Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 4, April 2018, pp. 785-794
DOI: 10.23940/ijpe.18.04.p20.785794

Cooperative Differential Evolution with Dynamical Population for Short-Term Traffic Flow Prediction Problem

Danping Wang[a,b,c], Kunyuan Hu[a], Maowei He[d,*], and Hanning Chen[d]

[a]*Department of Information Service & Intelligent Control, Shenyang Institute of Automation,Chinese Academy of Sciences, Shenyang, 110016, China*
[b]*University of Chinese Academy of Sciences, Beijing, 100039, China*
[c]*Shenyang University, Shenyang, 110044, China*
[d]*School of Computer Science and Software, Tianjin Polytechnic University, Tianjin, 300387, China*

Abstract

Differential Evolution (DE) is a heuristic stochastic search algorithm based on population differences, which has advantages of simple parameters and fast convergence rate. However, it has weak robustness, especially for multimodal problems. Therefore, this paper proposes a Cooperative Differential Evolution with Dynamical population (DynCDE). In the proposed algorithm, the K-means method is employed to partition the whole population. For the high convergence rate of DE/current-to-best/ 1/bin, the neighbor-based mutation strategy is applied and the dynamic population size method based on aging mechanism and lifecycle mechanism is designed to keep the balance between exploration and exploitation. This modified DE has the potential to improve prediction accuracy of neural networks. Finally, this DynCDE-based neural network model is applied to solving the short-term traffic flow prediction problem, which offers very excellent results.

Keywords: differential evolution; K-means method; dynamic population size method; short-term traffic flow prediction

(Submitted on January 13, 2018; Revised on February 16, 2018; Accepted on March 24, 2018)

1. Introduction

Differential Evolution (DE) is an optimization algorithm based on swarm intelligence theory, which works through the cooperation and competition guidance among different individuals in the swarm. It has the advantages of less control parameters, global search and so on. However, with the continuous deepening of evolution and the decrease of group diversity, it has the potential risk of falling into the local optimum with weak robustness.

To improve the performance of DE, some scholars have proposed some methods. Paper [6] proposed a generalized mutation strategy framework, which is convenient for users to choose the appropriate type of mutation operation and also facilitates the development of new mutation operators. Paper [8] introduced trigonometric variation in DE, treating the individual as the center point of a hyper-triangle, generating the three edges of the hyper-triangle formed by three groups of weighted difference vectors, each moving in different new variant individuals to increase the probability of the algorithm to jump out of local minima. Paper [7] introduced the acceleration and migration operations in DE, in which the acceleration operation used gradient information to direct the best individual to a better region. To prevent premature convergence, the migratory operation is used to regenerate the new individual in the vicinity of the optimal individual and replace the old individual when the dispersity of the population is lower than a certain threshold. Paper [5] proposed a variety of population DE and used it to solve the multi-pole optimization problem. Paper [1] illustrated a self-adjusting weight centroid variation strategy constructed by designing an adaptive convergence factor, which can obviously improve its local enhancement ability.

* Corresponding author.
E-mail address:hemaowei@hotmail.com.

This paper presents a Cooperative Differential Evolution with Dynamical Population based on a dynamic population strategy (named DynCDE). First, the method of means of average clustering is used to realize the multi-population division by distance. To overcome the convergence of DE / current-to-best / 1 / bin algorithm being too fast, a mutation strategy based on neighbor relationship is designed. In addition, the sizes of subpopulations are dynamically controlled by means of age mechanism and life cycle strategy to achieve the balance between local search and global search.

Due to the uncertainties and highly nonlinear characteristics of short-term traffic flow prediction, it is very hard to find a mathematical model that can accurately represent the characteristics of traffic flow. Neural network technology has a very strong ability of non-linear mapping. It is relatively easy to carry out modeling and analysis of complex traffic problems. However, there are many limitations to neural networks. Therefore, this paper presents a short-term traffic prediction model based on the DynCDE neural network.

