dataset, as presented in Section 5.2, corresponding to 157 iterations for the Diabetes dataset and 434 iterations for the UCI Cardiotocography 3-Class Dataset. Figures 14 and 15 show the convergence obtained by the ABC algorithms during the parameterization of the R-ELM for Diabetes dataset and UCI Cardiotocography 3-Class Dataset, respectively. In each figure, there is a black segmented line that represents the average of the best result found by the exhaustive search for comparison purposes (analyzed below).



(**a**) Objective Function: Accuracy (**b**) Objective Function: G-Mean

Figure 14. Performance Results in Diabetes Dataset.



(**a**) Objective Function: Accuracy (**b**) Objective Function: G-Mean

Figure 15. Performance Results in Cardiotocography 3-Class Dataset.

In Figure 14a, it can be observed how when optimizing the accuracy, the ABC parameterizers are clearly grouped into three levels of convergence (see Section 3). The lowest group is given by the base ABC, AP-ABC, FABC, and AP-FABC. The middle group is made up of COLBL ABC, OBL FABC, and OBL ABC. Consequently, the highest group is formed by COLBL AP-ABC, COLBL FABC, and the method that incorporates all the COLBL AP-FABC modifications. Although these two best methods reach the same optimum value, COLBL AP-FABC reaches it on average 8 iterations earlier. In Figure 14b, the convergence

of the ABC parameterizers can be seen in the case of optimizing the g-mean metric. The methods are not presented grouped; it is worth noting that, again, the COLBL AP-FABC algorithm exposes a higher convergence rate, followed by two other parameterizers that make use of the proposed COLBL AP-ABC and OBL FABC methods.

In Figure 15a,b, the ABC parameterizers are very grouped, all converging to a value very close to the best found by the exhaustive search, highlighting among the parameterizers the COLBL AP-ABC and COLBL AP-FABC with a clear greater convergence. Unlike the previous test, COLBL FABC considerably decreases its convergence when optimizing accuracy, which may imply inconsistency in its performance. Regarding The random search, it is possible to observe that its convergence was considerably reduced in this test; this may be due to the increase in the total search space, where the application of exploration techniques alone is insufficient, and heuristic exploitation criteria are necessary. Because the system performance given by the random search method is the worst and the goal of the paper is experimentation into ABC approaches, we excluded these results in the rest of the paper.

5.5. Parameterizers Times Results

Below are the results obtained during the parameterization of the R-ELM during the tests described in Sections 5.3 and 5.4 as a function of the time invested. Each boxplot includes a zoom area to correctly appreciate the differences between the ABC methods. The hardware and software specifications used are presented in Table 1. The codes were developed in MATLAB R2017B, programming them from scratch and without using specific MATLAB functions, so they can be implemented in other environments with minimal adaptations.

Table 1. Specifications of the system used during testing.

Component	**Specification**
Processor	AMD Ryzen 7 5800H 3.20–4.40 GHz
RAM	32 GB DDR4 3200 MHz (16 × 2)
Disk	SSD NVME Samsung 980 Pro 2 TB
Operating System	Windows 11 Home
Code development software	MATLAB R2017B

Figures 16 and 17 show boxplots as a function of the time (expressed in seconds) taken by the ABC and exhaustive search parameterizers when parameterizing the R-ELM in the Diabetes Dataset and the UCI Cardiotocography 3-Class Dataset.

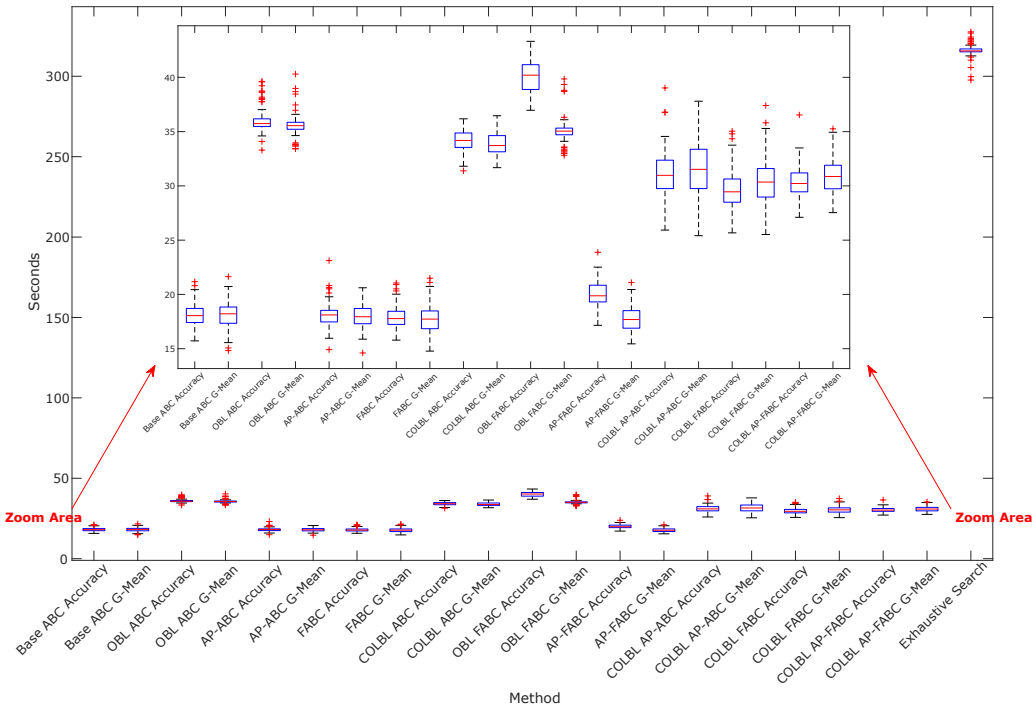


Figure 16. Times of ABC parameterizers and exhaustive search in Diabetes dataset.

