Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7258780, 18 pages
https://doi.org/10.1155/2020/7258780

*Research Article*

# Adaptive Cultural Algorithm-Based Cuckoo Search for Time-Dependent Vehicle Routing Problem with Stochastic Customers Using Adaptive Fractional Kalman Speed Prediction

**H. Xue** [1,2,3,4]

[1]*School of Navigation, Jimei University, Xiamen, China*
[2]*National and Local Joint Engineering Research Center of Ship Aided Navigation Technology, Jimei University, Xiamen, China*
[3]*Fujian Shipping Research Institute, Jimei University, Xiamen, China*
[4]*Xiamen Southeast International Shipping Research Center, Jimei University, Xiamen, China*

Correspondence should be addressed to H. Xue; imlmd@163.com

Received 24 March 2020; Revised 27 June 2020; Accepted 30 June 2020; Published 24 July 2020

Academic Editor: A. M. Bastos Pereira



For the Time-Dependent Vehicle Routing Problem with Stochastic Customers (TDVRPSC), an adaptive Cultural Algorithm-Based Cuckoo Search (CACS) has been proposed in this paper. The convergence of the new algorithm is proved. An adaptive fractional Kalman filter (AFKF) for traffic speed prediction is proposed. An adaptive mechanism for choosing the covariance of state noise is designed. Its mathematical process is proved. Several benchmark instances with different scales are tested, and new solutions are discovered, which are better than the published solutions. The effects of the parameters on the convergence and the results are studied. According to cargo weight of customers to be delivered, the customers can be divided into large, small, and retail customers. The algorithm is tested with fixed demand probability and also different customer types with stochastic demand. The traffic speeds in different business districts in Xiamen at different times are predicted by AFKF. The results show that AFKF has smaller prediction error and better prediction accuracy than fractional Kalman filter and Kalman filter. The effect of different fractional orders on prediction error is compared. The performance of the new algorithm is compared with that of the cultural algorithm and the Cuckoo Search. The result shows that the new algorithm can efficiently and effectively solve DTVRPSC and improve the accuracy of vehicle routing planning of time-varying actual urban traffic road.

## 1. Introduction

With the rapid development of the logistics industry, Vehicle Routing Problem (VRP) gets more and more attention. The traditional VRP uses static road network model and cannot accurately estimate the travelling time based on the actual changing traffic conditions. In the actual traffic applications, there are many uncertainties, such as random customer demand, random number of customers, random traffic speed, random conditions of road traffic, and rush hours. Therefore, Time-Dependent Vehicle Routing Problem with Stochastic Customers (TDVRPSC) is a stochastic optimization problem with extensive practical significance. It is important to study new intelligent algorithms to realize logistics automation under dynamic traffic information. The results of this paper can be applied to vehicle routing, artificial intelligence, and data filtering.

Guo studied dynamic multiobjective Vehicle Routing Problem with a corresponding carbon emission model and it was set as an optimization objective [1]. After finding optimal robust virtual routes for all customers by adopting multiobjective particle swarm optimization in the first phase, static vehicle routes for static customers are formed by removing all dynamic customers from robust virtual routes in next phase. The dynamically appearing customers append to be served according to their service time and the vehicles' statues. Global vehicle routing optimization is triggered only when no suitable locations can be found for dynamic customers. A metric measuring the algorithms robustness is given. Guo proposed an adaptive immune clonal selection

cultural algorithm [2]. Dual structure of cultural algorithm was adopted, and a hybrid selection strategy integrating selection and clonal selection was put forward. The proportion of population influenced by each selection method was adaptively adjusted according to implicit knowledge extracted from the evolution process. Guo proposed a novel multipopulation cultural algorithm adopting knowledge migration derived from human cultural interaction [3]. Knowledge extracted from the evolution process contained more effective information. By migrating knowledge among subpopulations at constant intervals, the algorithm realized more effective interaction with less communication cost. Geng presented a modified centralized algorithm based on particle swarm optimization (MCPSO) [4]. Three distributed algorithms were compared against three centralized algorithms. Results showed that MCPSO could be used as a benchmark for distributed algorithms for determining the performance gap.