2. Cooperative Difference Evolutionary Algorithm Based on Dynamic Population Strategy

2.1 Differential evolution algorithm

Mutation: DE algorithm achieves mutation through the difference strategy, which is the main difference from the GA algorithm:

$$v_i(g+1) = x_{r1}(g) + F \cdot \left(x_{best}(g) - x_i(g)\right) + F \cdot (x_{r1}(g) - x_{r2}(g)), \quad i \neq r1 \neq r2, \tag{1}$$

where F is scaling factor, $x_i(g)$ is i-th individual of the g-th population, and x_{best} is best individual. The validity of the variants vector needs to be judged and cannot exceed the boundary. If it does not meet the boundary conditions, then it should be deleted and the initial method is used to generate a new individual.

Crossover: The two initial crossover individuals are the individuals x_{ij} in the G generation population and their variants.

$$u_{ij}(g+1) = \begin{cases} v_{ij}(g+1), & if\ rand(0,1) \le CR\ or\ j = j_{rand} \\ x_{ij}(g), & othersize \end{cases} \tag{2}$$

where CR is the crossover probability, j_{rand} is random integer, and v_{ij} is j-th dimension of i-th individual. In order to ensure that at least one gene is inherited to the next generation, the first crossover gene is randomly selected from the variant individual $v_i = (g+1)$ as an allele. The subsequent crossover process is generated by Equation 2 by comparing the random number with CR.

Selection: The greedy selection technique is often used in DE algorithms to decide which individuals should be selected:

$$x_i(g+1) = \begin{cases} u_i(g+1), & if\ f(u_i(g+1)) \le f(x_i(g)) \\ x_i(g), & othersize \end{cases} \tag{3}$$

Comparing the fitness values of the individuals vi = (g + 1) and xi(g), the individuals with good fitness are selected to the next generation.

2.2 Modified Differential Evolution Algorithm

To this end, this paper proposes a series of Cooperative Differential Evolution with Dynamical Population (DynCDE). This algorithm uses the K-means clustering method to achieve multi-population partitioning and improve the selection of differential vectors. Finally, the relationship between sub-population development and subpopulation exploration is balanced by age mechanism and life-cycle mechanism.

2.2.1 Multi-population strategy based on K-means clustering

Clustering is the process of classifying and recombining similar data according to some attributes. K-means clustering, which is a simple but classic clustering method, belongs to unsupervised learning, taking distance as a basis. The mean is the criterion for distance and K is the number of clusters.

Table1. Pseudo-code for *K*-means clustering

Algorithm 1: *K* mean clustering	
1	select *K* points as the initial clustering centers
2	**Repeat**
3	each point is assigned to the nearest clustering centers, to form *K* clusters
4	recalculate the clustering centers of each cluster
5	**Until** clusters do not change or reach the maximum number of iterations

First, *K* points are assigned as the initial clustering centers, and then the distance from other points to the center point is calculated and these points are divided to t certain ceni into ceni ones. However, the distance from other points to the cluster center is called the cluster radius. Here's how to calculate the cluster center and cluster radius.

If class *i* contains n_i individuals, these individuals are set as x_1, x_2, x_{ni}, Then, the clustering centers *cen* (*i*) is calculated as:

$$cen(i) = \frac{\sum_{i=1}^{n_i} x_i}{n_i} \tag{4}$$

The cluster radius *R* is the average Euclidean distance of members in the population to the center. The *i*-th cluster radius is calculated as:

$$R(i) = \frac{\sum_{l=1}^{n_i} \left\| x_l - cen(i) \right\|}{n_i} \tag{5}$$

The general steps [4]:

Step 1: Initialization: The predefined cluster type N objects in the matrix X are randomly selected as the initial cluster center.

Step 2: Iteration: According to the similarity criterion, data are assigned to the nearest cluster centers to form a class. Initialize membership matrix.

Step 3: Update clustering center. Then, use the average vector of each class as a new clustering center, and then redistribute the data.

Step 4: Repeat Step 2 and Step 3 until the suspension condition is met.

