A Gradient-Based Particle-Bat Algorithm for Stochastic Configuration Network

Jingjing Liu 1,2, Yefeng Liu 2,* and Qichun Zhang 3

1 Department of Basic Courses, Shenyang Institute of Technology, Shenyang Demonstration Area 113122, China
2 Liaoning Key Laboratory of Information Physics Fusion and Intelligent Manufacturing for CNC Machine, Shenyang Institute of Technology, Shenyang Demonstration Area 113122, China
3 Department of Computer Science, University of Bradford, Bradford BD7 1DP, UK
* Correspondence: liuyefeng@situ.edu.cn

Abstract: Stochastic configuration network (SCN) is a mathematical model of incremental generation under a supervision mechanism, which has universal approximation property and advantages in data modeling. However, the efficiency of SCN is affected by some network parameters. An optimized searching algorithm for the input weights and biases is proposed in this paper. An optimization model with constraints is first established based on the convergence theory and inequality supervision mechanism of SCN; Then, a hybrid bat-particle swarm optimization algorithm (G-BAPSO) based on gradient information is proposed under the framework of PSO algorithm, which mainly uses gradient information and local adaptive adjustment mechanism characterized by pulse emission frequency to improve the searching ability. The algorithm optimizes the input weights and biases to improve the convergence rate of the network. Simulation results over some datasets demonstrate the feasibility and validity of the proposed algorithm. The training RMSE of G-BAPSO-SCN increased by \(5.57 \times 10^{-5}\) and \(3.2 \times 10^{-3}\) compared with that of SCN in the two regression experiments, and the recognition accuracy of G-BAPSO-SCN increased by 0.07% on average in the classification experiments.

Keywords: bat algorithm; gradient; PSO algorithm; stochastic configuration networks

1. Introduction

Randomized algorithms have the characteristics of fast learning and show great potential in machine learning [1–3]. These algorithms generally select the input parameters randomly and then calculate the output parameters based on the least square method. There are many categories of random weight networks. In addition to the typical feedforward neural network, there are also recurrent neural networks with random weights and randomized kernel approximations [4].

At the beginning, most randomized weights networks could not determine the appropriate number of hidden nodes. To solve this problem, the incremental learning algorithm was proposed. The network gradually increases the hidden nodes until the tolerance is reached. This method optimizes the network structure and avoids the waste of hidden nodes. Based on this idea, SLFNN, which has approximation properties, was proposed by Kwok et al [5]. They also adopted a modified Quickprop algorithm to update the network weights. The random Vector Function Link (RVFL) network also used the incremental strategy to build its structure, which probabilistically converges when the input side parameters of the network are appropriately selected [6,7]. Otherwise, the convergence cannot be guaranteed [8]. In 2009, Tyukin I proposed that the RVFL network needed a supervision mechanism, and they verified that the RVFL network could not approximate a given objective function through the experiments [9]. Subsequently, this phenomenon had been further proved mathematically [10]. The proposal of SCN solves the problem that the network did not have universal approximation property [8]. SCN used an inequality constraint mechanism to allocate input parameters randomly and expanded the selection...
range of random parameters adaptively, so as to ensure the approximation of the established stochastic learning network. It was an incremental generative network, and the input parameters were stochastically configured under the inequality constraint, and the output parameters were determined by constructional method or by solving the least square problem. Subsequently, SCN based on the depth model was also proposed, and depth SCN still has universal approximation properties under the inequality supervision mechanism [11]. At present, SCN has been successfully applied to underground airflow quantity modeling, ball mill load condition recognition, sewage quality index estimation, and nonlinear system intelligent modeling [12–15].

The universal approximation property of SCN is an advantage that other randomized learning techniques do not have, as the adaptive search algorithm of the weights and biases tries to keep the parameters in a small range as possible. However, the output weights are usually large, so the generalization performance of the network is affected. Overall, the optimization of SCN focuses on the following aspects: first, improving the generalization performance, for instance, $L_1$ and $L_2$ norm regularization methods based on SCN are proposed respectively to avoid the risk of the model overfitting [13,15]. The partial least squares method is used to calculate the output parameters, replacing the ordinary least squares method [14]. In addition, the famous negative correlation learning method could evaluate the output side parameters based on the SCN mode. The Block Jacobi and Gauss–Seidel methods are used to solve the output weights of ill-posed equations iteratively based on heterogeneous feature groups, in which the convergence analysis is given and the uniqueness of these iterative solutions is proved [16]. The second aspect is to optimize the generation mode of the network; some scholars have proposed to change the incremental mode of the hidden nodes in the process of SCN construction, and the incremental mode of hidden nodes was improved from single incremental to block incremental [17,18]. The third aspect is to optimize the hyperparameters of the network. In the discussion of the input weights, some scholars indicate that the method based on the inequality supervision mechanism to search weights leads to a certain rejection rate, therefore, the conditions of inequality constraints should be relaxed, so as to improve the acceptance rate of the random value and reduce the number of iterations for the searching of input weights [19]. However, the accuracy of the network will be reduced once the inequality constraints are relaxed, which has a certain cost. Some scholars also focus on regularization parameters and scale factors of weights and bias that affect the network performance, and optimize them using a chaotic sparrow search algorithm, which makes the network achieve better performance [20]. Meanwhile, some scholars also applied various feedback learning mechanisms to optimize SCN [21,22].

