Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 7672847, 9 pages
https://doi.org/10.1155/2020/7672847

Research Article

Bus Arrival Time Prediction Using Wavelet Neural Network Trained by Improved Particle Swarm Optimization

Yuanwen Lai,[1] **Said Easa**,[1,2] **Dazu Sun**,[1,3] **and Yian Wei**[1]

[1]*College of Civil Engineering, Fuzhou University, Fuzhou 350116, China*
[2]*Department of Civil Engineering, Ryerson University, Toronto, ON M5B2K3, Canada*
[3]*Wenzhou Transportation Planning and Design Institute, Wenzhou 325000, China*

Correspondence should be addressed to Yuanwen Lai; laiyuanwen@fzu.edu.cn

Received 2 August 2019; Revised 13 November 2019; Accepted 20 December 2019; Published 13 January 2020

Academic Editor: Gonçalo Homem de Almeida Correia

Prediction of bus arrival time is an important part of intelligent transportation systems. Accurate prediction can help passengers make travel plans and improve travel efficiency. Given the nonlinearity, randomness, and complexity of bus arrival time, this paper proposes the use of a wavelet neural network (WNN) model with an improved particle swarm optimization algorithm (IPSO) that replaces the gradient descent method. The proposed IPSO-WNN model overcomes the limitations of the gradient-based WNN which can easily produce local optimum solutions and stop the training process and thus improves prediction accuracy. Application of the model is illustrated using operational data of an actual bus line. The results show that the proposed model is capable of accurately predicting bus arrival time, where the root-mean square error and the maximum relative error were reduced by 42% and 49%, respectively.

1. Introduction

In recent years, with the accelerated pace of China's urbanization process, urban transport problems have become increasingly prominent. Public transport is widely regarded as the best choice to solve the traffic problems and improve the urban environment [1]. The Chinese government proposed "*Give Priority to the Development of Urban Public Transport*" policy in 2004 and released "*Give Priority to the Development of Urban Public Traffic Guidance*" in 2012. Advanced urban public transport systems are under construction and will continue to improve. Bus arrival time prediction is the core content of such systems for bus travel information and bus travel-route guidance. It is an important part of the urban public transport system.

At present, there are many models for predicting bus arrival time of public transit, such as nonparametric regression models, support-vector machine (SVM) models, Kalman filters, artificial neural network (ANN) models, and hybrid models. Lin et al. [2] used the historical data mean method to predict the average bus arrival time delay. Patnaik et al. [3] used automatic passenger counts of bus data to establish a prediction model of multivariable regression. Sun et al. [4] proposed a model to predict the arrival time using the weighted mean of historical data and real-time global positioning system (GPS) data. Padmanaban et al. [5] proposed an arrival time prediction model that is based on real-time bus data and bus operation delay. Xue et al. [6] developed a mathematical model based on the analysis of the process of bus operation and bus station characteristics.

He et al. [7] proposed a new bus arrival time prediction model with multi-index evaluation which is based on SVM and verified its feasibility. Yu et al. [8] developed an SVM prediction model considering the time period and segment, weather, and operation time of current and downstream sections. Li [9] developed a prediction model for road-section operation time based on real-time correction of bus speed. Zuo and Wang [10] developed a finite-state machine forecasting model based on real-time GPS data. Shalaby et al. [11] used a Kalman filter to predict bus running time based on GPS data. Chien and Kuchipudi [12] developed a Kalman filter to predict the arrival time at bus station based on road and stop characteristics. Vanajakshi et al. [13] proposed the use of automatic vehicle location (AVL) data and Kalman filter to predict bus arrival time in mixed traffic environments, where model parameters were adjusted in real time according to the prediction error.

Park and Rilett [14] argued that the ANN model can provide better prediction performance than the Kalman filter. Chien et al. [15] proposed an adaptive feedback ANN model based on the operation time of arterial segment and stop station. The model can automatically adjust the parameters according to the real-time prediction error. Lin et al. [16] proposed a two-layer ANN model that considered the effect of time and intersection signal lights, but this model required a large amount of training data. The effect of different weather conditions on bus travel time was analyzed by Bladikas et al. [17].

Hybrid models have also been developed for the analysis of bus operation. Ran [18] proposed a hybrid model that combined multivariable regression and ANN based on bus real-time AVL data. Liu [19] proposed a hybrid prediction model, based on a Kalman filter and ANN, that effectively combined historical and real-time data. Among the existing techniques, ANN has the characteristic of nonlinear adaptive information processing, which provides a great advantage in prediction. In particular, a wavelet neural network (WNN) that combines ANN and wavelet analysis exhibits good time-frequency localization characteristics and neural network self-learning function. Therefore, WNN has strong abilities of recognition, fault tolerance, and accurate prediction of bus arrival time. However, the traditional WNN has used the gradient descent learning method to correct the weighting parameters, which result in slow training speed and the possibility of being trapped into a local optimum solution.

To address the preceding issues, this paper proposes a hybrid model of bus arrival time prediction that combines WNN and an improved particle swarm optimization (IPSO) algorithm. The next sections present the IPSO algorithm, the proposed IPSO-WNN model, and its implementation for bus arrival time prediction. Application of the model to an actual case study is then presented, followed by the conclusions.

2. Improved Particle Swarm Optimization Algorithm

2.1. Traditional Particle Swarm Optimization. The traditional particle swarm optimization (PSO) is a stochastic computational intelligent method that has a simple structure, where a few parameters need to be adjusted [1]. Similar to other evolutionary algorithms, PSO is initialized with random particles (potential solutions). However, in PSO, each particle is assigned a random velocity and then flies in the N-dimensional space, where its velocity is dynamically adjusted according to the flying experiences of other particles in the group and its own experience. Subsequently, through an iterative update of the position and speed of the particles, the optimal solution is found. The sketch map of the PSO algorithm is shown in Figure 1.