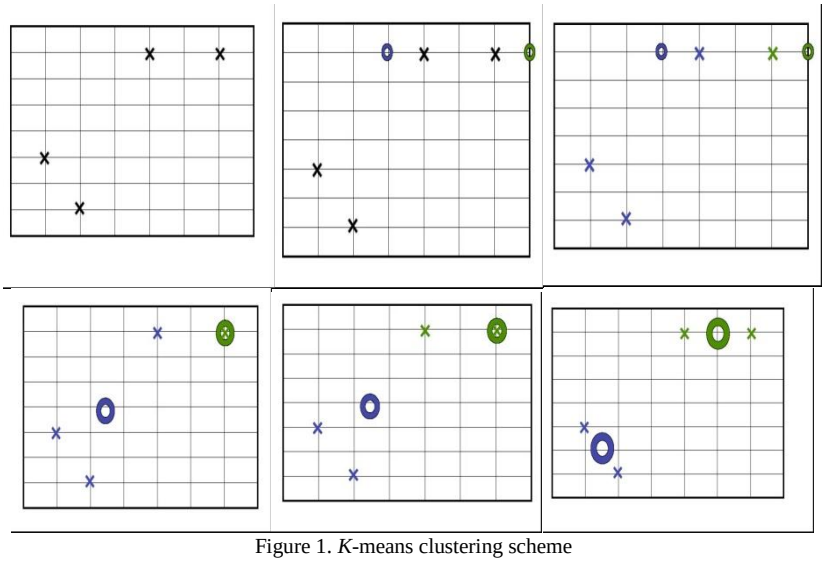


Figure 1. *K*-means clustering scheme

Give a simple example to illustrate the problem:

There is a set of data sets x_1=(2,1), x_2=(1,3), x_3=(6,7), x_4=(4,7)

(1) Select the clustering center, which can be arbitrarily selected through the histogram or be taken from the first two values.

(2) Calculate the distance between each of the two clustering centers.

2.2.2 Mutation Strategy Based on Neighbor Relationship

The DE/current-to-best/1/bin algorithm has a very high rate of convergence but low confidence. To overcome this shortcoming, this chapter will improve the first mutation strategy of the DE algorithm to maintain the high degree of convergence, enhance the diversity of algorithms, and achieve the purpose of enhancing the credibility of the algorithm at the same time.

For the first difference vector, the original strategy is to use the global optimum of individuals with the current individual. In this paper, the difference between the nearest individual in the "memory set" and the current individual is used. The "memory set" includes the best individuals found by all subpopulations. This strategy can prevent the population from going to the global optimal convergence and can encourage sub-populations to explore the local optimal region.

For the second difference vector, the best and worst individuals in the updated neighbor are chosen. It can guide the current individual to move closer to adjacent individuals.

Therefore, the first mutation strategy formula will be updated as:

$$v_{mut,G}^{j} = x_{i,G}^{j} + F_{men}^{j} \cdot (x_{men,G}^{j} - x_{i,G}^{j}) + F_{bw}^{j} \cdot (x_{n\_best,G}^{j} - x_{n\_worst,G}^{j}) \tag{6}$$

$$F_{men}^{j} = 0.3 + 0.7 \cdot rand(0,1) \cdot (1 - \frac{\left| x_{men,G}^{j} - x_{i,G}^{j} \right|}{\left| SR^{j} \right|}) \tag{7}$$

$$F_{bw}^{j} = 0.3 + 0.7 \cdot rand(0,1) \cdot (1 - \frac{\left| x_{best,G}^{j} - x_{worst,G}^{j} \right|}{\left| SR^{j} \right|}) \tag{8}$$

where, $\left| SR^{j} \right|$ represents the search scope of the j-th search dimension, $x_{n\_best,G}^{j}$ and $x_{n\_worst,G}^{j}$ are two individuals, respectively, that minimize positive and negative $\frac{1}{r_{ik}}(1 - \frac{f(X_{k,G})}{f(X_{i,G})})$ $k = 1,2,...,m$ *and* $k \neq i$.

2.2.3 Age Mechanism and Life Cycle Strategy

This paper uses the age mechanism to control the speed of individual exploration of space and prevent premature fall into the local most. At the same time, the life cycle mechanism is adopted to realize the competition among the populations and realize the dynamic change of the population size.