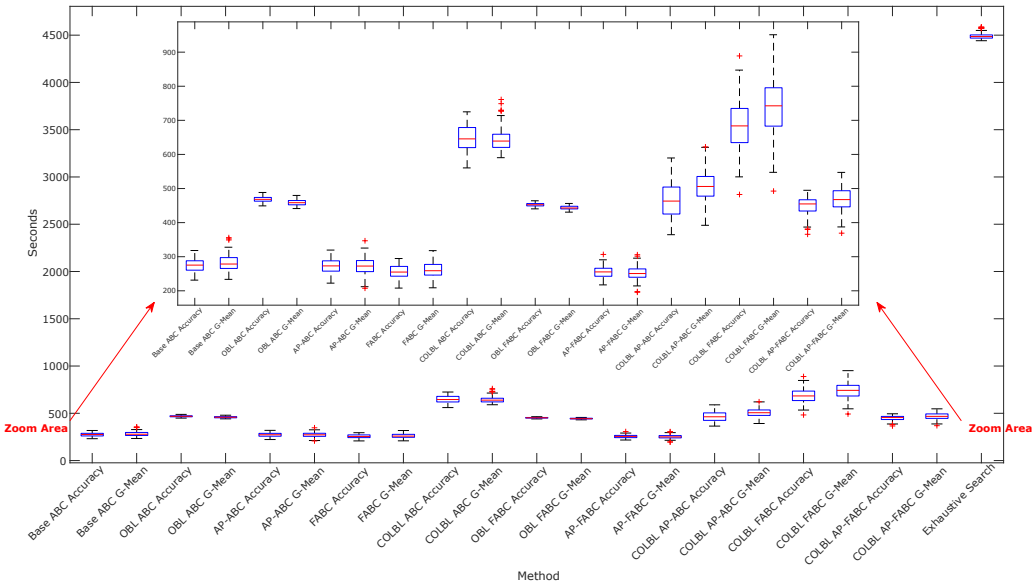


Figure 17. Times of ABC parameterizers and Exhaustive Search in UCI Cardiotocography 3-Class Dataset.

In Figure 16, the times invested by the different parameterizers when optimizing the R-ELM for the Diabetes dataset are presented. It can be observed how the ABC methods are distributed in two groups, which is due to whether or not they use one of the OBL variants. Regarding the methods that do not implement OBL variants, they have similar time behaviors. Among the methods that implement OBL variant techniques, those that include COLBL have a shorter time due to their conditional implementation, which, in turn, produces a greater dispersion in the results. A decrease in execution time can be seen in all methods where FRI are included, and it can also be observed that the results tend to be closer to the average, allowing us to affirm that the aforementioned proposal makes the results more consistent. It is necessary to highlight the time difference that the ABC parameterizers have with respect to the exhaustive search, which will be discussed in more detail in Section 5.6.

In Figure 17, the times taken by the different parameterizers to optimize the R-ELM for the UCI Cardiotocography 3-Class dataset are presented. The two groups can be observed again based on the use or non-use of OBL-based techniques. In this case, the times of the OBL-based techniques are not the highest, being surpassed by the COLBL-based methods. This may be due to the location of the optimal regions of the search space in this dataset, where the value of the neural number is high and adds complexity to the network training process. It is necessary to highlight the COLBL AP-FABC parameterizer that implements the three proposals and has the best average time and the lowest dispersion of all the ABC parameterizers based on COLBL.

5.6. Percentage Relative to Exhaustive Search Results

In Table 2, we present the results of performance and time invested (expressed in seconds) by the ABC parameterizers presented in Sections 5.4 and 5.5 as a percentage of the results obtained by the exhaustive search.

Regarding performance (for accuracy as well as g-mean), it is possible to observe that all the ABC parameterizers obtain a performance of over 98%; therefore, it is possible to affirm that all the ABC parameterizers manage to effectively optimize, to a greater or lesser extent, the R-ELM for the datasets studied.

In terms of the time, the ABC parameterizers use between 5% and 16.6% of the time spent by the exhaustive search in general for both databases. The differences between the methods in this metric are given by the presence of OBL, whether it is conditional or not, the different number of scout bee calls, the focus on R-ELM executions with different number of hidden neurons, and finally, although to a lesser extent, the complexity of the algorithm of each proposal. Although COLBL AP-FABC has three implemented proposals, its time is below the average of the rest of the OBL-based algorithms, which could mean that the complexity of the proposals is negligible with respect to the total execution time of the neural network. It is necessary to take into account that FRI forces the parameterizers to always explore new locations. Consequently, as the local optimum zones are explored, the algorithm acquires a greater probability of locating itself in new zones. It can reduce premature convergence (a recurring problem in the base ABC algorithm), in addition to generating a greater balance between exploitation and exploration, by allowing a convergence towards the global optimum of the search space.

Table 2. Percentage relative to exhaustive search results.

Method	Percentage Relative to Exhaustive Search							
	Diabetes Dataset				UCI Cardiotocography 3-Class Dataset			
	Accuracy Performance	Accuracy Time	G-Mean Performance	G-Mean Time	Accuracy Performance	Accuracy Time	G-Mean Performance	G-Mean Time
Base ABC	98.4044%	5.7097%	97.7997%	5.7120%	99.7870%	6.1155%	99.2308%	6.1856%
OBL ABC	98.7066%	11.3936%	97.9506%	11.2541%	99.8509%	10.4003%	99.3757%	10.1814%
AP-ABC	98.3682%	5.7206%	97.5535%	5.6868%	99.7976%	6.0642%	99.1639%	6.0525%
FABC	98.3924%	5.6694%	97.3998%	5.6094%	99.8083%	5.6627%	99.1750%	5.7498%
COLBL ABC	98.6583%	10.8063%	97.8737%	10.7166%	99.8296%	15.3204%	99.5429%	14.2724%
OBL FABC	98.7429%	12.6844%	98.0786%	11.0936%	99.8296%	10.0494%	99.3088%	9.8366%
AP-FABC	98.3319%	6.3221%	97.7200%	5.6049%	99.8083%	5.6727%	99.2308%	5.5666%
COLBL AP-ABC	98.9484%	9.8242%	98.3604%	9.9635%	99.9680%	10.2865%	99.6878%	11.2390%
COLBL FABC	98.8879%	9.3610%	98.3989%	9.6350%	99.7550%	15.2427%	99.4983%	16.5556%
COLBL AP-FABC	98.9484%	9.6062%	98.5142%	9.7821%	99.9148%	10.1014%	99.6544%	10.3833%

5.7. Visit Parameterizers Results

Below are the results obtained from the average count of visits made by the ABC parameterizers over the search space for both datasets. For each of the methods, the average counts of positions not visited, visited only once, visited between two and five times, visited between six and nine times, and visited ten or more times are considered.