Random VRP was first proposed by Stewart and Golden and has gotten more and more attention [5]. Bertsimas considered a natural probabilistic variation of the classical vehicle routing problem and used stochastic demands [6]. Bianchi et al. analyzed the performance of metaheuristics on the Vehicle Routing Problem with stochastic demands [7]. Bozorgi-Amiri et al. developed a multiple objective robust stochastic programming approach for disaster relief logistics under uncertainty [8]. Sungur et al. introduced a robust optimization approach to solve the VRP with demand uncertainty [9]. Tan et al. considered VRP with stochastic demand, where actual demand was revealed only when the vehicle arrived at the customer [10]. Ibarra-Rojas et al. studied a VRP to optimize accessibility based on six indicators [11]. Wollmer et al. used a cutting plane algorithm for solving the stochastic programming and proved the algorithm to be finite [12]. Kenyon and Morton used Monte Carlo sampling based solution procedure for stochastic VRP with large sample spaces [13]. The solution space of the classical NP-hard problem is of large scale and complex. Large-scale networks may be found in many fields and a crucial step is to estimate them from the data. Sparse reciprocal graphical models have been considered for learning large-scale networks using efficient algorithms. Alpago et al. proposed an identification procedure of a sparse graphical model associated with a Gaussian stationary stochastic process [14].

At present, perfect solutions can only be reached as close as possible within a reasonable run time. Therefore, new intelligent methods are required. In 2009, Yang and Deb proposed Cuckoo Search (CS) to solve optimization problems [15]. Mareli and Twala developed three Cuckoo Search algorithms based on dynamically changing switch parameters [16]. Ishak Boushaki et al. proposed a quantum chaotic Cuckoo Search algorithm for data clustering [17]. Ayoubi et al. proposed a nonhomogeneous Cuckoo Search algorithm for allocation of passive filters [18].

To accelerate the convergence speed, this paper adopts the double mechanism in cultural algorithm (CA) [19] and proposed a Cultural Algorithm Based Cuckoo Search (CACS). CACS can overcome the defects of slow convergence rate of CS, as well as the blind guidance of CA group space, which makes the whole population algorithm not easy to converge. Its adaptive mechanism can overcome the dependence of parameters on algorithm performance. In CACS, the individuals with high fitness form the corresponding culture of the group in the Belief space by accepting the rules. The representative individuals in the population, utilizing the outstanding individuals after adjustment, will further affect the evolution of next-generation population.

Kalman filter was proposed by Kalman in 1960 [20]. In recent years, it has been widely applied to adaptive control and system recognition [21]. With the presence of uncertainties, the robust Kalman filter based on the tau-divergence which accounts for uncertainties could be used. San Gultekin et al. considered the nonlinear Kalman filtering problem using $\alpha$-divergence measures as optimization criteria [22]. In order to extend the application of Kalman filter from integer-order to fractional-order system, Solís-Pérez et al. presented a fractional-order Kalman filter according to the Riemann–Liouville definition [23]. Sun et al. used the extended Kalman filtering for fractional-order nonlinear discrete system with Lévy noises [24]. Liu et al. proposed a generalized fractional central difference Kalman filter for nonlinear discrete fractional dynamic systems [25].

In this paper, an adaptive fractional Kalman filter for traffic speed prediction is proposed to update and estimate the time-varying vehicle speed. An adaptive mechanism is adopted in computing the covariance of prediction error. Its mathematical process is proved. A solution of classic VRP cases that is better than those optimal solutions published before is found. Using AFKF and real-time urban traffic data, vehicle speed is predicted to adjust vehicle routing in time to avoid congested roads and improve the accuracy of VRP. The contributions of the paper are summarized as follows:

(1) An adaptive Cultural Algorithm Based Cuckoo Search is proposed

(2) An adaptive fractional Kalman filter for traffic speed prediction is proposed