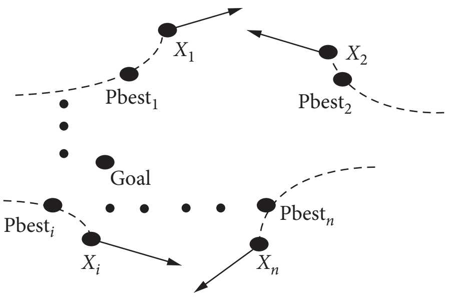

Figure 1: Sketch map of the particle swarm optimization algorithm.

Let the position and speed of particle i of the population in the N-dimensional solution space be expressed as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{iN})$, respectively. Then, the speed and position of particle i are updated as follows:

$$V_i = \omega V_i + C_1 \cdot \text{rand}() \cdot (\text{Pbest}_i - X_i) + C_2 \cdot \text{rand}() \cdot (\text{Nbest}_i - X_i), \tag{1}$$

$$X_i = X_i + V_i, \tag{2}$$

where V_i = speed of particle i, ω = inertia weight, C_1 and C_2 = learning factors, which refer to the acceleration weight of particles that fly to individual and group extremums, respectively, rand () = random number between 0 and 1, Pbest_i = position of the optimal solution that particle i has found so far (personal best), X_i = position of particle i, and Nbest_i = position of the optimal solution that the neighborhood of particle i has found so far (global best).

Appropriate values of C_1 and C_2 can accelerate the convergence and avoid falling into a local optimum, where a larger $V_{\max}$ can guarantee the global search ability of the particle population. The coefficients ω, C_1, and C_2 determine the capacity of the space search of the particle. The preceding PSO is a standard algorithm and is the basis for the current research to improve the algorithm.

2.2. PSO Algorithm Improvements. In PSO, based on the experiences of the group and the particle's own experiences, a particle flies to the best particle that has a strong global search ability and a better solution area. However, in the process of the optimization of complex high-dimensional problems, the traditional PSO algorithm has a more global ability at the start and a more local ability at the end of the process. Therefore, PSO is more likely to explore local optimum solutions at the end. In addition, the search performance of the algorithm depends on the values of the parameters. To address these limitations, two improvements to the traditional algorithm were adopted: (1) improving subgroup strategy and (2) updating particle velocity and learning factors.

For the subgroup strategy improvement, let the total number of particles N be divided into M subgroups that are multiples (that is, N is a multiple of$M$). Initialize the particle swarm, calculate the fitness value of each particle, and sort the particles according to their fitness values from large to small, where the sorted particle numbers are 1, 2, . . ., N. At each interval $i = N/M$, extract the particle subgroups in turn. The particles contained in subgroup j are $\{j/j = j + i \times k\}$. This process can effectively avoid uneven grouping of the subgroups. In addition, the better particles can drive the bad particles in all groups, resulting in a balanced evolution of each subgroup [8].

For the improvement related to updating particle velocity and learning factors, the particle velocity updating of equation (1) is revised as follows:

$$\begin{aligned} V_i = \omega V_i + C_1 \cdot \text{rand}() \cdot (\text{Pbest}_i - X_i) + C_2 \cdot \text{rand}() \\ \cdot (\text{Nbest}_i - X_i) + C_3 \cdot \text{rand}() \cdot (\text{NLbest}_i - X_i), \end{aligned} \tag{3}$$

where C_3 = learning factor and NLbest_i = position of the optimal solution that the subgroup particles have found so far. Then, the position of particle i, X_i, is updated using equation (2). The learning factors are given by

$$\begin{aligned} C_1 &= C_{1s} - \frac{t \times (C_{1s} - C_{1e})}{T_{\max}}, \\ C_2 &= C_{2s} - \frac{t \times (C_{2s} - C_{2e})}{T_{\max}}, \\ C_3 &= C_{3s} - \frac{t \times (C_{3s} - C_{3e})}{T_{\max}}, \end{aligned} \tag{4}$$

where C_{1s}, C_{2s}, and C_{3s} = corresponding values of C_1, C_2, and C_3 at the start of the algorithm and C_{1e}, C_{2e}, and C_{3e} = corresponding values of C_1, C_2, and C_3 at the end of the algorithm.

At the start of the algorithm, the value of C_1 is larger and the values of C_2 and C_3 are smaller. This is advantageous to the search of the particles in the whole space and provides a stronger global searching ability. At the end of the algorithm, C_1 becomes smaller and C_2 and C_3 become larger and this helps the particles to have a strong local searching ability and in turn finds the global optimal solution.

The process of improving the PSO algorithm is shown in Figure 2. The specific implementation steps are as follows:

Step 1: initialize the particle swarm. The position and velocity of the initial particles are randomly generated within the specified range, and the Pbest_i coordinates of each particle are set to their current positions. The optimal particle for each subgroup is the best individual value of the subgroup in which the particle is located, and NLbest_i is set to the current position of the optimal particle. The optimal particle of the entire neighborhood is the best individual of the optimal particles in each subgroup, and Nbest_i is set to the current position of the optimal particle.