Furthermore, a column of the total average visits and how many of those average visits are unique are included.

In order to visually support the analysis, different cloud points are presented. These results allow us to appreciate how the proposed methods affect the behavior of the parameterizers when traversing the search space and subjectively evaluate consistency, dispersion, and focus in the areas with optimal values. These are obtained from the sum of the 100 tests performed on each dataset with the same configuration as the rest of the experiments. A thresholding is applied for better visualization.

In Tables 3 and 4, the average results of the visits made on the search space of the ABC parameterizers optimizing accuracy and g-mean in the Diabetes dataset are presented, respectively, while Tables 5 and 6 are the same but for the UCI Cardiotocography 3-Class Dataset. It is observed that those parameterizers that include FRI have all their visits unique, those with OBL have twice as many visits, those with COLBL have slightly less than twice as many, and those with AP distribute their visits slightly towards a greater number of repetitions.

Table 3. Diabetes Dataset Visit Results Optimizing Accuracy.

parameterizer	Number of Visits						
	Zero	One	Two to Five	Six to Nine	Ten or More	Total Travels	Unique Travels
Base ABC	30085.08 ± 33.7437	1065.08 ± 42.4354	204.12 ± 13.4842	8.41 ± 3.0587	2.31 ± 1.3536	1660.20 ± 4.2664	1279.92 ± 33.7437
OBL ABC	28905.80 ± 61.7658	1988.74 ± 72.3011	443.62 ± 26.5854	20.12 ± 6.1616	6.72 ± 3.7121	3323.76 ± 9.4164	2459.20 ± 61.7658
AP-ABC	30150.64 ± 27.6181	967.94 ± 32.0904	231.37 ± 11.7882	12.02 ± 3.0251	3.03 ± 1.7259	1664.00 ± 4.7119	1214.36 ± 27.6181
FABC	29707.09 ± 4.7314	1657.91 ± 4.7314	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1657.91 ± 4.7314	1657.91 ± 4.7314
COLBL ABC	29236.88 ± 73.4249	1673.60 ± 85.4938	407.07 ± 23.8870	33.07 ± 5.7441	14.38 ± 3.6119	3177.49 ± 20.5050	2128.12 ± 73.4248
OBL FABC	28044.17 ± 9.5145	3320.72 ± 9.6044	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3320.94 ± 9.4343	3320.94 ± 9.4343
AP-FABC	29706.61 ± 4.7840	1658.39 ± 4.7840	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1658.39 ± 4.7840	1658.39 ± 4.7840
COLBL AP-ABC	29442.06 ± 64.5935	1389.55 ± 66.0473	477.35 ± 22.7673	36.24 ± 5.5816	19.80 ± 3.7118	3179.65 ± 16.3530	1922.94 ± 64.5935
COLBL FABC	28198.98 ± 20.8830	3166.02 ± 20.8830	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3166.02 ± 20.8830	3166.02 ± 20.8830
COLBL AP-FABC	28189.94 ± 18.3863	3175.06 ± 18.3863	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3175.06 ± 18.3863	3175.06 ± 18.3863

Table 4. Diabetes Dataset Visit Results Optimizing G-Mean.

parameterizer	Number of Visits						
	Zero	One	Two to Five	Six to Nine	Ten or More	Total Travels	Unique Travels
Base ABC	30106.49 ± 31.9227	1044.00 ± 37.0242	202.32 ± 12.0007	9.27 ± 2.6585	2.92 ± 1.5679	1653.39 ± 4.4606	1258.51 ± 31.9227
OBL ABC	28937.92 ± 70.5413	1951.64 ± 85.1947	446.80 ± 28.5048	21.90 ± 6.5219	6.74 ± 3.5693	3313.22 ± 10.2529	2427.08 ± 70.5413
AP-ABC	30168.93 ± 35.9883	948.65 ± 39.5113	231.80 ± 12.8055	11.92 ± 3.3505	3.70 ± 1.9358	1656.56 ± 5.0738	1196.07 ± 35.9883
FABC	29713.11 ± 4.5923	1651.89 ± 4.5923	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1651.89 ± 4.5923	1651.89 ± 4.5923
COLBL ABC	29292.39 ± 87.6145	1624.88 ± 99.1406	398.71 ± 24.2501	33.89 ± 5.5448	15.13 ± 3.8498	3140.58 ± 21.6138	2072.61 ± 87.6145
OBL FABC	28053.63 ± 8.2016	3311.26 ± 8.2617	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3311.48 ± 8.1532	3311.48 ± 8.1532
AP-FABC	29712.68 ± 4.4333	1652.32 ± 4.4333	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1652.32 ± 4.4333	1652.32 ± 4.4333
COLBL AP-ABC	29505.34 ± 67.8397	1331.76 ± 68.3087	469.74 ± 23.0255	36.90 ± 5.9671	21.26 ± 4.7623	3144.62 ± 20.8911	1859.66 ± 67.8397
COLBL FABC	28235.78 ± 22.1074	3129.22 ± 22.1074	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3129.22 ± 22.1074	3129.22 ± 22.1074
COLBL AP-FABC	28225.90 ± 22.1389	3139.10 ± 22.1389	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3139.10 ± 22.1389	3139.10 ± 22.1389