Table 2. Pseudo-code for Age Mechanisms

	Algorithm 2: Algorithm for Aging Mechanism:
1	**If** i-th subpopulation has found out the global optimum
2	Do none;
3	**Else if** j-th individual owns an outstanding performance in i-th subpopulation
4	$Age\_best(i, j) = Age\_best(i, j) + 1$
5	**If** $Age\_best(i, j) \geq 10$
6	re-generate the i-th subpopulation and reset $Age\_best(i,:)$ and $Age\_worst(i,:)$ as 0.
7	**End If**
8	**Else if** j-th individual has an unacceptable performance in i-th subpopulation
9	$Age\_worst(i, j) = Age\_worst(i, j) + 1$
10	**If** $Age\_worst(i, j) \geq 10$
11	re-generate the j-th individual and reset $Age\_best(i,:)$ and $Age\_worst(i,:)$ as 0.
12	**End If**
13	**End If**

A) Age mechanism [2]

Age_best and Age_worst are two matrices of $Ms * m$, where m is the number of individuals in a subpopulation and Ms is the number of subpopulations. The positions i and j in Age_best and Age_worst represent the i-th individual and the j-th subpopulation, respectively. An individual with a consistently bad performance should be regenerated, and its corresponding value in Age_worst is reset as 0. Oppositely, an individual with an outstanding performance should be regenerated, and the corresponding values in Age_best and Age_worst are both reset as 0. The maximum and minimum values of the age mechanism are set to $Age^+ = Age^- = 10$. The working methods of the specific age mechanism can be referred to Table 2.

B) Life cycle mechanism [9]

The life cycle mechanism uses individual performance evaluation strategies based on nutritional values. Each individual is divided into two classes according to the average fitness f_{mean} of the entire population. The first class is a set of good individuals whose performance is better than the average, and the other is a poor set of individuals whose performance is worse than the average. The current individual, as a good individual, calculates their nutritional value according to their location. If the nutrient value is above a certain threshold $Lc+$ (here, set to 5), the individual will be classified to generate an individual near the original, but some of the dimensions will mutate. Similarly, if the individual is in a poor concentration of individuals, the individual will be removed if their nutritional value continues to decrease, reaching a certain threshold $Lc-$ (here set to -5). The working methods of specific life cycle mechanism can be referred to Table 3.

Table 3. Pseudo-Code for Lifecycle Mechanisms

Algorithm 3: Life Cycle Mechanism

Calculate the average of all individual fitness values $f_{\text{mean;}}$
All individuals are ranked according to the fitness values F. And *f*mean is used as a boundary, the fitness value sorting F is combined into two subsets $F+$ and $F--$;
While ($i \le$ Ns)
    If $f_i > f_{\text{mean}}$
        Find the place $F_i^+$of individual in ascending order F^+
        The nutritional value $Nutri(i) = Nutri(i) + F_i^+ / \text{Num}(F^+)$
        If $Nutri(i) \ge 5$
            The individual splits
            Population increases $Ns = Ns + 1$;
            For j=1:D
                If rand(0, 1) > 0.9
                    The dimension is redistributed;
                    $Pop(i, j) = Limit^-(j) + rand(0,1) \cdot (Limit^+(j) - Limit^-(j))$
                End If
                $Nutri(i) = \text{mean}(Nutri)$
            End
        End If
    Else
        Find the place $F_i^-$of individual in ascending order F^-
        The nutritional value $Nutri(i) = Nutri(i) - F_i^- / \text{Num}(F^-)$
        If $Nutri(i) \le -5$
            The individual died;
            $Nutri(i) = []$;
            Population increases $Ns = Ns - 1$;
        End If
    End If
End

2.2.4 Overall process description

The operation mechanism of the DynCDE algorithm is as follows: Firstly, the parameters are initialized and the K-means clustering method is used to divide the population. In the cycle, the individual's position is first updated for each subpopulation using formulas (6), (7) and (8), using an age mechanism within that subpopulation to prevent individuals from falling into local optima and preventing individuals from converging. After each subpopulation has run the age mechanism, life cycle mechanism is then run throughout the population.

The life-cycle strategy is a global way of judging the subpopulation's superiority, so as to control the size of the sub-population. The excellent population will acquire more individuals, and the number of poor individuals will shrink within the population.