There are few studies on how to optimize the input side parameters directly. SCN searches the input side parameters randomly until the parameters could satisfy the inequality constraints. The specific method is to expand the search interval gradually linearly, and randomly generate a large number of weights and biases in the interval to judge whether the inequality constraints are satisfied, then select the best one. The method will affect the efficiency of SCN. Therefore, an efficient optimization algorithm needs to be designed. The gradient descent method is often used to solve optimization problems in practical engineering, that is, to solve for a minimum in the direction of gradient descent or to solve for a maximum in the direction of gradient ascent. However, this method is easy to fall into the local minimum for non-convex optimization problems, so it needs to be combined with an intelligent optimization algorithm to achieve better performance [23,24]. The bat algorithm [25,26] and Particle Swarm Optimization (PSO) algorithm [27] are population-based stochastic optimization technical algorithms, which seek the optimal solution by imitating the group behavior of the biological world. The bat algorithm is a search algorithm to simulate the predation behaviors of bats, which has been successfully applied to microgrid scheduling [28], harmful plants classification [29], text categorization [30], and the categorization salesman problem [31]. These algorithms could search more regions in the solution space of the objective function simultaneously,
which is flexible and easy to implement, and requires fewer parameters to be adjusted. In addition, the gradient information could be used to determine the search direction because of the randomness of intelligent optimization algorithms. Therefore, using the hybrid optimization algorithm to search the input parameters of SCN could optimize the network structure and improve the convergence speed.

This paper focuses on improving the search algorithm for the input weights and biases. In order to accelerate the convergence speed, an optimization model was established based on the inequality constraint mechanism by analyzing the convergence principle of SCN. Then, an improved optimization algorithm (G-BAPSO) is proposed, which combines the gradient information of the objective function and searches for the optimal weights and biases using the hybrid PSO and bat algorithm. Finally, an optimized stochastic configuration network based on the G-BAPSO algorithm is established.

The structure of this paper is as follows: The theory of SCN is introduced in Section 2; G-BAPSO-SCN is proposed in Section 3, including the optimization model and algorithm. In Section 4, the G-BAPSO algorithm is verified on some benchmark functions, and the performance of G-BAPSO-SCN and other models is compared in some regression and classification problems. The experimental results are analyzed in Section 5, and Section 6 gives the conclusion of this paper.

2. Preliminaries

2.1. Stochastic Configuration Network

SCN has been a powerful data modeling tool in recent years. The model is incrementally generated by stochastic configuration algorithms, which performs well in classification and regression problems and has the ability of universal approximation. The structure of SCN is shown in Figure 1, where \( x \) denotes the input data, \( s \) denotes the activation function, and \( f \) denotes the output of network.

![Figure 1. The structure of SCN with a hidden layer.](image)

For the dataset \( X = [x^1, x^2, \ldots, x^N], x^i \in \mathbb{R}^d, i = 1, 2, \ldots, N \), and the corresponding output datasets \( T = [t^1, t^2, \ldots, t^N], t^i \in \mathbb{R}^m \). If \( L - 1 \) nodes have been configured on the network, the outputs of \( L \)-th node in the hidden layer is calculated by Formula (1)

\[
s_L = h(w_L^T x_u + b_L), \quad (L = 1, 2, \ldots; \quad u = 1, 2, \ldots, d)
\]

where \( h \) stands for the activation function, \( w_L \) and \( b_L \) is the weight and bias of the first layer. The weights of the second layer is calculated by the least squares algorithm, as shown in Formula (2).

\[
\hat{\beta}_{L,q} = \frac{\langle e_{L-1,q} s_L \rangle}{\|s_L\|^2}
\]

\[
e_{L-1,q} = t_q - f_q, \quad (q = 1, 2, \ldots, m)
\]
where \( e_{L-1,q} \) stands for the \( q \)-th error of the network output when there are already \( L - 1 \) nodes in the hidden layer. \( f_q \) stands for the \( q \)-th output of SCN, which is calculated by Formula (4), and the error of SCN with \( L \)-nodes is shown in Formula (5).

\[
f_q = \sum_{i=1}^{L} \beta_{iq} s_i, \quad (q = 1, 2, ..., m)
\]

(4)

\[
e_L = [e_{L,1}, ..., e_{L,m}] = t - f
\]

(5)

\[
f = [f_1, ..., f_m]
\]

(6)

\( w_L \) and \( b_L \) need satisfy the supervision mechanism (7).

\[
\sum_{q=1}^{m} (e_{L-1,q} s_L)^2 \geq \text{bound}_s^2 \delta_L
\]

(7)

\[
\delta_L = (1 - r - \mu_L) \|e_{L-1}\|^2
\]

(8)

where \( \text{bound}_s \) stands for the upper bound of the activation function, \( \mu_L \) is a sequence that goes to zero, and \( r \) is a regularization parameter close to 1. Hidden nodes are added in this sequence until the model reaches the tolerance, that is, when the error between the model outputs and target outputs reaches the preset threshold, the structure, weight, and threshold parameters of the SCN are determined, then the learning process stops. It has been proved mathematically that the model has universal approximation capability, namely \( \lim_{L \to +\infty} \|e_L\| = 0 \). Principles and Formulas (1)–(8) of SCN are from Ref. [8].

The specific search algorithm for the input weights and biases of SCN is as follows.

Step 1. Set the sequence of interval length for searching the parameters, 
\( \lambda = \lambda_{\min} : \Delta \lambda : \lambda_{\max} \).