Table 5. ABC Methods optimizing Accuracy in UCI Cardiotocography 3-Class Dataset.

parameterizer	Number of Visits						
	Zero	One	Two to Five	Six to Nine	Ten or More	Total Travels	Unique Travels
Base ABC	83875.82 ± 95.4838	2291.85 ± 96.3087	511.34 ± 24.3266	49.31 ± 7.0949	22.68 ± 4.9785	4437.51 ± 4.4823	2875.18 ± 95.4838
OBL ABC	81042.23 ± 157.7937	4489.12 ± 159.4795	1073.59 ± 53.3597	99.26 ± 14.6216	46.80 ± 9.2594	8890.16 ± 8.9609	5708.77 ± 157.7937
AP-ABC	84108.70 ± 75.6688	1975.96 ± 70.7841	584.03 ± 25.7024	55.20 ± 7.3113	27.11 ± 4.4174	4439.59 ± 4.816	2642.30 ± 75.6688
FABC	82313.01 ± 4.4027	4437.99 ± 4.4027	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4437.99 ± 4.4027	4437.99 ± 4.4027
COLBL ABC	81978.26 ± 181.1018	3577.42 ± 174.7433	1001.28 ± 45.8912	104.14 ± 12.5762	80.90 ± 8.3066	8645.72 ± 20.6143	4772.74 ± 181.1018
OBL FABC	77861.02 ± 7.5035	8889.98 ± 7.5035	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8889.98 ± 7.5035	8889.98 ± 7.5035
AP-FABC	82312.91 ± 4.8702	4438.09 ± 4.8702	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4438.09 ± 4.8702	4438.09 ± 4.8702
COLBL AP-ABC	82636.49 ± 141.1983	2821.45 ± 112.4287	1052.93 ± 62.8198	140.73 ± 12.3140	99.40 ± 9.3980	8647.08 ± 21.5668	4114.51 ± 141.1983
COLBL FABC	78123.96 ± 22.8017	8627.53 ± 22.8017	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8627.53 ± 22.8017	8627.53 ± 22.8017
COLBL AP-FABC	78121.76 ± 21.8272	8629.24 ± 21.8272	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8629.24 ± 21.8272	8629.24 ± 21.8272

Table 6. ABC Methods optimizing G-Mean in UCI Cardiotocography 3-Class Dataset.

parameterizer	Number of Visits						
	Zero	One	Two to Five	Six to Nine	Ten or More	Total Travels	Unique Travels
Base ABC	83820.14 ± 104.1866	2345.43 ± 100.4503	520.07 ± 26.2829	44.83 ± 7.0210	20.53 ± 4.5957	4430.90 ± 4.3935	2930.86 ± 104.1866
OBL ABC	80961.16 ± 176.7830	4545.86 ± 166.6301	1107.16 ± 58.7072	95.52 ± 12.9961	41.30 ± 8.2993	8880.06 ± 7.3551	5789.84 ± 176.7830
AP-ABC	84051.50 ± 89.0159	2028.52 ± 82.4429	593.69 ± 30.8220	51.86 ± 7.6145	25.43 ± 4.0358	4432.63 ± 4.2251	2699.50 ± 89.0159
FABC	82319.26 ± 4.4916	4431.74 ± 4.4916	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4431.74 ± 4.4916	4431.74 ± 4.4916
COLBL ABC	81942.60 ± 228.0671	3578.07 ± 221.4293	1051.65 ± 51.3818	104.70 ± 10.4083	73.98 ± 9.8483	8610.73 ± 21.0880	4808.40 ± 228.0671
OBL FABC	77869.80 ± 8.6293	8881.20 ± 8.6293	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8881.20 ± 8.6293	8881.20 ± 8.6293
AP-FABC	82319.95 ± 4.1715	4431.05 ± 4.1715	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4431.05 ± 4.1715	4431.05 ± 4.1715
COLBL AP-ABC	82604.68 ± 156.1923	2827.05 ± 127.1223	1082.95 ± 61.2376	142.01 ± 14.4421	94.31 ± 9.4534	8610.68 ± 21.8039	4146.32 ± 156.1923
COLBL FABC	78151.50 ± 21.2510	8599.50 ± 21.2510	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8599.50 ± 21.2510	8599.50 ± 21.2510
COLBL AP-FABC	78152.01 ± 24.3992	8598.99 ± 24.3992	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8598.99 ± 24.3992	8598.99 ± 24.3992

In order to confirm the previous inferences, Figures 18 and 19, the point clouds generated by the path taken by the ABC parameterizers optimizing the accuracy and g-mean metrics in the Diabetes dataset are depicted, while Figures 20 and 21 show the same information but for the UCI Cardiotocography 3-Class Dataset. To aid in visual inspection, the heat maps on the right correspond to the inclusion of the FRI method with respect to those on the left. In general, all parameterizers have a clear focus on the area with the highest performance seen in the exhaustive search in Section 5.3, in both metrics and in both databases. In other words, all methods are properly implemented and show their utility in terms of time and performance.