Table 4. Pseudo-code to improve the algorithm

Algorithm: DynCDE algorithm
Initialization parameters: population size Ns, *Age_best* and *Age_worst* matrices, number of subpopulations M_s;
Initialize population *Pop* and calculate fitness value *F*;
Using K-means clustering, the population is divided into several subpopulations ,where *K*=*Ms*;
While (whether to terminate)
    For *i* = 1:*Ms*
        For *j* = 1:Sub_Pop_i
            Update the individuals in the subpopulation according to equations (6), (7), (8)
        End
        Age mechanism. Remove individuals that converge too quickly or do not converge
    End
    Run the life cycle mechanism to determine which individuals split and which individuals died;
    Recalculate fitness values for each individual F_i
End

2.3 Algorithm performance test

2.3.1 The Benchmark Function

Four test benchmark functions are selected, as shown in Table 5. All four test functions are multimodal functions. The dimension of the test function is 30 dimensions. F1 and F2 are selected from the basic test function set, and F3 and F4 are selected from the CEC2005 function set.

Table 5. Test functions and their characteristics

Func. Number	Func.	Dimension	Minimum	Maximum
F_1	Rastrigrin	30	-5.12	5.12
F_2	Schewfel	30	-500	500
F_3	Shifted Rastrigrin	30	-5	5
F_4	Shifted Schwefel	30	-100	100

2.3.2 Parameters Setting

The typical DE algorithm and the SaDE algorithm are selected to be compared with DynCDE. The parameters of the DE algorithm include: *F* is set to 0.3, and the *Cr* is set to 0.8; the parameters of the SaDE algorithm include: *c* is set to 0.1, and *p* is set to 0.05; the DynCDE algorithm is set as: the population number *Ms* is set to 8, the Age mechanism upper and lower line *Age* is 10, and the life cycle upper and lower line *Lc* is 5.

Table 6: Test function in the 30-dimensional case of the test results of the algorithm

Benchmark Function		DE	SaDE	DynCDE
F_1	Mv	1.3563E+000	3.5339E-007	0
	Std	6.3609E-001	7.0073E-008	0
F_2	Mv	4.1378E+001	2.3035E-004	0
	Std	1.8733E+001	2.0890E-005	0
F_3	Mv	4.3504E-003	5.7743E-001	1.3563E-018
	Std	9.8941E-004	1.6404E-001	1.0308E-018
F_4	Mv	1.2942E+003	-1.3589E+001	-4.5000E+002
	Std	3.6260E+002	-3.2099E+001	0

My: average value; Std: variance

2.3.3 Analysis and discussion

Figure 2 shows the search convergence curves of the DynCED, SaDE and DE algorithms. Clearly, the convergence speed of the DynCDE algorithm outperforms that of the others. This is mainly because of the acceleration of the statement cycle. Table 6 also gives the statistical results of each algorithm for the multimodal function test. As seen from the results, the search accuracy of the DynCDE algorithm is significantly higher. This is mainly because the multi-population strategy can control the population diversity effectively. At the same time, the life-cycle mechanism limits the occurrence of significant local optimums.