Step 2: calculate the fitness value of the particle. For each particle, the current fitness is compared to the fitness of the best position, Pbest_i, that it has experienced. If it is better than the previous value, the function value of Pbest_i is updated; otherwise, it remains unchanged. The fitness of each particle of this iteration is compared to the fitness of NLbest_i experienced by the subgroup in which it is located. If it is better than the previous value, the function value of NLbest_i is updated; otherwise, it remains unchanged. The fitness of each particle in this iteration is compared to the fitness of the best Nbest_i experienced by the whole group. If it is better than the previous value, the function value of Nbest_i is updated; otherwise, it remains unchanged.

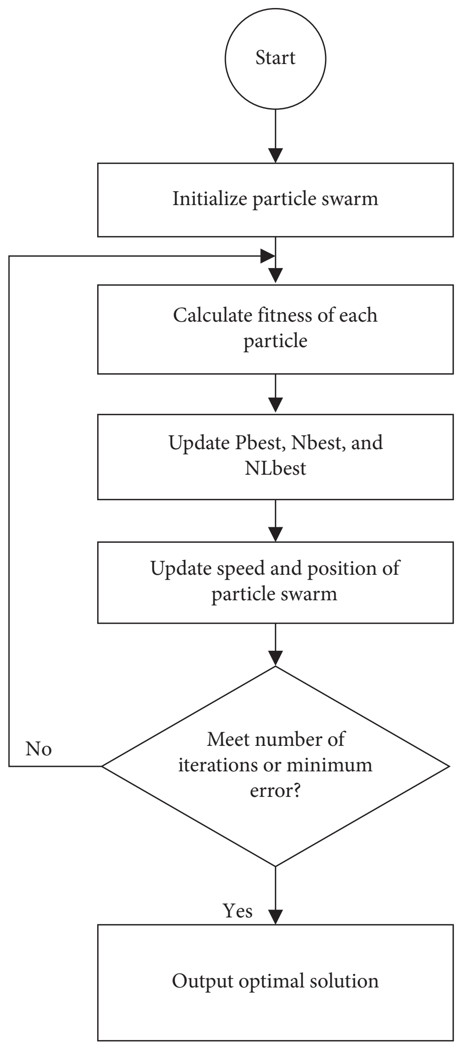
Start
Initialize particle swarm
Calculate fitness of each particle
Update Pbest, Nbest, and NLbest
Update speed and position of particle swarm
Meet number of iterations or minimum error?
No
Yes
Output optimal solution

Figure 2: Particle swarm optimization process.

Step 3: update particle speed and position. The speed and position of each particle are updated according to equations (2) and (3).

Step 4: check whether the end condition is met. When the maximum number of iterations is reached or the minimum error is satisfied, the optimal solution is output; otherwise, return to Step 2.

3. Proposed IPSO-WNN Model

As previously mentioned, the proposed model of bus arrival time prediction combines the improved PSO with WNN and is called IPSO-WNN. A description of the WNN technique and the IPSO-WNN model is presented in this section.

3.1. Wavelet Neural Network. The WNN is a mathematical model that combines wavelet analysis and neural network. It is based on the topology of the backpropagation (BP) neural network and the wavelet basis function as the transfer function of the hidden layer nodes, instead of the original sigmoid function. In other words, the wavelet function is introduced as the transfer function of the BP network. A transfer function of WNN is used in the shift and scaling factors, allowing a stronger ability for recognition, fault tolerance, and prediction. The WNN structure is shown in Figure 3.

Given the input sample data X_i $(i = 1,\ 2,\ \ldots,\ k)$, the mathematical expression of the hidden layer output is expressed as

$$h(j) = h_j\left[\frac{\sum_{i=1}^{k} w_{ij}x_i - b_j}{a_j}\right], \quad j = 1, 2, \ldots, l, \tag{5}$$

where $h(j) =$ output value of the j node in the hidden layer, $h_j =$ wavelet basis function, $w_{ij} =$ linked weights between the input and hidden layers, $b_j =$ shift factor of the wavelet basis function, $a_j =$ scaling factor of the wavelet basis function, and l = number of nodes in the hidden layer.

The mathematical expression of the output layer is given by

$$y(k) = \sum_{i=1}^{l} w_{jk} h(j), \quad k = 1, 2, \ldots, m, \tag{6}$$

where $y(k) = k$-value of the output layer, $w_{jk} =$ weight between the hidden layer j and the output layer k, and m = number of nodes of the output layer.

The method of modifying the weights and thresholds of the traditional WNN is similar to that of the correcting algorithm for BP neural network weights. Using the gradient correction method to constantly correct network weights and thresholds of the wavelet basis function can reduce the gap between the expected and predicted outputs. When the error reaches a specified limit, the correction can stop. The WNN correction process involves two steps, as follows:

Step 1: calculate network prediction error:

$$e = \sum_{k=1}^{m} \left(y_n(k) - y(k)\right), \tag{7}$$

where $e =$ prediction error of WNN and $y_n(k) =$ expected output value of k.

Step 2: correct the weights of WNN and the coefficients of the wavelet according to the network prediction error e, as follows:

$$\begin{aligned} w_{n,k}^{(i+1)} &= w_{n,k}^{i} + \Delta w_{n,k}^{(i+1)}, \\ a_{k}^{(i+1)} &= a_{n,k}^{i} + \Delta a_{k}^{(i+1)}, \\ b_{k}^{(i+1)} &= b_{n,k}^{i} + \Delta b_{k}^{(i+1)}, \end{aligned} \tag{8}$$

where $\Delta w_{n,k}^{(i+1)}$, $\Delta a_{k}^{(i+1)}$, and $\Delta b_{k}^{(i+1)}$ are calculated based on the network prediction error as follows:

$$\begin{aligned} \Delta w_{n,k}^{(i+1)} &= -\eta \frac{\partial e}{\partial w_{n,k}^{(i)}}, \\ \Delta a_{k}^{(i+1)} &= -\eta \frac{\partial e}{\partial a_{k}^{(i)}}, \\ \Delta b_{k}^{(i+1)} &= -\eta \frac{\partial e}{\partial b_{k}^{(i)}}, \end{aligned} \tag{9}$$

where η is the learning rate.