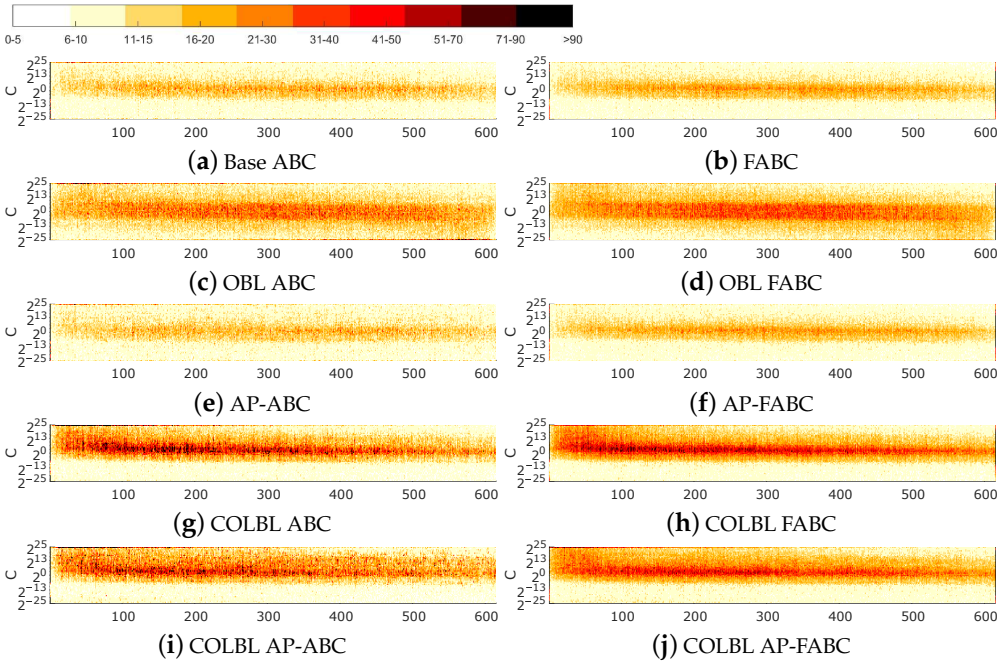


Figure 18. Proposed algorithm point cloud accuracy results in Diabetes dataset.

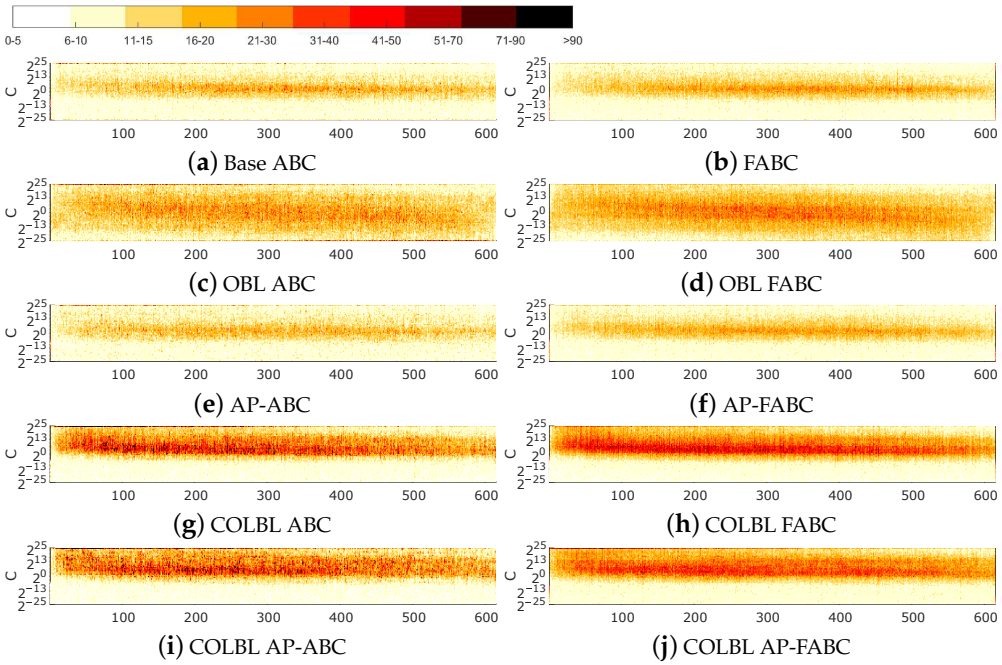


Figure 19. Proposed algorithm point cloud g-mean results in Diabetes dataset.

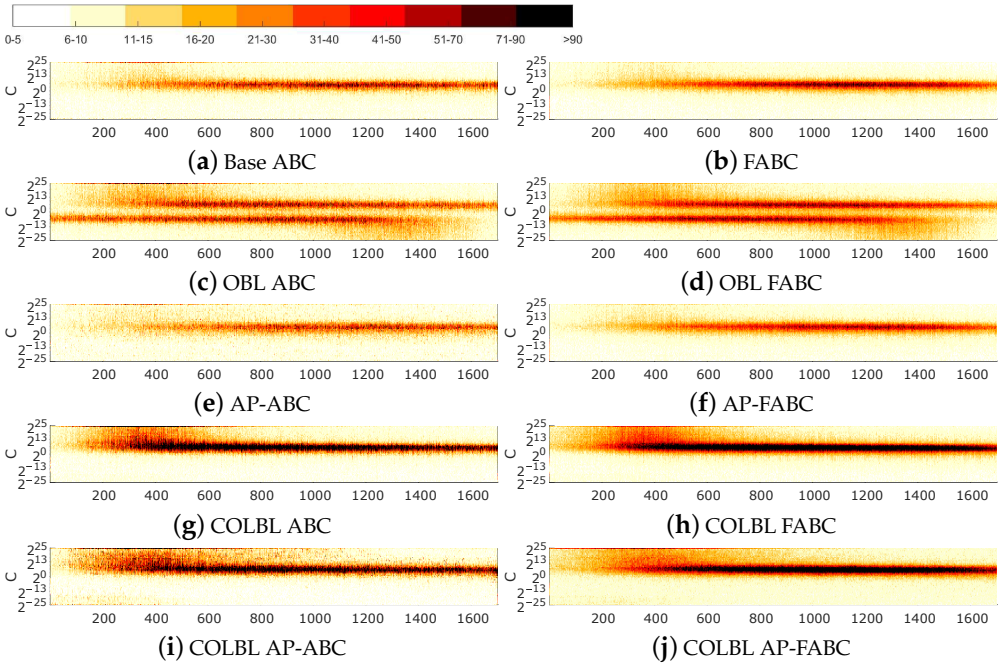


Figure 20. Proposed algorithm point cloud accuracy results in UCI Cardiotocography 3-Class Dataset.