Step 2. Generate a set of random values in the interval \([ -\lambda_{\min}, \lambda_{\min} ]\), then judge whether it satisfies the inequality constraint or not;

Step 3. If there are parameters satisfying the inequality constraints, the parameters that could maximize \( e_L \) (Formula (9)) are selected to configure the new hidden node. Otherwise, expand the search interval to \([ -(\lambda_{\min} + \Delta \lambda) : (\lambda_{\min} + \Delta \lambda) ]\), and repeat the above steps until the tolerance is reached [8].

\[
e_L = \sum_{q=1}^{m} \left( \frac{(e_{L-1,q}(X)^T s_L(X))^2}{s_L(X)^T s_L(X)} - (1 - r - \mu_L)e_{L-1,q}(X)^T e_{L-1,q}(X) \right)
\]

(9)

\[
s_L(X) = [s_L(w_1^T x_1 + b_L), s_L(w_1^T x_2 + b_L), ..., s_L(w_1^T x_N + b_L)]
\]

(10)

2.2. Particle Swarm Optimization Algorithm and Bat Algorithm

PSO is an iterative algorithm based on the simulation of the foraging process of birds. Firstly, the population is randomly initialized, including the position and velocity of the individual. Secondly, we calculate the fitness of each particle to obtain the initial optimal solution of each particle. Thirdly, the velocity and position are updated by Formulas (11) and (12) in each iteration until the solution meets the conditions.

\[
v_{t+1} = v_t + c_1 \text{rand}_1 (p_{bt} - p_t) + c_2 \text{rand}_2 (g_{bt} - p_t)
\]

(11)

\[
p_{t+1} = p_{t} + v_{t+1}
\]

(12)

where \( v_t \) stands for the velocity of the individual at time \( t \), \( p_t \) stands for the location of the individual at time \( t \), \( p_{bt} \) and \( g_{bt} \) stand for the individual extremum and global extremum, respectively, at time \( t \). \( c_1 \) and \( c_2 \) are learning factors, \( \text{rand}_1 \) and \( \text{rand}_2 \) are random numbers.

The bat algorithm is a heuristic search algorithm to simulate the predation behaviors of bats. The bats constantly adjust the positions to determine the optimal solution according to
the unique frequency, wavelength, and loudness. It has simple model and fast convergence speed. However, the optimization accuracy of the algorithm is low. It is based on the following assumptions: Firstly, all bats could sense distance by echolocation, and distinguish targets and obstacles in a special way; Secondly, bats fly randomly at the location \( p_i \) with the speed \( v_i \), search for targets with loudness, frequency, and variable wavelength, and adjust the wavelength (or frequency) and pulse emission rate \( rp \) automatically by the distance from the target; Thirdly, the loudness gradually decreases from \( A_0 \) to \( A_{min} \) \[32\].

The updated formulas of frequency, speed, and position are as follows:

\[
fre_i = fre_{\text{min}} + (fre_{\text{max}} - fre_{\text{min}})\beta
\]

\[
v_{i}^{t+1} = v_{i}^{t} + (p_{i}^{t} - p^{*})fre_{i}
\]

\[
p_{i}^{t+1} = p_{i}^{t} + \alpha_{r}e_{i}L
\]

where \( fre_{i} \) stands for the frequency of the \( i \)-th bat, its adjustment range is \([fre_{\text{min}}, fre_{\text{max}}]\), \( \beta \in [0, 1] \), \( p^{*} \) is the current optimal solution.

Generates a random number \( rand_{1}^{t} \) for the current local search field. If \( rand_{1}^{t} > rp_{i} \), update the value of \( p \) by Formula (16).

\[
p_{\text{new}} = p_{\text{old}} + \epsilon A^{t}
\]

where \( \epsilon \in [-1, 1] \), \( A^{t} \) stands for the average loudness of bats at time \( t \).

The loudness drops to a fixed value as the bat approaches its target, then \( rp \) continues to increase. Generate a random number \( rand_{0}^{t} \), if \( rand_{0}^{t} < A_{i} \) and \( f(p_{\text{new}}) > f(p_{\text{old}}) \), \( p_{\text{new}} \) is accepted, namely \( p_{i}^{t+1} = p_{\text{new}} \). The loudness and the pulse emission rate are updated under the Formulas (17) and (18).

\[
A_{i}^{t+1} = \alpha A_{i}^{t}
\]

\[
rp_{i}^{t+1} = rp_{i}^{t}[1 - e^{-\delta t}]
\]

where \( 0 < \alpha < 1 \) and \( \delta > 0 \), \( A_{i}^{t} \) and \( rp_{i}^{t} \) stand for the loudness and pulse emission rate of the \( i \)-th bat at time \( t \), respectively.

### 3. G-BAPSO-SCN

#### 3.1. Establish the Optimization Model

Based on Equation (19) \[8\], SCN will converge if the inequality (20) is satisfied.

\[
||e_{L}||^2 - (r + \mu_{L})||e_{L-1}||^2 = (1 - r - \mu_{L})||e_{L-1}||^2 - (\sum_{q=1}^{m}⟨e_{L-1}, s_{L}⟩^2) / ||s_{L}||^2 \geq 0
\]

According to the convergence principle of SCN \[8\], taking the difference between the error of the model with \( L + 1 \) hidden nodes and the error of the model with \( L \) hidden nodes as the objective function is beneficial to directly improve the convergence rate of the model. The larger the \( \phi(w_{L}, b_{L}) \) is, the faster the network convergence rate is. Therefore, the optimization configuration of the input parameters is transformed into solving the following optimization problem under the constraint of inequality (20). \( \phi(w_{L}, b_{L}) \) is a continuous function, and the convergence speed of the network could be improved when the input weights and biases are configured by the solutions of the optimal problem (21).

\[
\max : \phi(w_{L}, b_{L}) = \frac{||s_{L}||^2}{||S_{L}||^2} - (1 - r - \mu_{L})||e_{L-1}||^2
\]
3.2. Solve the Optimization Problem Based on the G-BAPSO Algorithm

The input parameters are generated randomly in SCN, and the selected range will gradually and linearly expand to search the random parameters that meet the criteria (7). In order to make the search algorithm more flexible and speed up the convergence, an individual elimination strategy of the BAT algorithm under the framework of PSO. When the learning factor. The specific description of the G-BAPSO algorithm is shown in Algorithm 1, and the flow chart is presented in Figure 2.