3.2. Procedures of the IPSO-WNN Model. The fitness function, which indicates the accuracy of the neural network, is used to evaluate the quality of each particle. The following training error (mean squared deviation) of WNN is chosen as the fitness function of PSO:

$$f = \mathrm{MSE}(N) = \frac{1}{N}\sum_{k=1}^{N}\left[y_n(k) - y(k)\right]^2, \tag{10}$$

where $N =$ number of training samples and $y_n(k)$ and $y(k) =$ expected and actual output values of k, respectively. The optimization specific steps are as follows:

(1) Data normalization: normalize the sample data for input and output to produce dimensionless quantities.

(2) Parameter initialization: initialize the parameters of WNN, including PSO parameters, such as particle swarm iterations, population size, location, and maximum speed.

(3) Population initialization: randomly initialize the position and velocity of the particle and calculate the initial fitness values according to the fitness function.

(4) Finding initial extremum: determine individual and group extremums according to the initial particle fitness values.

(5) Iterative optimization: use the PSO algorithm to update the position and velocity of the particle according to the fitness value of the new updated individual and group extremums. When the fitness value converges or the specified number of iterations is reached, go to Step 6.

(6) Output optimal weights and thresholds: set the position of the global optimal particle as the optimal weights and WNN thresholds.

(7) Prediction of WNN: use the optimal weights and thresholds to predict the new samples.

4. Implementing IPSO-WNN Model for Bus Arrival Time Prediction

Using the improved PSO algorithm, the WNN model was optimized and the bus arrival time prediction model was

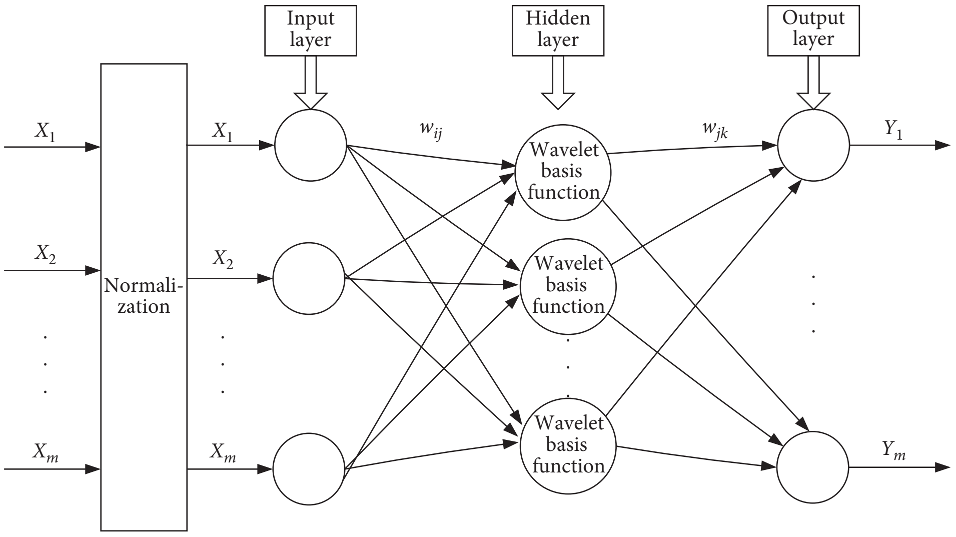

Figure 3: Topology of the wavelet neural network.

coded using Matlab. Details on preparing input data, input data processing, and transfer function and determining number of hidden layer nodes are described in this section.

4.1. Preparing Input Data. The input data to the proposed model are determined based on relevant literature [20–22] and the historical data on weather, date, time, and bus real-time operation. The bus arrival time at the next stop was selected as the output target. The input data include sample data vector, training dataset, weather factors, date factors, and time factors.

4.1.1. Sample Data Vector. This vector includes the following nine input variables:

$$h = \left(t_{b1}, t_{b2}, t_{b3}, t_{h1}, t_{h2}, t_{h3}, w, d, s\right)^T, \tag{11}$$

where t_{bi} = travel time of the three buses ahead of the bus under consideration from stop $(k-1)$ to stop k in the same time period of the day, where $i = 1, 2, 3$, and t_{hi} = travel time of buses whose departure times are in the same period in the previous three weeks from stop $(k-1)$ to stop k, w = weather conditions, d = date factor, and s = period factor.

4.1.2. Training Dataset. The training dataset is $D = (h_\iota, t_{r\iota})$, where $\iota = 1, 2, \ldots, n$, and n is number of training samples. The variable t_{ri} is actual operation time of the current bus from stop $(k-1)$ to stop k.

4.1.3. Weather Factors. Weather conditions of one day can be expressed as $w = \{0, 1, 2\}$, in which 0 means rainy day, 1 means sunny day, and 2 means other weather conditions.

4.1.4. Date Factors. Bus arrival time varies not only between working days and weekend, but also among the seven days of the week. The seven days of the week are expressed as $d = \{1, 2, 3, 4, 5, 6, 7\}$.