(3) The convergence of CACS and the mathematical process of AFKF are proved

(4) New solutions of VRP are discovered, which are better than the published solutions

(5) With real-time urban traffic data, vehicle speed is dynamically predicted to adjust the solution scheme of TDVRPSC in time

## 2. Mathematical Model of TDVRPSC

*2.1. Mathematical Model.* Stochastic time-dependent road network can be represented by a directed network $G(C, E, v_{ij}(k))$. $C$ represents a set of customer nodes, and $E$ represents a set of road segments. $|V| = N.v_{ij}(k)$ represents the travel speed from customer node $i$ to $j$ during time period $k$.

The goal is to obtain the shortest expected travel time and service time. $p\,(0 < p < 1)$ denotes the customer demand

probability. The total number of vehicles is $K$. $t_{ij}$ denotes the travelling time from customer $i$ to $j$. $\overline{x}_{ijl}$ denotes whether vehicle $l$ is responsible for the route from customer $j$ to $i$. $u_{il}$ denotes whether customer $i$ is serviced by vehicle $l$. $\tau_{il}$ denotes the service time of customer $i$ by vehicle $l$. $q_i$ is the demand of the customer $i$. $W_l$ is the total load of the vehicle $l$. The model can be described as follows:

$$\min E \max_{1\le l\le K}\left(\sum_{1\le i,j\le N, j\ne i} t_{ijl}\overline{x}_{ijl} + \sum_{1\le i\le N} p\tau_{il}u_{il}\right), \tag{1}$$

$$\text{s.t.} \sum_{1\le i\le N, i\ne j}\sum_{1\le l\le K} x_{ijl} = 1, 1\le j\le N, \tag{2}$$

$$\sum_{1\le j\le N, i\ne j}\sum_{1\le l\le K} x_{ijl} = 1, \quad 1\le i\le N, \tag{3}$$

$$\sum_{1\le l\le K} u_{il} = 1, \quad 1\le i\le N, \tag{4}$$

$$u_{il} \le \sum_{1\le j\le N} \overline{x}_{ijl} = 1, \quad 1\le i\le N, 1\le l\le K, \tag{5}$$

$$\sum_{1\le i\le N} q_i u_{il} \le W, \quad 1\le l\le K_l, \tag{6}$$

$$\overline{x}_{ijl} = \begin{cases} 1, & l \text{ selects arc } (i,j), 1\le i,j\le N, 1\le l\le K, \\ 0, & \text{else}, \end{cases} \tag{7}$$

$$u_{il} = \begin{cases} 1, & i \text{ serviced by } l, 1\le i\le N, 1\le l\le K, \\ 0, & \text{else}. \end{cases} \tag{8}$$

Hypothesis is stated as follows:

(1) Each customer is serviced exactly once. The loading and unloading services of each customer are completed only once. Equations (2) and (3) ensure that each customer is served exactly once.

(2) Each customer is loaded immediately after being visited by a vehicle. The total time includes travelling time and service time with no other times wasted.

(3) The total weight of goods demanded by customers cannot exceed the vehicle load servicing them. Equation (6) ensures the total weight of goods demanded by customers cannot exceed the vehicle load servicing them.

(4) Each customer is serviced by a vehicle only if he is visited on the route of the vehicle. Equation (5) ensures a customer is serviced by a vehicle only if he is visited on the route of the vehicle.

(5) Each customer can only be serviced by exactly one vehicle. Equation (4) ensures each customer is serviced exactly one vehicle.

(6) The positions of all customers are known.

(7) The position of the depot is known. The source of vehicles is determinant.

(8) The distances from the depot to each customer are known.

(9) The customer demand probability is a stochastic real number between zero and one.

(10) The travelling time between two customers is stochastic time-dependent due to the time-varying speed and real-time traffic conditions on roads.