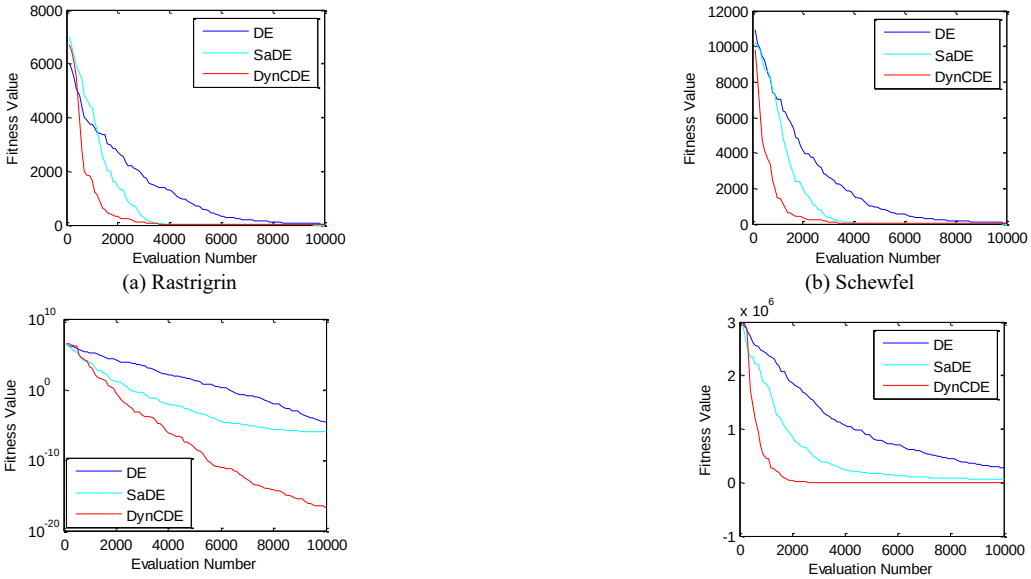


Figure 2. Fitness Curve

3. DynCDE's Application for neural network weight optimization

3.1 Neural Network

Neural network has some remarkable characteristics: it has the ability of nonlinear mapping, doesn't need a precise mathematical model, and is good at learning useful knowledge from input and output data. Also, it is easy to implement parallel computing. The neural network consists of a large number of simple computing units, which can be easily implemented by hardware and software [3].

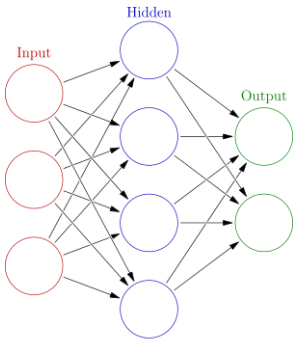


Figure 3. neural network

BP neural network includes three layers, as shown in Figure. 3. Because the neural network has the ability to complete space mapping through sample learning, it has become the main tool for modeling, simulation and prediction of nonlinear systems.

3.2 Neural network training process based on DynCDE

The essence of the neural network based on DynCDE is to map the weights and thresholds of neural networks to the individual position dimensions in DynCDE, that is, the dimension of the solution. These parameters are optimized by the continuous individual position in order to get optimal solution and network training. This network is defined as DynCDE-NN. The DynCDE-NN network training process is as follows:

According to the length of the input vector of neural network, determine the number of input nerve units *I*. The number of output layer neurons *O* is based on the output vector. Also, the number of neurons *H* in hidden layers is predefined. *MCN* is the maximum training times. The initialization networks of $w_{ij}, v_{jk}, \theta_j, \psi_k$ are the connection weights of three layers in the range of (0, 1).

The parameters needed to be optimized for the BP network are shown in Figure 3 and can be represented by a one-dimensional matrix:

$$\left| \overbrace{1 \quad 2 \quad \cdots \quad I \times H}^{w_{ij}}, \overbrace{1 \quad 2 \quad \cdots \quad H \times O}^{v_{jk}}, \overbrace{1 \quad 2 \quad \cdots \quad H}^{\theta_j}, \overbrace{1 \quad 2 \quad \cdots \quad O}^{\psi_k} \right|$$

Because of $i = 1, 2, ..., I$, $j = 1, 2, ..., H$, and $k = 1, 2, ..., O$, the size of the matrix that is combined together in these optimized parameters can be expressed as:

$$MSE = \frac{1}{P} \sum_{p=1}^{P} \sum_{k=1}^{O} (y_k - c_k)^2 \tag{9}$$

3.3 Curve fitting test

In this paper, the curve fitting of f_{s1}= sin (x) and f_{s2} = sin (x) +2*sin (2x). Figure 4 gives the fitting effect. The circle point in the graph is the training point. The transverse coordinates of a graph are $x_f = 2\pi * i / 20$, respectively, of which i = 0, 1, …, 19, 20. The point used for testing is $x_c = 2\pi * i / 100$, of which i = 0, 1, ..., 99, 100.