4.1.5. Time Factors. The study period of the day (5 : 00–23 : 00) was divided into seven different time periods, expressed as $s = 1, 2, \ldots, 7$, where 1 represents 5 : 00–7 : 00, 2 represents 7 : 00–9 : 00, 3 represents 9 : 00–11 : 30, 4 represents 11 : 30–14 : 30, 5 represents 14 : 30–17 : 00, 6 represents 7 : 00–8 : 00, and 7 represents 19 : 00–23 : 00.

4.2. Input Data Processing and Transfer Function. The normalized function *mapminmax* of Matlab used in this study is given by

$$x_k = \frac{\left(y_{\max} - y_{\min}\right)\left(x_k - x_{\min}\right)}{\left(x_{\max} - x_{\min}\right) + y_{\min}}, \tag{12}$$

where x_k = normalized data, $y_{\max} = 1$, $y_{\min} = -1$, and $x_{\max}$ and $x_{\min}$ = maximum and minimum values of the samples, respectively.

For the transfer function, in practice, the *Morlet* wavelet function is widely used and has achieved good results. This function is a single frequency complex *sine* function with the Gauss network, given by

$$y = \cos(1.75x)e^{-x^2/2}. \tag{13}$$

4.3. Determining Number of Hidden Layer Nodes. The structure of the neural network is composed of input layer, hidden layer, and output layer. The number of input layer nodes according to the preceding analysis was identified as 9. The output layer represents bus arrival time as the output value, and therefore, this layer has only one node. The optimum number of nodes in the hidden layer require many iterations during the training process. The reference formula for selecting the optimal number of nodes in the hidden layer is given by [23]

$$l < \sqrt{(m+n)} + a, \tag{14}$$

where l = optimum number of nodes in the hidden layer, m = number of nodes in the output layer, n = number of nodes in the input layer, and a = constant (0 to 10).

According to equation (14), the number of hidden layer nodes should be between 3 and 13. The idea of implicit node optimization is as follows. When the network training is not sufficient, the number of hidden layer nodes is increased, and the training and prediction errors will be reduced. If the number of hidden layer nodes continues to increase, the prediction error will increase. Therefore, this idea can be used to determine whether the number of nodes of the hidden layer is appropriate.

The training error of the sample represents the error obtained when the data of the training set are taken as the input sample. The prediction error of the sample is the error obtained when the test dataset is taken as the input sample. At the same time, an expected error E can be set, where E is a constant with a threshold ranging from 0 to 1 (0.5 was selected in this study).

The training error of the wavelet neural network when the number of hidden layer nodes is M is expressed as ${\varepsilon_{\text{train}}}^{M}$ and the prediction error of the wavelet neural network when the number of hidden layer nodes is M is expressed as ${\varepsilon_{\text{predict}}}^{M}$. When the number of hidden layer nodes is $(M-1)$, the training and prediction errors are expressed as ${\varepsilon_{\text{train}}}^{M-1}$ and ${\varepsilon_{\text{predict}}}^{M-1}$, respectively. When the number of hidden layer nodes is $(M+1)$, the training and prediction error are expressed as ${\varepsilon_{\text{train}}}^{M+1}$ and ${\varepsilon_{\text{predict}}}^{M+1}$, respectively.

Finally, whether M is the best hidden layer node is determined according to the following formulas:

$$\theta_1(M) = \frac{\varepsilon_{\text{train}}(M)}{E}, \tag{15}$$

$$\theta_1(M+1) = \frac{\varepsilon_{\text{train}}(M+1)}{E}, \tag{16}$$

$$\theta_1(M-1) = \frac{\varepsilon_{\text{train}}(M-1)}{E}, \tag{17}$$

$$\theta_2(M) = \frac{\varepsilon_{\text{predict}}(M)}{E}, \tag{18}$$

$$\theta_2(M+1) = \frac{\varepsilon_{\text{predict}}(M+1)}{E}, \tag{19}$$

$$\theta_2(M-1) = \frac{\varepsilon_{\text{predict}}(M-1)}{E}. \tag{20}$$

To determine the number of nodes in the optimal hidden layer, the process is as follows. When $\theta_2(M-1) > \theta_2(M) > \theta_2(M+1)$, the number of nodes in the current hidden layer is less than the number of nodes in the optimal hidden layer. Therefore, the number of nodes in the hidden layer should be increased. When $\theta_2(M-1) < \theta_2(M) < \theta_2(M+1)$, the number of nodes in the current hidden layer is larger than the number of nodes in the optimal hidden layer. Therefore, the number of nodes in the hidden layer should be decreased. When both conditions are simultaneously satisfied and $\theta_2(M-1) > \theta_2(M)$, the current node number M is the optimal hidden layer node number. After several iterations of training, the training and prediction errors are obtained and are substituted into equations (15)–(20) to obtain the optimal number of hidden layer nodes. This number was found to be 10. Therefore, the structure of the wavelet neural network is finally determined as 9-10-1; that is, the number of nodes in the input layer is 9, the number of nodes in the hidden layer is 10, and the number of nodes in the output layer is 1.

5. Case Study

The proposed model was applied to bus line 102 in Suzhou city to predict bus arrival times. This bus line runs from Baodai West Road to South Railway Station Square. The route is 12.8 km long and includes 20 bus stops. Bus operational data, which involved 1790 segments, were collected in 2019 during May 7–9, May 14–16, May 21–23, and May 27–30 (every week from Thursday to Saturday). The collected data were converted to 34,010 route segment operational times for each bus running along the route (1790×20). After data processing, a total of 472 sets of sample data were obtained, 382 groups were selected as the training data, and the remaining 90 groups were used as the test data. Part of the training sample data is shown in Table 1.