The existence of at least one solution of the above problem is proved in literature [12, 13]. A cutting plane algorithm for solving the above stochastic programming was proved to be finite [8]. The above stochastic objective function (1) was proved to be a convex function on the convex hull of the set of decision vectors satisfying constraints (2)–(8) [9]. Since the local minimum of convex optimization problem is the global minimum, the local minimum solution of the above problem will also be the global minimum.

*2.2. Adaptive Fractional-Order Kalman Filter for Speed Prediction.* The distance between customer $i$ and $j$ is represented by $d_{ij}$. $\Delta t$ denotes the sampling time interval. The calculation procedure for travelling time $t_{ij}$ is shown in Figure 1. $v_{ij}(k)$ is updated and estimated by Kalman filter algorithm.

The fractional-order difference is formulated as

$$\Delta^\alpha x_k = \frac{1}{h^\alpha}\sum_{j=0}^{k}(-1)^j\gamma_j x_{k-j}, \tag{9}$$

$$\gamma_j = \binom{\alpha}{j} = \begin{cases} 1, & j = 0, \\ \dfrac{\alpha(\alpha-1)\cdots(\alpha-j+1)}{j!}, & j > 0, \end{cases} \tag{10}$$

where $\alpha$ is the fractional order. $h$ is the sampling interval. $k$ is the number of samples. The fractional-order model of speed prediction is given by

$$\Delta^\alpha x_{k+1} = A_k x_k + w_k, \tag{11}$$

$$x_{k+1} = \Delta^\alpha x_{k+1} - \sum_{j=1}^{k+1}(-1)^j\gamma_j x_{k+1-j}, \tag{12}$$

$$z_k = G_k x_k + \varepsilon_k, \tag{13}$$

where $x_k$ is the state variable of the traffic speed. $z_k$ is the measurement output. $w_k$ is the system state noise. $\varepsilon_k$ is the measurement noise. $\widetilde{x}_k$ is calculated as

$$\widetilde{x}_{k+1} = E\left[x_{k+1}|z_k^*\right], \tag{14}$$

where $z_k^*$ is the measurement sequence containing measurement output $z_1, z_2, \cdots, z_k$. $\widetilde{P}_k$ denotes the covariance of prediction error at time instant $k$, which is calculated as

$$\widetilde{P}_{k+1} = E\left[\left(x_{k+1} - \widetilde{x}_{k+1}\right)\left(x_{k+1} - \widetilde{x}_{k+1}\right)^T\right]. \tag{15}$$

$Q_k$ denotes the covariance of system noise at the time instant $k$, which is calculated as

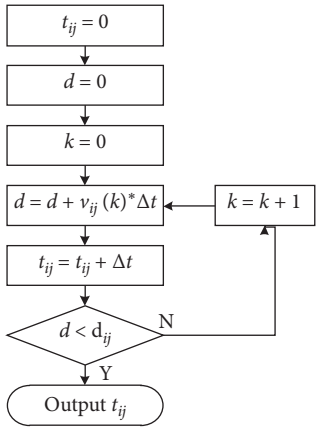


Figure 1: Calculation procedure for travelling time.

$$Q_k = E\left[w_{k-1}w_{k-1}^T\right]. \tag{16}$$

$R_k$ denotes the covariance of measurement noise at the time instant $k$, which is calculated as

$$R_k = E\left[\varepsilon_{k-1}\varepsilon_{k-1}^T\right]. \tag{17}$$

$\widehat{x}_k$ denotes the state estimation at time instant $k$, which is calculated as

$$\widehat{x}_{k+1} = E\left[x_{k+1}|z_{k+1}^*\right]. \tag{18}$$

$\widehat{P}_k$ is the covariance of estimation error at the time instant $k$, which is calculated as

$$\widehat{P}_{k+1} = E\left[\left(x_{k+1} - \widehat{x}_{k+1}\right)\left(x_{k+1} - \widehat{x}_{k+1}\right)^T\right]. \tag{19}$$

The covariance of state noise is designed as follows:

$$Q_k = \lambda E\left[w_{k-1}w_{k-1}^T\right]. \tag{20}$$

where $\lambda > 0$ is a gain coefficient.