(a) (b)

Figure 4. curve fitting effect (a) Simulation for $f_1(x)$ = Sin(x) (b) Simulation for $f_2(x)$=sin(x) + 2 * sin(2x)

From Figure. 4, we can see that the neural network trained by the DynCDE algorithm has a distinct advantage over the BP algorithm training neural network in fitting the curve accuracy. Especially in the hanging point area, the deviation from the real value is smaller. Table 7 gives static statistical results. By comparing training errors, it is obvious that BP-NN is an order of magnitude higher than the error of DynCDE-NN.

Table 7. final training error

Problem		BP-NN	DynCDE-NN
f_{s1}	MES_{Max}	4.0954	0.7381
	MES_{Min}	1.4666	0.4694
	MES_{Ave}	2.2426	0.2527
	MES_{Std}	1.1538	0.1461
f_{s2}	MES_{Max}	5.3537	0.9244
	MES_{Min}	2.0374	0.6447
	MES_{Ave}	3.1664	0.3496
	MES_{Std}	1.5662	0.2047

4. Short-term traffic flow prediction based on DynCDE

4.1 Target Problem

Urban road traffic is a dynamic complex system. With the reduction of observation time, traffic characteristics are transformed from certainty to randomness, and the prediction difficulty of traffic flow is also improved. Generally, the forecast of traffic flow from 5min to 1h is called short-term traffic flow prediction. It has a wide application prospect in intelligent traffic control and induction, which can alleviate or solve the problem of urban traffic congestion. The target problem is non-linear, random and uncertain. Presently, urban traffic control and induction generally adopt the pre-set scheme, while a few cities adopt the adaptive control mode based on traffic flow detection. However, the application of intelligent traffic control is still scarce.

4.2 Experimental conditions

Experiments are carried out using real data from the Shenyang Traffic Flow Measurement System. Three sections of Qingnian Road from south to north (section 1), Wanliutang Road from south to north (section 2) and Xishuncheng Road

from south to section (section 3) of Shenyang, Liaoning Province, China, were sampled. The sampling interval is 5min. The data of 1930 * 3 traffic flow from 6:00 am to 10:00 pm of the morning of November 21 to 2016-11-30 are taken as the training sample set for training the prediction model based on the neural network. At the same time, the data of 193*3 traffic from 6:00 to 22:00 of 2016-12-01 are the test sample set.

4.3 The experimental results

Mean absolute percentage error is used as the standard to evaluate the prediction accuracy:

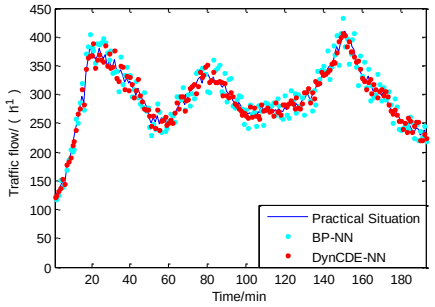
$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Y_i - Y_i^*}{Y_i}\right| \times 100\% \tag{10}$$

where Y_i is the true measurement value and Y_i* is the predicted value.

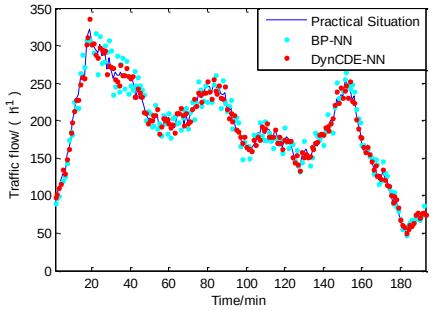
Here, we compare the prediction accuracy of the neural network model (BP-NN) based on BP algorithm and the DynCDE-NN based on the DynCDE algorithm.