5.1. Training Samples

5.1. Training Samples. The following input data were assumed: maximum number of iterations of the algorithm = 500, population size = 50, accelerating factors $C_{1s} = 2.5$, $C_{2s} = 2.0$, $C_{3s} = 1.5$, $C_{1e} = 1.5$, $C_{2e} = 2.0$, and $C_{3e} = 2.5$. The output fitness curves of the WNN and IPSO-WNN models are shown in Figure 4. For the WNN model (Figure 4(a)), the early training speed was slow, and 500 iterations were needed to achieve convergence with an error of 0.05. On the contrary, the proposed IPSO-WNN model (Figure 4(b)) achieved rapid convergence in the training iterations number 0–50. When the number of iterations is about 310, the solution converged and the error precision reached 0.01. Compared to WNN, the proposed IPSO-WNN model has a faster convergence rate and a smaller convergence error.

5.2. Establishing Bus Arrival Time Prediction Model. As previously mentioned, PSO was used to replace the traditional descent gradient method. In the training process, the weights and thresholds of the WNN were optimized. Based on the optimization results, the parameters of the calibrated IPSO-WNN model were determined. Then, the prediction model of bus arrival time was established as follows:

$$y(k) = \sum_{i=1}^{l} w_{jk} h(j)\left[\cos(1.75x_i)e^{-x_i^2/2}\left(\frac{\sum_{i=1}^{k} w_{ij}x_i - b_j}{a_j}\right)\right]. \tag{21}$$

The calibrated parameter values of the IPSO-WNN model are presented in Table 2. Given the nine variables shown previously in equation (11), one can obtain 102 prediction values of bus arrival time.

Table 1: Part of the training sample data.

Departure time	t_{h1}	t_{h2}	t_{h3}	t_{b1}	t_{b2}	t_{b3}	w	d	s	t_r
7 : 01	393	87	392	391	427	480	1	5	2	382
7 : 09	393	106	388	382	391	427	1	5	2	437
7 : 15	397	125	334	437	382	391	1	5	2	453
7 : 22	220	124	314	453	437	382	1	5	2	334
7 : 29	217	90	261	334	453	437	1	5	2	332
7 : 35	243	74	260	332	334	453	1	5	2	307
7 : 42	190	76	232	307	332	334	1	5	2	184
7 : 49	190	172	242	184	307	332	1	5	2	169
7 : 56	186	87	269	169	184	307	1	5	2	142
8 : 01	184	132	230	142	169	184	1	5	2	151

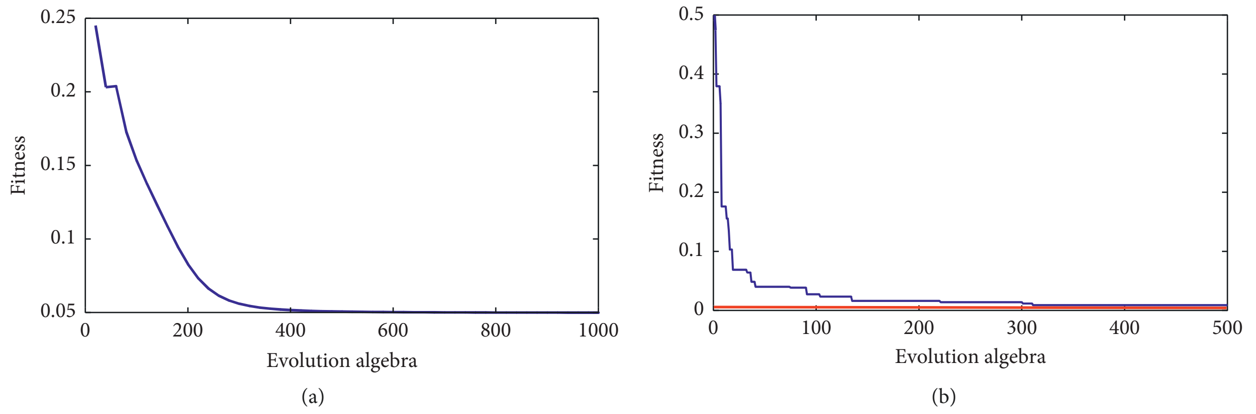

Figure 4: Fitness curves of WNN and IPSO-WNN models: (a) WNN model (maximum evolution = 1000); (b) IPSO-WNN model (maximum evolution = 500).

Table 2: Calibrated parameters of the IPSO-WNN model.

Weight	Calibrated parameters									
	1	2	3	4	5	6	7	8	9	10
	0.428	−0.289	−0.675	−2.689	0.435	−0.025	−0.018	0.458	0.432	−0.074
	1.384	−0.291	0.732	−0.620	−0.583	0.506	−0.507	0.061	−0.376	0.294
	0.178	1.382	−1.342	0.987	−0.173	−0.085	0.366	0.230	1.007	0.405
	0.568	0.187	−0.206	0.351	−2.020	−0.140	0.084	−0.569	−0.900	−0.990
w_{ij}	−0.221	0.601	−0.023	−0.246	−0.021	−0.667	−0.291	−0.388	0.018	0.032
	0.076	−0.239	0.386	−0.126	−0.726	−0.058	−0.974	−0.079	0.664	0.405
	−0.689	−1.068	1.010	−0.458	0.229	1.663	−1.788	0.315	0.737	−0.468
	0.081	−0.208	0.470	−0.180	−1.030	−0.085	−0.789	−0.086	0.932	0.363
	0.547	0.292	−0.193	0.326	−1.800	−0.119	0.124	−0.589	−0.852	−1.135
w_{jk}	0.285	0.271	−0.034	−0.475	−1.131	0.154	−0.381	0.136	0.001	−0.988
a_j	−1.602	−0.355	−0.694	0.223	−1.711	−0.004	−0.154	−1.092	−0.317	−0.832
b_j	0.403	−0.219	0.767	−1.867	−1.509	−1.011	0.489	0.491	−1.466	1.427