When the covariance of state noise is small, the convergence speed is slow and the convergence accuracy is high. If the state changes too fast in a short time, the slow convergence rate will easily lead to tracking failure. When the covariance of state noise is large, the convergence speed is fast and the accuracy is low. Before reaching a stable state, choosing a larger covariance of state noise can accelerate the convergence speed. After reaching a stable state, the convergence accuracy can be improved by choosing smaller covariance of state noise.

In standard Kalman filter, $\lambda$ is a constant. However, it is difficult to determine an optimal gain in the whole process. When the covariance of state noise is small, the number of estimations will update slowly if $\lambda$ is too small. When the error changes sharply, the estimation will oscillate and even the convergence process will be unstable if $\lambda$ is too large. To accelerate the convergence process, an adaptive gain $\lambda$ is used to make the estimation process adjusted reasonably and adapt to different data characteristics. $\lambda$ increases when the covariance of state noise is small and decreases when the covariance of state noise is large. According to the noise covariance information, $\lambda$ is calculated dynamically. Therefore, an adaptive mechanism for choosing $\lambda$ is designed as follows:

$$\lambda = \frac{\lambda_{\max}}{\sqrt{1 + \widehat{P}_{k+1}^2}}. \tag{21}$$

The Kalman gain $K_k$ is calculated as

$$K_{k+1} = \widetilde{P}_{k+1}G_{k+1}^T\left(G_{k+1}\widetilde{P}_{k+1}G_{k+1}^T + R_{k+1}\right)^{-1}. \tag{22}$$

The fractional Kalman filter is formulated as

$$\widetilde{x}_{k+1} = A_k\widehat{x}_k - \sum_{j=1}^{k+1}(-1)^j\gamma_j\widehat{x}_{k+1-j}, \tag{23}$$

$$\begin{aligned} R_{k+1} = {} & \left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right)\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right)^T \\ & - G_{k+1}\widetilde{P}_{k+1}G_{k+1}^T, \end{aligned} \tag{24}$$

$$\widehat{x}_{k+1} = \widetilde{x}_{k+1} + K_{k+1}\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right), \tag{25}$$

$$\begin{aligned} \widehat{P}_{k+1} = {} & \left(1 - K_{k+1}G_{k+1}\right)\widetilde{P}_{k+1}\left(1 - K_{k+1}G_{k+1}\right)^T \\ & + K_{k+1}R_{k+1}K_{k+1}^T, \end{aligned} \tag{26}$$

$$\widetilde{P}_{k+1} = \left(A_k + \gamma_1\right)\widehat{P}_k\left(A_k + \gamma_1\right)^T + \frac{Q_k}{\lambda} + \sum_{j=2}^{k+1}\gamma_j\widehat{P}_{k+1-j}\gamma_j^T. \tag{27}$$

*Assumption 1.* $Ew_k = 0$. $w_k$ is not related to $z_k$.

*Assumption 2.* $E\varepsilon_k = 0$. $\varepsilon_k$ is not related to $x_k$ and $\widetilde{x}_k$.

**Theorem 1.** *For the fractional-order system with system state noise and measurement noise described by formulas (11)–(13), AFKF is given by formulas (23)–(27).*

*Proof.* Substituting (11) and (12) into (14), one can obtain

$$\begin{aligned} \widetilde{x}_{k+1} &= E\left[A_kx_k + w_k - \sum_{j=1}^{k+1}(-1)^j\gamma_jx_{k+1-j}|z_k^*\right] \\ &= A_k\widehat{x}_k + A_k\left(E\left[x_k|z_k^*\right] - \widehat{x}_k\right) + E\left[w_k|z_k^*\right] \\ &\quad - \sum_{j=1}^{k+1}(-1)^j\gamma_jE\left[x_{k+1-j}|z_k^*\right]. \end{aligned} \tag{28}$$