3.2.1. Calculate the Gradient Information

The gradient direction plays an important role in the search process. Calculate the gradient \( \nabla \phi = [\frac{\partial \phi}{\partial w_{L,1}}, \frac{\partial \phi}{\partial w_{L,2}}, ..., \frac{\partial \phi}{\partial w_{L,d}}, \frac{\partial \phi}{\partial b_L}] \) based on the objective function (21). The results are shown in Formulas (22)–(25), where \( d \) stands for the features of the input samples, \( m \) stands for the output dimension, \( N \) stands for the number of samples, and \( L \) stands for the hidden node.

\[
\frac{\partial \phi}{\partial w_{L,u}} = 2 \sum_{q=1}^{m} \left[ \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} \right) \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} \frac{\partial \phi}{\partial w_{L,u}} \right) \sum_{i=1}^{N} (s_{L})^{2} - \left( \sum_{i=1}^{N} s_{L} \frac{\partial \phi}{\partial w_{L,u}} \right) \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} s_{L}^{i} \right) \right] \quad \text{||}s_{L}||^{4}
\]

\[
\frac{\partial s_{L}^{i}}{\partial w_{L,u}} = s_{L}(1-s_{L})x_{a}, \quad u = 1, 2, ..., d
\]

\[
\frac{\partial \phi}{\partial b_{L}} = 2 \sum_{q=1}^{m} \left[ \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} \right) \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} \frac{\partial \phi}{\partial b_{L}} \right) \sum_{i=1}^{N} (s_{L})^{2} - \left( \sum_{i=1}^{N} s_{L} \frac{\partial \phi}{\partial b_{L}} \right) \left( \sum_{i=1}^{N} \epsilon_{L-1,q}^{i} s_{L}^{i} \right) \right] \quad \text{||}s_{L}||^{4}
\]

\[
\frac{\partial s_{L}^{i}}{\partial b_{L}} = s_{L}(1-s_{L})
\]

3.2.2. G-BAPSO Algorithm

The G-BAPSO algorithm introduces the local search mechanism and adaptive individual elimination strategy of the BAT algorithm under the framework of PSO. When the pulse frequency is low and the current individual fitness is less than the average fitness, Formula (16) is used to update the local individual position. Otherwise, individual velocity is updated by combining gradient information in the process of global search. With the guidance of gradient direction, the search speed is accelerated; the search mechanism of the bat-PSO algorithm could fully explore the solution space and help to jump out of the local minimum.

The combination of the two could improve the search efficiency. The global update formula is shown in Formulas (26)–(28), where \( \text{grad} \) represents the gradient and \( c_3 \) is a learning factor. The specific description of the G-BAPSO algorithm is shown in Algorithm 1, and the flow chart is presented in Figure 2.

\[
v_{t+1} = v_{t} + c_1 \text{rand} (pb_{t} - p_{t}) + c_2 \text{rand} (gb_{t} - p_{t}) + c_3 \text{grad}
\]

\[
p_{t+1} = p_{t} + v_{t+1}
\]

\[
\text{grad} = [\frac{\partial \phi}{\partial w_{L,1}}, \frac{\partial \phi}{\partial w_{L,2}}, ..., \frac{\partial \phi}{\partial w_{L,d}}, \frac{\partial \phi}{\partial b_{L}}]
\]
where grad represents the gradient and $c_3$ is the learning factor.

Figure 2. The flow chart of G-BAPO.
Algorithm 1 The G-BAPSO algorithm
1: Parameters initialization: \(c_1, c_2, c_3, \text{maxgen}, \text{popsize}, \text{popmin}, \text{popmax}, \) initial pulse frequencies \(r_p\), pulse frequency enhancement coefficient \(\sigma\);
2: Initialize the population position \((w \text{ and } b)\) and the velocity \(v\);
3: Calculate the fitness by (21), record the individual optimal position of the primary particle;
4: if the optimum fitness satisfies the inequality (20) then
5: record the global best position;
6: else
7: Adjust the value of \(r\) and continue to judge until inequality (20) is satisfied;
8: end if
9: for \(i = 1 \rightarrow \text{maxgen}\) do
10: for \(j = 1 \rightarrow \text{sizepop}\) do
11: if \(r_j < \text{rand} \text{ and fitness}_j < \text{mean(fitness)}\) then
12: Update individual location according to (16);
13: else
14: Calculate the gradient according to (22)–(25), update individual speed according to (26), and then update individual position according to (27);
15: end if
16: Updating individual fitness values and pulse frequencies;
17: end for
18: for \(k = 1 \rightarrow \text{sizepop}\) do
19: if fitness\(_k\) > fitness\(_{gbest}\)(i) then
20: Update individual optimal position;
21: end if
22: if fitness\(_k\) > fitness\(_{zbest}\)(i) then
23: Update optimal location of the group;
24: end if
25: end for
26: end for
27: return \(z_{best}\)

Set \(L_{max}\) as the maximum number of hidden layer nodes. Under the G-BAPSO algorithm, the optimized configuration algorithm of SCN, called ‘G-BAPSO-SC’, is shown in Algorithm 2.