5.3. Model Validation. The 90 sets of the test samples were normalized, and then the optimal network weights and thresholds were used as input to the model. The prediction errors are shown in Figures 5 and 6. The root-mean square error, used to evaluate model accuracy, is given by the following equation:where RMSE = root-mean square error, T_i = actual value, and y_i = predicted value. The smaller the RMSE is, the higher the accuracy is. The RMSE of the WNN and IPSO-WNN models were 20.9% and 10.6%, respectively, indicating that the overall accuracy of the IPSO-WNN model is better. In addition, the results show that the relative error of bus arrival time prediction of the WNN model ranged from 5.2% to 16.8%, while that of the IPSO-WNN model ranged from 3.5% to 9.8%. Thus, the proposed model reduced the maximum relative error of the WNN model by 42% and the RMSE by 49%. Clearly, the proposed IPSO-WNN model has obvious advantages in bus arrival time prediction.

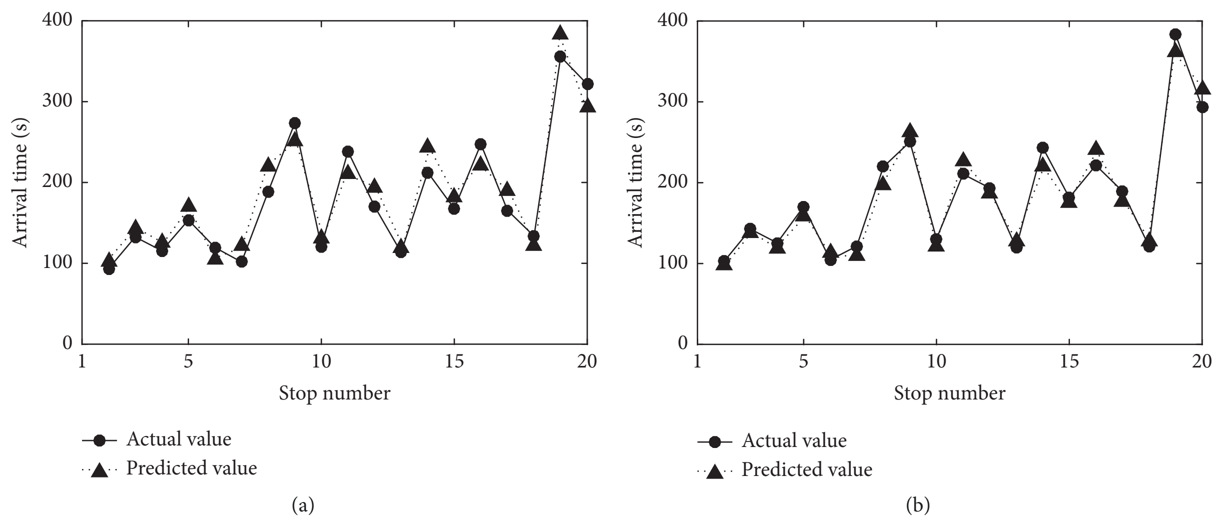

FIGURE 5: Comparison of predicted bus arrival times of WNN and IPSO-WNN models with actual values: (a) WNN model; (b) IPSO-WNN model.

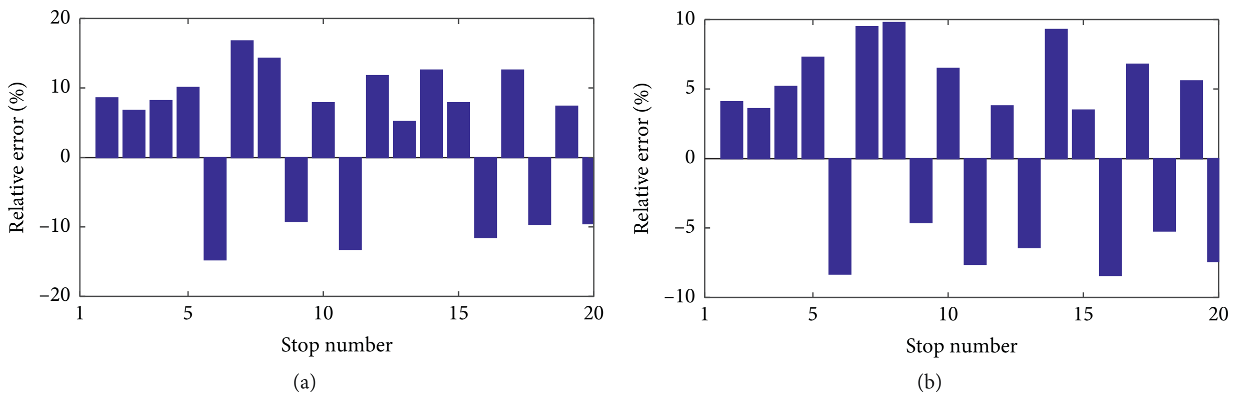

FIGURE 6: Relative error of predicted bus arrival time by WNN and IPSO-WNN models: (a) WNN model; (b) IPSO-WNN model.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_i - y_i)^2} \times 100\%, \tag{22}$$