From Assumption 1, one can obtain

$$\begin{aligned} E\left[w_k|z_k^*\right] &= \sum_{i=1}^{\infty} w_k P\left(w = w_k|z = z_k^*\right) \\ &= \sum_{i=1}^{\infty} w_k \frac{P\left(w = w_k, z = z_k^*\right)}{P\left(z = z_k^*\right)} \\ &= \sum_{i=1}^{\infty} w_k \frac{P\left(w = w_k\right)P\left(z = z_k^*\right)}{P\left(z = z_k^*\right)} \\ &= \sum_{i=1}^{\infty} w_k P\left(w = w_k\right) \\ &= E\left(w_k\right) \\ &= 0. \end{aligned} \tag{29}$$

Substituting (29) into (28), one can obtain

$$\begin{aligned} \widetilde{x}_{k+1} &= A_k\widehat{x}_k + A_k\left(E\left[x_k|z_k^*\right] - \widehat{x}_k\right) \\ &\quad - \sum_{j=1}^{k+1} (-1)^j \gamma_j E\left[x_{k+1-j}|z_k^*\right]. \end{aligned} \tag{30}$$

Substituting (18) into (29), one can obtain

$$\begin{aligned} \widetilde{x}_{k+1} &= A_k\widehat{x}_k + A_k\left(\widehat{x}_k - \widehat{x}_k\right) - \sum_{j=1}^{k+1} (-1)^j \gamma_j E\left[x_{k+1-j}|z_k^*\right] \\ &= \widehat{x}_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j E\left[x_{k+1-j}|z_k^*\right] \\ &= \widehat{x}_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j E\left[x_{k+1-j}|z_{k+1-j}^*\right]. \end{aligned} \tag{31}$$

Substituting (18) into (31), one can obtain

$$\widetilde{x}_{k+1} = A_k\widehat{x}_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j \widehat{x}_{k+1-j}. \tag{32}$$

Formula (23) is obtained.
Substituting (11) and (12) into (15), one can obtain

$$\begin{aligned} \widetilde{P}_{k+1} &= E\left[\left(x_{k+1} - \widetilde{x}_{k+1}\right)\left(x_{k+1} - \widetilde{x}_{k+1}\right)^T\right] \\ &= E\left[\left(A_k x_k + w_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j x_{k+1-j} - \widetilde{x}_{k+1}\right)\right. \\ &\quad \left. \cdot \left(A_k x_k + w_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j x_{k+1-j} - \widetilde{x}_{k+1}\right)^T\right]. \end{aligned} \tag{33}$$

Substituting (32) into (33), one can obtain

$$\begin{aligned} \widetilde{P}_{k+1} &= E\left[\left(A_k x_k + w_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j x_{k+1-j} - A_k\widehat{x}_k \right.\right. \\ &\quad \left. + \sum_{j=1}^{k+1} (-1)^j \gamma_j \widehat{x}_{k+1-j}\right) \\ &\quad \cdot \left.\left(A_k x_k + w_k - \sum_{j=1}^{k+1} (-1)^j \gamma_j x_{k+1-j} - A_k\widehat{x}_k + \sum_{j=1}^{k+1} (-1)^j \gamma_j \widehat{x}_{k+1-j}\right)^T\right] \\ &= \left(A_k + \gamma_1\right)E\left[\left(x_k - \widehat{x}_k\right)\left(x_k - \widehat{x}_k\right)^T\right]\left(A_k + \gamma_1\right)^T \\ &\quad + \sum_{j=2}^{k+1} \gamma_j E\left[\left(x_{k+1-j} - \widehat{x}_{k+1-j}\right)\left(x_{k+1-j} - \widehat{x}_{k+1-j}\right)^T\right]\gamma_j^T \\ &\quad + E\left[w_{k-1}w_{k-1}^T\right]. \end{aligned} \tag{34}$$

Substituting (19) and (20) into (34), one can obtain

$$\widetilde{P}_{k+1} = \left(A_k + \gamma_1\right)\widehat{P}_k\left(A_k + \gamma_1\right)^T + \frac{Q_k}{\lambda} + \sum_{j=2}^{k+1} \gamma_j \widehat{P}_{k+1-j}\gamma_j^T. \tag{35}$$