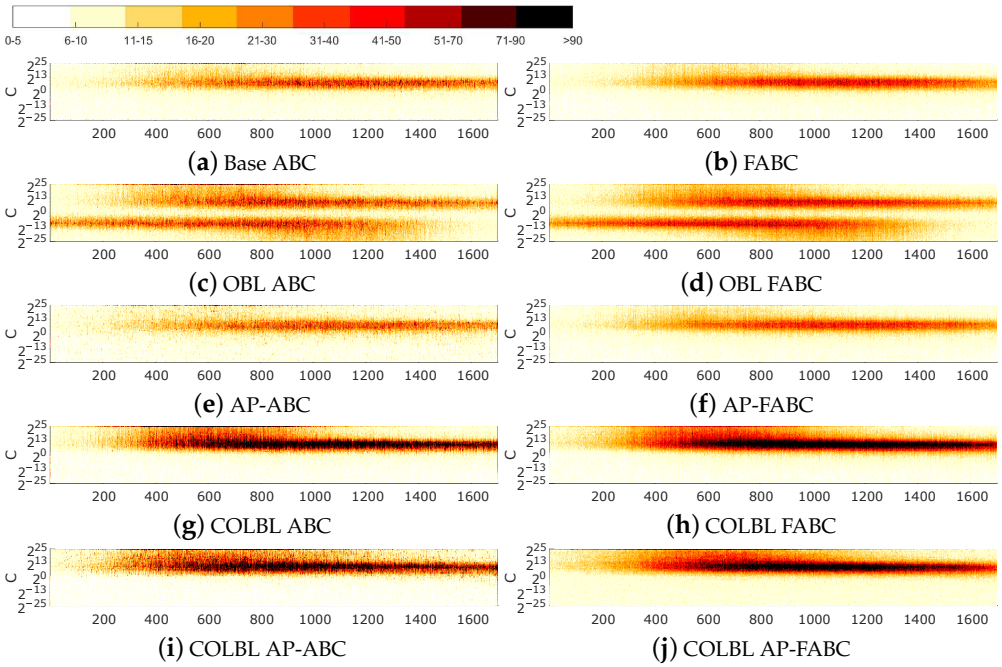


Figure 21. Proposed algorithm point cloud g-mean results in UCI Cardiotocography 3-Class Dataset.

Looking at the same time, in Tables 3–6 and Figures 18–21, it is observed that the specific impact of each proposed method on the parameterizers is the following:

- OBL (OBL ABC): The number of executions increases to just over double, with the standard deviation increasing in a similar range for each number of visits. The distribution of visits remains similar to that with base ABC. The visual analysis shows a mirror effect across the main diagonal, indicating wasted computation in unpromising areas.
- AP (AP-ABC): The number of executions is maintained. The standard deviation decreases initially but increases as the number of visits increases. The distribution of visits moves slightly to the right. In the heat maps, it can be observed that the central focus is blurred with respect to base ABC.
- FRI (FABC): The number of executions is maintained. Each visit is unique, greatly decreasing the standard deviation. Heat maps show a more uniform behavior than the ABC base.
- COLBL (COLBL ABC): The number of executions increases to just under double. Looking at the number of visits, the standard deviation increases to more than double initially but decreases as the number of repetitions of visits increases. The distribution of visits shifts largely to the right, implying a greater repetition of solutions. The heat maps suggest that these repetitions are focused on the areas with the highest performance, eliminating the mirror effect of the OBL.
- OBL and FRI (OBL FABC): All visits are unique. The standard deviation is twice that of FABC. Visually, the mirror effect of OBL is maintained but softened.
- AP and FRI (AP-FABC): All visits are unique. The standard deviation is similar to FABC. As for qualitative analysis, FRI smooths the result.
- AP and COLBL (COLBL AP-ABC): Standard deviation slightly more than twice that of AP-ABC. Hit counts are distributed across a larger number of repeats. The inclusion of AP causes the local scan radius of OLBL to start out maximally expanded, then contract and expand again, creating a visual effect of increased scanning with small, more granular areas being exploited.
- COLBL and FRI (COLBL FABC): All visits are unique. The standard deviation is approximately five times that of FABC. Visually, FRI smooths out the result.

- COLBL and AP and FRI (COLBL AP-FABC): The standard deviation is reduced with respect to COLBL AP-ABC and the hit count is unique. When compared with COLBL FABC in Figure 20, it can be seen that AP disperses the focus from 300 to 400 neurons and from 1600 neurons onward, producing an almost perfect focus in the area of greatest performance with respect to the exhaustive search. A similar effect is seen in Figure 21 from 1400 neurons onward.

6. Conclusions and Future Works

This paper presents the modifications required to use a metaheuristic algorithm as a parameterizer, a set of novel methods to balance the exploration and exploitation of the search space and the implementation of these methods on the ABC algorithm to obtain different parameterizer configurations, where these are exploited to the hyper-parameter optimization problem of R-ELM on two datasets related to the health area. The usefulness of the proposals in this area can be highlighted where time can play an important role with respect to the sacrifice of a small part of the performance, especially in applications of machine learning techniques, where most of the time is invested in the optimization process of the model for its later use.

The methods implemented in the ABC algorithm generate different parameterizer configurations, where some are more focused on exploitation, on exploration, or balanced. For the integrated methods (for instance COLBL AP-FABC), the three proposed methods provide an excellent balance between exploitation and exploration of the search space. This is achieved due to the synergy of the particularities of each of these methods. First, the proposed AP method blurs the areas where solutions are generated, granting greater generality and balance to the search process through iterations. Second, OLBL solves the mirror effect problem of OBL, which implies wasting computation in non-promising areas by centralizing the computation in promising areas. Finally, the proposed FRI method prevents redundant computation in already explored configurations, reducing the probability of falling into premature convergence and increasing the probability of converging to the global optimum. However, depending on the space to be optimized, there is a risk of excessive memory usage.