6. Conclusions

This paper has presented an improved particle swarm optimization model that was integrated with a wavelet neural network to predict bus arrival time, and a new IPSO-WNN model was developed. The improvements to the PSO algorithm, which were intended to help find the optimal solution quickly and avoid local optimum solutions, were related to improving subgroup strategy and updating of particle velocity and learning factors. The IPSO algorithm overcomes the limitations of the traditional PSO. The IPSO-WNN model was applied for the prediction of bus arrival time in an actual bus line. Actual bus operational data were used for model training, and the proposed model was run to obtain the best network weights and thresholds. The application results show that the RMSE of the IPSO-WNN model was 10.6% compared to 20.9% for the traditional WNN. This study has focused on the prediction of bus arrival time using the developed IPSO-WNN model. Future research will continue to improve the PSO algorithm and train it with other common traffic datasets and compare it with other methods.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was financially supported by the Science and Technology Fund of Education Department of Fujian Province (JAT160079). The assistance of Shutian Xu, Ronglin Su, and Lilin Huang in data collection and analysis is gratefully acknowledged.

References

[1] H. Y. Xiang and X. W. Peng, "Current study and development trend of bus arrival time prediction," *Journal of Transport Information and Safety*, vol. 32, no. 4, pp. 57–61, 2014.

[2] W.-H. Lin, J. Zeng, and L. Zeng, "Experimental study of real-time bus arrival time prediction with GPS data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1666, no. 1, pp. 101–109, 1999.

[3] J. Patnaik, S. Chien, and A. Bladikas, "Estimation of bus arrival times using APC data," *Journal of Public Transportation*, vol. 7, no. 1, pp. 1–20, 2004.

[4] D. H. Sun, H. Luo, L. Fu, W. N. Liu, X. Y. Liao, and M. Zhao, "Predicting bus arrival time on the basis of global positioning system data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2034, no. 2034, pp. 62–72, 2007.

[5] R. P. S. Padmanaban, K. Divakar, L. Vanajakshi, and S. C. Subramanian, "Development of a real-time bus arrival prediction system for Indian traffic conditions," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 189–200, 2010.

[6] J. J. Xue, C. Liao, K. Q. Wu, and S. P. Jia, "A study on the characteristics of arrival time of buses and the service level of platform," *Journal of Transportation Systems Engineering and Information Technology*, vol. 11, pp. 74–80, 2011.

[7] Z. He, H. T. Yu, Y. Du, and J. Wang, "SVM based multi-index evaluation for bus arrival time prediction," in *Proceedings of the 2013 International Conference on ICT Convergence (ICTC)*, pp. 86–90, Jeju, South Korea, October 2013.

[8] B. Yu, W. H. K. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1157–1170, 2011.

[9] F. S. Li, *Intelligent Bus Arrival Time Prediction Method*, Beijing Jiaotong University, Beijing, China, 2009.

[10] Z. Y. Zuo and L. Wang, "Bus arrival time forecasting and real-time information publication technology," *Journal of Transportation Systems Engineering and Information Technology*, vol. 13, pp. 64–75, 2013.

[11] A. Shalaby, A. Farhan, P. Eng, and M. A. Sc, "Bus travel time prediction model for dynamic operations control and passenger information systems," *Journal of the Transportation Research Board*, vol. 53, no. 1-2, pp. 131–147, 2003.

[12] S. I.-J. Chien and C. M. Kuchipudi, "Dynamic travel time prediction with real-time and historic data," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 608–616, 2003.

[13] L. Vanajakshi, S. C. Subramanian, and R. Sivanandan, "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses," *IET Intelligent Transport Systems*, vol. 3, no. 1, pp. 1–9, 2009.

[14] D. Park and L. R. Rilett, "Forecasting freeway link travel times with a multilayer feedforward neural network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 14, no. 5, pp. 357–367, 1999.

[15] S. I.-J. Chien, Y. Ding, and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks," *Journal of Transportation Engineering*, vol. 128, no. 5, pp. 429–438, 2002.

[16] Y. Lin, X. Yang, N. Zou, and L. Jia, "Real-time bus arrival time prediction: case study for Jinan, China," *Journal of Transportation Engineering*, vol. 139, no. 11, pp. 1133–1140, 2013.

[17] A. Bladikas, F. M. Tsai, and S. I. Chien, "Evaluation of bus travel time and schedule adherence under adverse weather," in *Proceedings of the 88th Transportation Research Board Annual Meeting*, pp. 3494–3508, Washington, DC, USA, January 2009.

[18] H. J. Ran, *The Prediction of Bus Arrival Time Using Automatic Vehicle Location Systems Data*, Texas A&M University, College Station, TX, USA, 2004.

[19] J. Y. Liu, *The Design and Implementation of the Arrival Time Prediction Function of the Real Time Bus Information Query System*, Beijing University of Posts and Telecommunications, Beijing, China, 2012.

[20] C. Y. Lv and G. B. Zhao, "An improved particle swarm optimization algorithm to solve the problem of machining center composition in group technology," *Journal of Guilin University of Technology*, vol. 33, pp. 555–559, 2013.

[21] L. Xie, *Research on Bus Arrival Time Prediction and Transfer Mechanism*, Soochow University, Suzhou, China, 2014.

[22] Z. W. Zhang, *The Design and Implementation of Bus Arrival Time Prediction Based on Machine Learning*, Beijing University of Posts and Telecommunications, Beijing, China, 2014.

[23] H. Y. Shen, Z. X. Wang, and C. Y. Gao, "Determining the number of BP neural network hidden layer units," *Journal of Tianjin University of Technology*, vol. 24, no. 5, pp. 13–15, 2008.