Table 8. Comparison between predicted and actual measurement results

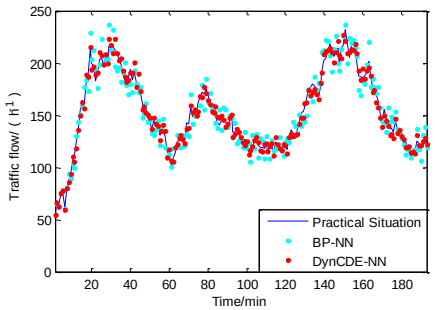
Algorithm	Question	MAPE/%
BP-NN	Section 1	4.37%
	Section 2	5.08%
	Section 3	6.73%
DynCDE-NN	Section 1	2.34%
	Section 2	3.63%
	Section 3	4.09%



(a)



(b)



(c)

Figure 5. Comparison of Prediction Value and Measured Value of Traffic Flow in Three Sections. (a) Prediction results for Section 1; (b) Prediction results for Section 2; (c) Prediction results for Section 3

Figure 5 shows the comparison between the predicted value of traffic flow and the measured value of three sections. As shown in Figure 5, the road conditions of the three sections are different. Although the traffic flow intensity of the morning peak and the evening peak are different, the time of occurrence is similar.

The light blue dots in the graph represent the predicted results obtained by BP-NN, and the red dots represent the predicted results obtained by DynCDE-NN. Intuitively, in Figure 5, the prediction results based on BP-NN are inferior to the prediction results of DynCDE-NN. Table 8 also gives the statistical results of the data. By comparing the MAPE values obtained by comparing BP-NN and DynCDE-NN, it is obvious that the neural network optimized based on DynCDE can better characterize the short-term traffic information and predict the congestion situation on the road.

5. Conclusions

Short-term traffic flow prediction is an important part of urban transportation systems. The key to traffic control and guidance is accurate prediction. In this paper, the improved differential evolution algorithm DynCDE combined neural network is applied to short-term traffic prediction in intelligent transportation. Compared with the BP neural network, the improved algorithm has higher prediction accuracy. The results show that the performance of this model based on the DynCDE-NN short-term traffic prediction model is good.

Acknowledgements

This work is supported by the National Key Research and Development Plan of China under grant No. (2016YFB1100501, 2017YFB1103603, 2017YFB1103000), National Natural Science Foundation of China under grant No. (61772365, 41772123, 61602343, 51607122, 51575158 and 51378350, 61603261), Liaoning Province Education Administration (L2015360) and Tianjin Province Science and Technology Projects under grant No. (16ZLZDZF00150, 17JCQNJC04500, 17JCYBJC15100).

References

1. N. Awad, M. Z. Ali, and R. G. Reynolds, “A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization”, *in Proceedings of 2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 103 - 108, Sendai, Japan, July 2015.
2. S. Das, A. Mandal, and R. Mukherjee, “An Adaptive Differential Evolution Algorithm for Global Optimization in Dynamic Environments”, *IEEE Transactions on Cybernetics*, vol. 44, no. 6, pp. 966-978, June 2014.
3. N. Gupta, “Artificial Neural Network”, Network and Complex Systems, Vol.3, No.1, pp. 1-5, September 2013.
4. U. Halder, S. Das, D. Maity “A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments”, *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 881 - 897, June 2013.
5. L. D. Hang, H. H. Vinh, N. T. Trung, and N. Q. Hung, “A new design approach based on differential evolution algorithm for geometric optimization of magnetorheological brakes”, *Smart Materials and Structures*, vol. 25, no. 12, pp. 341-344, November 2016.
6. S. M. Islam, S. Das, S. Ghosh, and S. Roy, “An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization”, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 42, no. 2, pp. 482-500, April 2012.
7. D. Teijeiro, X. C. Pardo, D. R. Penas, P. González, J. R. Banga, and R. Doallo, “A cloud-based enhanced differential evolution algorithm for parameter estimation problems in computational systems biology”, *Cluster Computing*, vol.20, no. 3, pp. 1937–1950, September 2017
8. C. Vaishali, T. K. Sharma, A. Abraham, and J. Rajpurohit, “Trigonometric Probability Tuning in Asynchronous Differential Evolution”, *Soft Computing: Theories and Applications* (online since November 2017) (DOI 10.1007/978-981-10-5699-4_26).
9. X. H. Yan, Y. L. Zhu, H. Zhang, H. N. Chen, and B. Niu, “An Adaptive Bacterial Foraging Optimization Algorithm with Lifecycle and Social Learning”, *Discrete Dynamics in Nature and Society*, vol. 10, no 1, pp. 1-20, October 2012.