Each ABC parameterizer meets its objective of parameterizing the R-ELM, although some have a higher convergence than others. In general, all the algorithms achieve at least 98% of the performance found by the exhaustive search and in approximately 5% to 16.6% of their invested time, depending on whether they use OBL-based techniques or not, respectively. The COLBL AP-FABC parameterizer can be highlighted, which has the implementation of the three proposals and its performance (the best on average). The exhaustive search shows an upper bound in the area of improvement in convergence, while the random search represents a lower bound. Future work could address the comparison of this best-performing parameterizer with other heuristics (with and/or without the proposals) or other known state-of-the-art alternatives, such as Bayesian optimization.

Another future work consists of using ABC parameterizers during parameter optimization in other known applications, such as more complex neural networks, digital image filters, and support vector machines, among others. The adaptations presented in Section 4.1 can be used to apply metaheuristics to the aforementioned methods, where the function to be optimized is not differentiable, nor of known geometry, and in the case an exhaustive search may tend to become infeasible as the dimensionality of the search space grows. Based on the above, a possible improvement to be implemented in future research is to improve the exploitation capacity in higher dimensionality applications. This was proposed by the original author in [45], a feature that allows for generating diverse solutions and reducing the stagnation of the method.

Although it has been shown that the three proposed methods together have excellent synergy, achieving an average performance of 99.26% (the best of all) in 9.97% of the time in the exhaustive search, future work remains to address the individual weaknesses of each one. In the case of the AP method, it could be improved by changing the geometry

of its mapping or by setting its minimum value to a value above zero to avoid a period of iterations where the obtaining of equal solutions is repeated. It is essential in deterministic objective functions. In the case of the OLBL technique, it greatly improves the exploitation process at the cost of an additional computational cost. Then, it would be interesting to reduce the generation of opposite local solutions in a way other than a conditional application, in addition to presenting a similar drawback to that of AP. In the case of the FRI method, it has the problem of needing to discretize the search space in order to strengthen the exploration mechanisms. Consequently, it is essential to propose an improvement that allows for working in continuous dimensions so that the parameterizer can generate solutions without equidistant values.

Author Contributions: Conceptualization, P.V.-I. and A.E.P.; methodology, P.V.-I. and A.E.P.; software, P.V.-I. and A.P; validation, P.V.-I. and A.E.P.; formal analysis, P.V.-I., A.E.P., and J.F.-C.; investigation, P.V.-I., A.E.P., J.F.-C., and P.G.; resources, P.V and A.E.P.; data curation, P.V and A.P; writing—original draft preparation, P.V.-I., A.E.P., J.F.-C., and P.G.; writing—review and editing, A.E.P., D.Z.-B., R.A.-G., and D.L.; visualization, P.V.-I., A.E.P., and D.Z.-B.; supervision, D.Z.-B., D.L., and P.G.; project administration, P.V.-I., A.E.P., D.Z.-B., and D.L.; funding acquisition, D.Z.-B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: A.E.P. gratefully acknowledges the financial support provided by ANID-Subdirección de Capital Humano/Doctorado Nacional/2024-21242342. J.F.-C. acknowledges the financial support of 2023 Doctoral Scholarship of Facultad de Ingeniería Universidad Católica del Maule. R.A.-G. gratefully acknowledges the financial support provided by ANID-Subdirección de Capital Humano/Doctorado Nacional/2024-21241043.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061.
2. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: a new learning scheme of feedforward neural networks. In Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), Budapest, Hungary, 25–29 July 2004; Volume 2, pp. 985–990.
3. Zabinsky, Z.B. Random Search Algorithms. Technical Report, Department of Industrial and Systems Engineering, University of Washington, Seattle, WA, USA, 2009.
4. Beni, G.; Wang, J. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics*; Springer: Berlin/Heidelberg, Germany , 1993; pp. 703–712.
5. Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. In *Nature-Inspired Computing and Optimization: Theory and Applications*; Springer International Publishing: Cham, Switzerland, 2017; Volume 10, pp. 475–494.
6. Dorigo, M. Optimization, Learning and Natural Algorithms. Ph. D. Thesis. Politecnico di Milano, Milan, Italy, 1992.
7. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November –1 December 1995; Volume 4, pp. 1942–1948.
8. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-Qaness, M.A.; Gandomi, A.H. Aquila optimizer: a novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250.
9. Trojovskỳ, P.; Dehghani, M. Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. *Sensors* **2022**, *22*, 855.
10. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report, Technical Report-tr06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Turkey, 2005.
11. Liu, H.; Abraham, A.; Snášel, V. Convergence analysis of swarm algorithm. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 1714–1719.
12. Tizhoosh, H.R. Opposition-based learning: a new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005, Volume 1, pp. 695–701;
13. Glover, F. Tabu search—part I. *ORSA J. Comput.* **1989**, *1*, 190–206.