Algorithm 2 The G-BAPSO-SC algorithm
1: Initialize network parameters \(e_0 = [t_1, t_2, ..., t_N]^T, 0 < r < 1;\)
2: while \(dol \leq L_{max}\) do
3: Calculate \(w_L\) and \(b_L\) by G-BAPSO;
4: Calculate \(W_{L,q} = (e_{L-1,q}h_{L}^T)/(h_{L}^T h_{L}^T),\)
5: \(q = 1, 2, ..., m;\)
6: \(W_L = [W_{L,1}, W_{L,2}, ..., W_{L,m}]^T;\)
7: Calculate \(e_L = e_{L-1} - W_L h_{L}^T;\)
8: Update \(e_0 := e_L; L = L + 1;\)
9: end while
10: return \(W_1, W_2, ..., W_L;\)
11: \(w^* = [w_1^*, w_2^*, ..., w_n^*],\)
12: \(b^* = [b_1^*, b_2^*, ..., b_L^*].\)

4. Numerical Experiments
4.1. Evaluation of the G-BAPSO Algorithm
In order to illustrate the efficiency of G-BAPSO, four benchmark functions (29)–(32) are selected to evaluate its performance, and the minimum value of each function is zero. The G-BAPSO algorithm is used to optimize these functions, and it will be compared
with Gradient-based PSO (GPSO), PSO, bat, and FOA algorithm. The error of the optimal solution is used as an evaluation index for comparison.

\[ f_1 = \sum_{i=1}^{3} x_i^2, x_i \in [-5.12, 5.12] \]  
\[ f_2 = 100(x_1 - x_2)^2 + (1 - x_1)^2, x_i \in [-2, 2] \]  
\[ f_3 = \frac{1}{4000}(x_1^2 + x_2^2) - \cos(x_1) \cos(x_2) + 1, x_i \in [-6, 6] \]  
\[ f_4 = \sum_{i=1}^{3} (\sum_{j=1}^{i} x_j)^2, x_i \in [-5, 5] \]

Continuous functions are selected as the benchmark functions in this section because the objective function proposed in Section 2 is multivariate continuous functions. \( f_3 \) is multi-peak function and the rest are concave function. The graphs of the four functions are shown in Figures 3–6. Figures 3 and 6 show the graphs of binary functions corresponding to \( f_1 \) and \( f_4 \), respectively.

![Figure 3. The figure of function \( f_1 \).](image)

![Figure 4. The figure of function \( f_2 \).](image)

![Figure 5. The figure of function \( f_3 \).](image)
Figure 6. The figure of function $f_4$.

Popsize, $c_1$, $c_2$, $c_3$ and the number of iterations have an important influence on the G-BAPSO algorithm. In order to determine the optimal value, Refs. [33,34] suggested that $c_1$ and $c_2$ should be 2. Refs. [35,36] conducted sensitivity analysis on popsize, $c_1$, $c_2$ and other parameters of acceleration. Based on the benchmark test function $f_1$, the sensitivity analysis of relevant parameters of GBAPOSO is shown in Tables 1–3, and the parameters used in other experiments also follow this method.

Table 1 shows the comparison results of the minimum values of the function $f_1$ when $c_1$, $c_2$ and popsize are changing. It can be seen that when $c_1$, $c_2$ and $c_3$ are fixed, the larger the popsize, the smaller the error. However, the order of magnitude of the error only changes slightly as the population size increases to 75 or 100, and the larger the popsize, the longer the calculation time consumed by the algorithm. Therefore, it is necessary to select an appropriate value that can balance the calculation time and error. Fifty is selected as the population number in solving the minimum value of $f_1$. When the popsize and $c_3$ are fixed, the smaller $c_1$ and $c_2$ are, the smaller the error will be. When $c_1$ and $c_2$ drop to 0.3, the magnitude of the error has reached $10^{-23}$, which is within the acceptable range.

Table 2 shows the sensitivity analysis for $c_3$. As $c_3$ increases from 0.2 to 0.8, the error tends to increase first and then decrease. This is because the BAPSO algorithm plays a major role in the algorithm when $c_3$ is very small, and the gradient descent algorithm plays a major role when $c_3$ becomes large. When the value of $c_3$ is near 0.4, there is a balance between the two, which also verifies the effectiveness of the hybrid algorithm. Table 3 shows the sensitivity analysis for the iterations. As the iterations increase, the error of the optimal solution becomes smaller and smaller. However, the more iterations, the greater the time cost. It can be seen that when iterations are 20, the error has reached the magnitude of $10^{-23}$, which is within the acceptable range.