Formula (27) is obtained.
The cost function is defined by

$$\begin{aligned} \widehat{x}_{k+1} = \arg\min_{x_{k+1}} \left\{\right. &\left(\widetilde{x}_{k+1} - x_{k+1}\right)\widetilde{P}_{k+1}^{-1}\left(\widetilde{x}_{k+1} - x_{k+1}\right)^T \\ &+ \left[z_{k+1} - G_{k+1}\widetilde{x}_{k+1} - G_{k+1}\left(x_{k+1} - \widetilde{x}_{k+1}\right)\right]R_{k+1}^{-1} \\ &\cdot \left.\left[z_{k+1} - G_{k+1}\widetilde{x}_{k+1} - G_{k+1}\left(x_{k+1} - \widetilde{x}_{k+1}\right)\right]^T\right\}. \end{aligned} \tag{36}$$

Differentiating (36) with respect to $x_{k+1}$ and making the result equal to zero, one can obtain

$$\widetilde{P}_{k+1}^{-1}\left(\widehat{x}_{k+1} - \widetilde{x}_{k+1}\right) = G_{k+1}^T R_{k+1}^{-1}\left[z_{k+1} - G_{k+1}\widetilde{x}_{k+1} - G_{k+1}\left(\widehat{x}_{k+1} - \widetilde{x}_{k+1}\right)\right]. \tag{37}$$

From (37), one can obtain

$$\begin{aligned} \widehat{x}_{k+1} &= \left(R_{k+1}G_{k+1}^{-1}\widetilde{P}_{k+1}^{-1} + G_{k+1}\right)^{-1} \\ &\quad \cdot \left(R_{k+1}G_{k+1}^{-1}\widetilde{P}_{k+1}^{-1}\widetilde{x}_{k+1} + z_{k+1} - G_{k+1}\widetilde{x}_{k+1} + G_{k+1}\widetilde{x}_{k+1}\right) \\ &= \widetilde{x}_{k+1} + \left(R_{k+1}G_{k+1}^{-1}\widetilde{P}_{k+1}^{-1} + G_{k+1}\right)^{-1}\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right) \\ &= \widetilde{x}_{k+1} + \widetilde{P}_{k+1}G_{k+1}^T\left(G_{k+1}\widetilde{P}_{k+1}G_{k+1}^T + R_{k+1}\right)^{-1} \\ &\quad \cdot \left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right). \end{aligned} \tag{38}$$

Substituting (22) into (38) yields

$$\widehat{x}_{k+1} = \widetilde{x}_{k+1} + K_{k+1}\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right). \tag{39}$$

Formula (25) is obtained.
Substituting (39) into (19) yields

$$\begin{aligned} \widehat{P}_{k+1} = E\left[\right.&\left(x_{k+1} - \widetilde{x}_{k+1} - K_{k+1}\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right)\right) \\ &\cdot \left.\left(x_{k+1} - \widetilde{x}_{k+1} - K_{k+1}\left(z_{k+1} - G_{k+1}\widetilde{x}_{k+1}\right)\right)^T\right]. \end{aligned} \tag{40}$$

Substituting (13) into (40) yields

$$\begin{aligned}\widehat{P}_{k+1} &= E\left[\left(x_{k+1}-\widetilde{x}_{k+1}-K_{k+1}\left(G_{k+1}x_{k+1}+\varepsilon_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)\right)\right.\\ &\quad\left.\cdot\left(x_{k+1}-\widetilde{x}_{k+1}-K_{k+1}\left(G_{k+1}x_{k+1}+\varepsilon_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)\right)^{T}\right]\\ &= E\left[\left(\left(I-K_{k+1}G_{k+1}\right)\left(x_{k+1}-\widetilde{x}_{k+1}\right)-K_{k+1}\varepsilon_{k+1}\right)\right.\\ &\quad\left.\cdot\left(\left(I-K_{k+1}G_{k+1}\right)\left(x_{k+1}-\widetilde{x}_{k+1}\right)-K_{k+1}\varepsilon_{k+1}\right)^{T}\right].\end{aligned} \tag{41}$$