14. Wang, Q.; Song, S.; Li, L.; Wen, D.; Shan, P.; Li, Z.; Fu, Y. An extreme learning machine optimized by differential evolution and artificial bee colony for predicting the concentration of whole blood with Fourier Transform Raman spectroscopy. *Spectrochim. Acta Part Mol. Biomol. Spectrosc.* **2023**, *292*, 122423.
15. Xu, X.; Rogers, R.A.; Estrada, M.A.R. A novel prediction model: ELM-ABC for annual GDP in the case of SCO countries. *Comput. Econ.* **2023**, *62*, 1545–1566.
16. Udaiyakumar, S.; Victoire, T.A.A. Week Ahead Electricity Price Forecasting Using Artificial Bee Colony Optimized Extreme Learning Machine with Wavelet Decomposition. *Teh. Vjesn.* **2021**, *28*, 556–567.
17. Yang, Y.; Duan, Z. An effective co-evolutionary algorithm based on artificial bee colony and differential evolution for time series predicting optimization. *Complex Intell. Syst.* **2020**, *6*, 299–308.
18. Xiao, J.; Zhou, J.; Li, C.; Xiao, H.; Zhang, W.; Zhu, W. Multi-fault classification based on the two-stage evolutionary extreme learning machine and improved artificial bee colony algorithm. *Proc. Inst. Mech. Eng. Part J. Mech. Eng. Sci.* **2014**, *228*, 1797–1807.
19. He, Y.; Li. J.M.; Ruan, S.; Zhao, S. A Hybrid Model for Financial Time Series Forecasting: Integration of EWT, ARIMA with The Improved ABC Optimized ELM. *IEEE Access* **2020**, *8*, 84500–84515.
20. Pushpa, M.; Sornamageswari, M. Early stage autism detection using ANFIS and extreme learning machine algorithm. *J. Intell. Fuzzy Syst.* **2023**, *45*, 4371–4382.
21. 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.
22. Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. *Phys. Stat. Mech. Its Appl.* **2012**, *391*, 2193–2196.
23. Cuevas-Jiménez, E.V.; Oliva-Navarro, D.A.; Díaz-Cortés, M.A.; Osuna-Enciso, J.V. *Optimización: Algoritmos Programados con MATLAB*; Alpha Editorial: Bogota, Colombia, 2016.
24. Karaboga, D.; Gorkemli, B.; Ozturk, C.; Karaboga, N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artif. Intell. Rev.* **2014**, *42*, 21–57.
25. Zhao, J.; Lv, L.; Sun, H. Artificial bee colony using opposition-based learning. In Proceedings of the Genetic and Evolutionary Computing: Proceeding of the Eighth International Conference on Genetic and Evolutionary Computing, Nanchang, China, 18–20 October 2014; Springer: Berlin/Heidelberg, Germany , 2015; pp. 3–10.
26. Sharma, T.K.; Gupta, P. Opposition learning based phases in artificial bee colony. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 262–273.
27. Yigitbasi, E.D.; Baykan, N.A. Edge detection using artificial bee colony algorithm (ABC). *Int. J. Inf. Electron. Eng.* **2013**, *3*, 634.
28. Gonzalez, P.; Iglesias, P.; Silva, E. Restricted particle swarm optimization meta-heuristic method. In Proceedings of the 2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 23–26 October 2023; pp. 1–5.
29. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. https://doi.org/10.1016/j.neucom.2005.12.126.
30. Deng, C.; Huang, G.; Xu, J.; Tang, J. Extreme learning machines: new trends and applications. *Sci. China Inf. Sci.* **2015**, *58*, 1–16. https://doi.org/10.1007/s11432-014-5269-3.
31. Cao, W.; Wang, X.; Ming, Z.; Gao, J. A review on neural networks with random weights. *Neurocomputing* **2018**, *275*, 278–287. https://doi.org/10.1016/j.neucom.2017.08.040.
32. Broomhead, D.; Lowe, D. *Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks*; Technical Report; Royal Signals and Radar Establishment Malvern: Worcestershire, UK, 1988.
33. Pao, Y.H.; Park, G.H.; Sobajic, D.J. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing* **1994**, *6*, 163–180. Backpropagation, Part IV, https://doi.org/10.1016/0925-2312(94)90053-1.
34. Schmidt, W.F.; Kraaijveld, M.A.; Duin, R.P. Feed forward neural networks with random weights. In Proceedings of the International Conference on Pattern Recognition, The Hague, the Netherlands, 30 August–3 September 1992; IEEE Computer Society Press: Los Alamitos, CA, USA, 1992; pp. 389-395.
35. Deng, W.; Zheng, Q.; Chen, L. Regularized extreme learning machine. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining, Nashville, TN, USA, 30 March–2 April 2009; pp. 389–395.
36. Zong, W.; Huang, G.B.; Chen, Y. Weighted extreme learning machine for imbalance learning. *Neurocomputing* **2013**, *101*, 229–242. https://doi.org/10.1016/j.neucom.2012.08.010.
37. Huang, G.; Song, S.; Gupta, J.N.D.; Wu, C. Semi-Supervised and Unsupervised Extreme Learning Machines. *IEEE Trans. Cybern.* **2014**, *44*, 2405–2417. https://doi.org/10.1109/TCYB.2014.2307349.
38. Ye, W.; Feng, W.; Fan, S. A novel multi-swarm particle swarm optimization with dynamic learning strategy. *Appl. Soft Comput.* **2017**, *61*, 832–843.
39. Rahnamayan, S.; Jesuthasan, J.; Bourennani, F.; Salehinejad, H.; Naterer, G.F. Computing opposition by involving entire population. In Proceedings of the 2014 IEEE congress on evolutionary computation (CEC), Beijing, China, 6–11 July 2014; pp. 1800–1807.
40. Yang, S. Enhanced opposition-based differential evolution using dynamic optimum for function optimization. *DEStech Trans. Eng. Technol. Res.* **2017**, *2*, 308–315.
41. Zabala-Blanco, D.; Hernández-García, R.; Barrientos, R.J. SoftVein-WELM: A Weighted Extreme Learning Machine Model for Soft Biometrics on Palm Vein Images. *Electronics* **2023**, *12*, 3608.
42. Zhang, X.; Qin, L. An Improved Extreme Learning Machine for Imbalanced Data Classification. *IEEE Access* **2022**, *10*, 8634–8642. https://doi.org/10.1109/ACCESS.2022.3142724.

43. Smith, J.W.; Everhart, J.E.; Dickson, W.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the the Annual Symposium on Computer Application in Medical Care, San Diego, CA, USA, 30 October–3 November 2021; American Medical Informatics Association: Washington, DC, USA, 1988; p. 261.
44. Ayres-de Campos, D.; Bernardes, J.; Garrido, A.; Marques-de Sa, J.; Pereira-Leite, L. SisPorto 2.0: a program for automated analysis of cardiotocograms. *J. -Matern.-Fetal Med.* **2000**, *9*, 311–318.
45. Akay, B.; Karaboga, D. A modified artificial bee colony algorithm for real-parameter optimization. *Inf. Sci.* **2012**, *192*, 120–142.