<table>
<thead>
<tr>
<th>Popsize</th>
<th>The Error of the Minimum Value of ‘f1’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$c_{1,2,3} = 0.3, 0.3, 0.4$</td>
</tr>
<tr>
<td></td>
<td>$c_{1,2,3} = 1, 1, 0.4$</td>
</tr>
<tr>
<td></td>
<td>$c_{1,2,3} = 2, 2, 0.4$</td>
</tr>
<tr>
<td>25</td>
<td>$3.4575 \times 10^{-23}$</td>
</tr>
<tr>
<td>50</td>
<td>$1.0590 \times 10^{-20}$</td>
</tr>
<tr>
<td>75</td>
<td>$1.8236 \times 10^{-23}$</td>
</tr>
<tr>
<td>100</td>
<td>$4.0223 \times 10^{-12}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c_3$</th>
<th>The Error of the Minimum Value of ‘f1’</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$6.2255 \times 10^{-18}$</td>
</tr>
<tr>
<td>0.4</td>
<td>$1.816 \times 10^{-23}$</td>
</tr>
<tr>
<td>0.6</td>
<td>$2.9205 \times 10^{-23}$</td>
</tr>
<tr>
<td>0.8</td>
<td>$4.1324 \times 10^{-18}$</td>
</tr>
</tbody>
</table>
Table 3. The sensitivity analysis of the iterations.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>The Error of the Minimum Value of ‘f1’</th>
<th>Run Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$8.0748 \times 10^{-12}$</td>
<td>0.0202</td>
</tr>
<tr>
<td>20</td>
<td>$1.1816 \times 10^{-23}$</td>
<td>0.0246</td>
</tr>
<tr>
<td>50</td>
<td>$7.7729 \times 10^{-70}$</td>
<td>0.0511</td>
</tr>
</tbody>
</table>

Table 4 shows the parameter Settings. The program was written on MATLAB R2019a and run on 8G memory and 1.6 Ghz CPU. Each algorithm was independently run 200 times. The experimental results are shown in Table 5.

Table 4. Parameters setting.

<table>
<thead>
<tr>
<th>Function</th>
<th>Popsize</th>
<th>Iterations</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$R_f$</th>
<th>$r_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>50</td>
<td>20</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.9</td>
<td>3</td>
</tr>
<tr>
<td>$f_2$</td>
<td>200</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>0.0001</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>$f_3$</td>
<td>100</td>
<td>50</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>$f_4$</td>
<td>50</td>
<td>10</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5. The average error of the optimal solution in 200 experiments.

<table>
<thead>
<tr>
<th>Function</th>
<th>G-BAPSO</th>
<th>G-PSO</th>
<th>PSO</th>
<th>Bat</th>
<th>FOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$1.1816 \times 10^{-23}$</td>
<td>$2.6410 \times 10^{-23}$</td>
<td>$5.4593 \times 10^{-6}$</td>
<td>$4.0801 \times 10^{-3}$</td>
<td>$1.2026 \times 10^{-2}$</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$2.3441 \times 10^{-15}$</td>
<td>$3.1405 \times 10^{-15}$</td>
<td>$2.8753 \times 10^{-15}$</td>
<td>$5.0260 \times 10^{-3}$</td>
<td>$2.2302 \times 10^{-3}$</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0</td>
<td>$1.3776 \times 10^{-13}$</td>
<td>$9.3501 \times 10^{-5}$</td>
<td>$0.0114 \times 10^{-2}$</td>
<td>$1.8720 \times 10^{-1}$</td>
</tr>
<tr>
<td>$f_4$</td>
<td>$6.8003 \times 10^{-6}$</td>
<td>$7.9036 \times 10^{-6}$</td>
<td>$3.2363 \times 10^{-5}$</td>
<td>$3.9358 \times 10^{-3}$</td>
<td>$1.8720 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

It can be seen from the results of four groups of experiments that G-BAPSO performs well. Compared with PSO, bat, and FOA, the optimal solution has a significant advantage in order of magnitude. It is worth noting that the performance of G-PSO is comparable to that of G-BAPSO when gradient information is introduced into PSO, indicating that gradient factors are relatively important for these four functions, especially concave functions with obvious characteristics, and it is easy to find the optimal solution quickly according to the gradient direction search. Therefore, the first two algorithms are more efficient than the last three algorithms. When comparing the first two algorithms, one can observe that G-BAPSO performs well because it adds a local search and adaptive elimination strategy of the bat algorithm, which can improve the convergence speed of the algorithm. Note that the error of G-BAPSO and G-PSO for $f_3$ is zero, which means the error is very small, approaching zero.

4.2. Performance of SCN Based on the G-BAPSO Algorithm

In order to illustrate the performance of SCN based on the G-BAPSO algorithm, four groups of numerical simulation experiments are used in this section. We include two groups of classification problems and two groups of regression problems: the problem of function regression, synchronous motor excitation current, identification of bank notes, and the iris data set.

4.2.1. Dataset Information

Dataset 1 is generated by function $f(x) = 0.2e^{-(10x-4)^2} + 0.5e^{-(80x-40)^2} + 0.3e^{-(80x-20)^2}$, $x \in [0, 1]$, 1000 points are randomly collected as the training set, and the test set is 300 equidistant points on the interval $[0, 1]$ [9].

Dataset 2 is the current data (SM) [37,38] in the UCI database. The task is to find the nonlinear relationship between synchronous motor excitation current and the input
features including power factor, load current, synchronous motor excitation current change, and power factor error. There are 557 groups of samples, where 400 samples are used as the training set and 157 samples are used as the test set.

Dataset 3 is the Banknote authentication dataset in the UCI database, which is extracted from images of samples of real and counterfeit bills. The task is to verify the authenticity of bills (classification). The data contains 4 input features (the variance, the skewness, the curtosis, and the entropy of the image), and the output corresponding to each group of input features is the banknote category (true or false). The dataset has 1372 samples, where 1038 samples are used for training and 334 samples are used for testing.

Dataset 4 is the Iris dataset in the UCI database. There are three categories of Iris, and the task is to identify the iris category based on four input attributes including the length and width of calyx, and the length and width of the petal. The dataset contains 150 groups of samples, of which each category contains 50 groups of samples. where 120 groups of samples are used for training and 30 groups of samples are used for testing.

Table 6 shows the relevant information of the four datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Samples</th>
<th>Input Variables</th>
<th>Output Variables</th>
<th>Training Samples</th>
<th>Test Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1300</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>557</td>
<td>4</td>
<td>1</td>
<td>400</td>
<td>157</td>
</tr>
<tr>
<td>3</td>
<td>1372</td>
<td>4</td>
<td>2</td>
<td>1038</td>
<td>334</td>
</tr>
<tr>
<td>4</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>120</td>
<td>30</td>
</tr>
</tbody>
</table>

4.2.2. Evaluate Metrics and Parameter Settings

Select the sigmoid function as the activation function, as shown in Formula (33). For regression problems, Root mean square error (RMSE) is used as the evaluation index, as shown in Formula (34). The smaller RMSE, the better the network performance. For classification problems, the accuracy rate (ACC) is taken as an evaluation index, as shown in Formula (35). The closer ACC is to 1, the better the network performance is.

\[
\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{33}
\]

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y_k - o_k)^2} \tag{34}
\]

\[
\text{ACC} = \frac{\text{AcNum}_i}{\text{TotalNum}_i} \tag{35}
\]

where \( y_k \) stands for the ideal output of the \( k \)-th sample, \( o_k \) stands for the network output of the \( k \)-th sample. \( \text{AcNum}_i \) stands for the number of correctly identified samples of class \( i \), and \( \text{TotalNum}_i \) stands for the total sample number of class \( i \).

Based on the above four datasets, G-BAPSO-SCN is compared with SCN [8], RVFL network [7] and ELM [39]. The parameters are shown in Table 7.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>The Hidden Nodes</th>
<th>Average Runs</th>
<th>Popsize</th>
<th>Iterations</th>
<th>Search Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>[−300, 300]</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>[−0.3, 0.3]</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>[−5, 5]</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>[−5, 5]</td>
</tr>
</tbody>
</table>
4.2.3. The Experimental Results

Due to the influence of random parameters, each group of experiments was run independently for 50 times, and the average results were finally used for the comparative analysis. The convergence curves of the average results of G-BAPSO-SCN and SCN in 50 experiments are drawn in Figures 7–10, respectively, and the average running results are shown in Tables 8–11.

Figure 7. The average convergence curve (Dataset 1).

Figure 8. The average convergence curve (Dataset 2).

Figure 9. The average convergence curve (Dataset 3).
Figure 10. The average convergence curve (Dataset 4).

Table 8. Performance comparison results (Dataset 1).

<table>
<thead>
<tr>
<th>Network</th>
<th>Training RMSE</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BAPSO-SCN</td>
<td>0.0008</td>
<td>0.0021</td>
</tr>
<tr>
<td>SCN</td>
<td>0.0040</td>
<td>0.0050</td>
</tr>
<tr>
<td>RVFL</td>
<td>0.1491</td>
<td>0.1645</td>
</tr>
<tr>
<td>ELM</td>
<td>0.2056</td>
<td>0.2235</td>
</tr>
</tbody>
</table>

Table 9. Performance comparison results (Dataset 2).

<table>
<thead>
<tr>
<th>Network</th>
<th>Training RMSE</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BAPSO-SCN</td>
<td>$6.4809 \times 10^{-6}$</td>
<td>$4.0114 \times 10^{-5}$</td>
</tr>
<tr>
<td>SCN</td>
<td>$6.2139 \times 10^{-5}$</td>
<td>$5.7800 \times 10^{-4}$</td>
</tr>
<tr>
<td>RVFL</td>
<td>$2.0778 \times 10^{-2}$</td>
<td>$1.0210 \times 10^{-1}$</td>
</tr>
<tr>
<td>ELM</td>
<td>$2.2864 \times 10^{-1}$</td>
<td>$2.7515 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

Table 10. Performance comparison results (Dataset 3).

<table>
<thead>
<tr>
<th>Network</th>
<th>Training ACC</th>
<th>Test ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BAPSO-SCN</td>
<td>99.93%</td>
<td>99.96%</td>
</tr>
<tr>
<td>SCN</td>
<td>99.79%</td>
<td>99.80%</td>
</tr>
<tr>
<td>RVFL</td>
<td>97.88%</td>
<td>97.90%</td>
</tr>
<tr>
<td>ELM</td>
<td>99.04%</td>
<td>98.50%</td>
</tr>
</tbody>
</table>

Table 11. Performance comparison results (Dataset 4).

<table>
<thead>
<tr>
<th>Network</th>
<th>Training ACC</th>
<th>Test ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BAPSO-SCN</td>
<td>98.58%</td>
<td>94.20%</td>
</tr>
<tr>
<td>SCN</td>
<td>98.58%</td>
<td>91.27%</td>
</tr>
<tr>
<td>RVFL</td>
<td>82.50%</td>
<td>90.00%</td>
</tr>
<tr>
<td>ELM</td>
<td>98.33%</td>
<td>86.67%</td>
</tr>
</tbody>
</table>

5. Discussion

Tables 8–11 show that G-BAPSO-SCN converges faster than other models when the number of hidden nodes is the same, and the RVFL network and ELM have relatively poor performance. In regression experiment 1, the training RMSE of G-BAPSO-SCN increased by $5.57 \times 10^{-5}$ compared with that of SCN; in regression experiment 2, the training RMSE of G-BAPSO-SCN increased by 0.0032, indicating that the improvement degree of training RMSE not only depends on the algorithm itself, but is also closely related to the distribution
characteristics of the data. In classification problems, G-BAPSO-SCN also has an advantage in recognition accuracy, with an average improvement of about 0.07% over SCN. At the same time, it could also be seen from the comparison of convergence curves in Figures 7–10 that G-BAPSO-SCN has faster convergence than SCN. Since RVFL and ELM are not incremental generation networks, and their hidden nodes are generated once, the convergence speed between them is not compared with the former two curves.

G-BAPSO-SCN performs well because it introduces a more flexible mechanism in the search of input parameters, allowing it to explore the solution space more thoroughly. The algorithm combines the advantages of the intelligent optimization method and traditional optimization method, and the introduction of gradient makes it possible to approach the optimal solution quickly. By introducing the emission frequency mechanism of bats into the PSO framework, local search could be carried out according to individual fitness and emission frequency at the initial stage, and the poor individuals could be eliminated adaptively, thus accelerating the convergence rate. SCN uses the linear expansion searching method to select the input parameters, which cannot detect the optimal solution with a long distance in the initial stage. The initial configuration is not optimized, thus increasing the searching iterations and the number of nodes to a certain extent. RVFLN is a typical incremental network, so we also compare it with G-BAPSO-SCN. However, RVFLN could not guarantee the convergence, and the accuracy of the network is slightly inferior. ELM's performance is relatively poor under the same number of hidden nodes as G-BAPSO-SCN. In addition to the optimization of input weights and biases, CSSA-SCN [20] optimizes their search intervals and the regularization factor r of the model, which is the optimization of hyperparameters, and its objective function and optimized parameters are different from those in this paper. Most of the other methods used to optimize SCNS focus on the the output parameters or the generation of model structures.

In conclusion, G-BAPSO-SCN, first proposed in this paper, has a faster convergence rate and a more optimized structure, which can improve the training efficiency of the model and have better performance in the application of regression or prediction.

6. Limitations and Future Works

Table 12 presents the comparison results of the running time of four models based on Dataset 1. It can be seen that G-BAPSO-SCN takes a relatively long time, which is determined by two factors: On the one hand, both G-BAPSO-SCN and SCN are incremental generation models, and the model structure needs to be determined adaptively in the training process, so both of them take a longer time than RVFL and ELM. On the other hand, G-BAPSO-SCN has the search process of the optimal solution based on the hybrid algorithm, and the gradient calculation process obviously increases the time cost. Therefore, in terms of time cost, G-BAPSO-SCN cannot save the training time, which is one of its limitations. However, G-BAPSO-SCN increases the rate of convergence of the model, which is very meaningful for practical problems. Meanwhile, the G-BAPSO algorithm uses some empirical values to initialize the value range (the domain of the objective function) of the input weights and biases, which should be studied and optimized in the future. In addition, other intelligent optimization algorithms can also be used to optimize the input parameters of SCN. The genetic algorithm is also an effective algorithm with fast convergence rate [40], which can be used in the subsequent research.

### Table 12. The comparison of run time based on Dataset 1.

<table>
<thead>
<tr>
<th>Network</th>
<th>Run Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BAPSO-SCN</td>
<td>1.8874</td>
</tr>
<tr>
<td>SCN</td>
<td>1.6420</td>
</tr>
<tr>
<td>RVFL</td>
<td>0.0063</td>
</tr>
<tr>
<td>ELM</td>
<td>0.0153</td>
</tr>
</tbody>
</table>
7. Conclusions

The proposed algorithm G-BAPSO provides an alternative solution for improving the convergence speed of SCN. The G-BAPSO algorithm focuses on the search for input weights and biases, considering the gradient information, the global optimal solution, and the individual optimal solution. In the process of searching for the optimal solution, the algorithm adaptively adjusts the local solutions according to the individual emission frequency and fitness in the initial stage. Thus, the convergence speed is improved by combining the traditional optimization algorithm with the intelligent optimization algorithm. At the same time, the algorithm strictly follows the inequality constraint mechanism of SCN, and each optimal solution is generated under the inequality supervision mechanism, which ensures the unique convergence of the SCN network. Numerical experiments also illustrate that SCN based on the G-BAPSO algorithm could be used to solve regression or classification problems. Compared with SCN, the training RMSE of G-BAPSO-SCN increased by $5.57 \times 10^{-5}$ and $3.2 \times 10^{-3}$, respectively, and the recognition accuracy of G-BAPSO-SCN increased by 0.07% on average. In industrial applications, the G-BAPSO-SCN proposed in this paper is suitable for nonlinear approximation, regression prediction, and pattern recognition, based on industrial big data.

Author Contributions: Conceptualization, J.L.; methodology, J.L. and Y.L.; software, J.L. and Q.Z.; writing—original draft preparation, J.L.; writing—review and editing, Y.L. All authors have read and agreed to the final version of the manuscript.

Funding: This paper is partly supported by National Science Foundation of China under Grants (62073226), Liaoning Province Natural Science Foundation (2020-KF-11-09, 2022-KF-11-01), Shen-Fu Demonstration Zone Science and Technology Plan Project (2020H13, 2021H07), Major Science and Technology Projects of Liaoning Province (2022H1/10400033), Young Teacher Foundation of Shenyang Institute of Technology (QN202210).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data included in this paper are available at http://archive.ics.uci.edu/ml/index.php (accessed on 19 October 2022).

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

References
22. Zhang, Q.; Li, W.; Li, H. Self-blast state detection of glass insulators based on stochastic configuration networks and a feedback transfer learning mechanism. *Inf. Sci.* 2020, 522, 259–274. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.