Substituting (15) and (17) into (41) yields

$$\begin{aligned}\widehat{P}_{k+1} &= \left(1-K_{k+1}G_{k+1}\right)\widetilde{P}_{k+1}\left(1-K_{k+1}G_{k+1}\right)^{T}+K_{k+1}R_{k+1}K_{k+1}^{T}\\ &\quad-2E\left[\left(I-K_{k+1}G_{k+1}\right)\left(x_{k+1}-\widetilde{x}_{k+1}\right)\varepsilon_{k+1}^{T}K_{k+1}^{T}\right].\end{aligned} \tag{42}$$

From Assumption 2, one can obtain

$$\begin{aligned}&E\left[\left(I-K_{k+1}G_{k+1}\right)\left(x_{k+1}-\widetilde{x}_{k+1}\right)\varepsilon_{k+1}^{T}K_{k+1}^{T}\right]\\ &=\left(I-K_{k+1}G_{k+1}\right)\left[E\left(x_{k+1}\varepsilon_{k+1}^{T}\right)-E\left(\widetilde{x}_{k+1}\varepsilon_{k+1}^{T}\right)\right]K_{k+1}^{T}\\ &=\left(I-K_{k+1}G_{k+1}\right)\left[Ex_{k+1}E\varepsilon_{k+1}^{T}-E\widetilde{x}_{k+1}E\varepsilon_{k+1}^{T}\right]K_{k+1}^{T}\\ &=0.\end{aligned} \tag{43}$$

Substituting (43) into (42) yields

$$\widehat{P}_{k+1}=\left(1-K_{k+1}G_{k+1}\right)\widetilde{P}_{k+1}\left(1-K_{k+1}G_{k+1}\right)^{T}+K_{k+1}R_{k+1}K_{k+1}^{T}. \tag{44}$$

Formula (26) is obtained.

Substituting (13) into (17) yields

$$\begin{aligned}R_{k+1} &= E\left[\left(z_{k+1}-G_{k+1}x_{k+1}\right)\left(z_{k+1}-G_{k+1}x_{k+1}\right)^{T}\right]\\ &= E\left[\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}-G_{k+1}\left(x_{k+1}-\widetilde{x}_{k+1}\right)\right)\right.\\ &\quad\left.\cdot\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}-G_{k+1}\left(x_{k+1}-\widetilde{x}_{k+1}\right)\right)^{T}\right].\end{aligned} \tag{45}$$

Substituting (15) into (45) yields

$$\begin{aligned}R_{k+1} &= \left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)^{T}+G_{k+1}\widetilde{P}_{k+1}G_{k+1}^{T}\\ &\quad-2E\left[\left(G_{k+1}\left(x_{k+1}-\widetilde{x}_{k+1}\right)+\varepsilon_{k+1}\right)\left(x_{k+1}-\widetilde{x}_{k+1}\right)^{T}G_{k+1}^{T}\right]\\ &= \left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)^{T}-G_{k+1}\widetilde{P}_{k+1}G_{k+1}^{T}\\ &\quad-2E\left[\varepsilon_{k+1}\left(x_{k+1}-\widetilde{x}_{k+1}\right)^{T}G_{k+1}^{T}\right].\end{aligned} \tag{46}$$

From Assumption 2, one can obtain

$$\begin{aligned}&E\left[\varepsilon_{k+1}\left(x_{k+1}-\widetilde{x}_{k+1}\right)^{T}G_{k+1}^{T}\right]=E\left[\varepsilon_{k+1}x_{k+1}^{T}G_{k+1}^{T}\right]\\ &\quad-E\left[\varepsilon_{k+1}\widetilde{x}_{k+1}^{T}G_{k+1}^{T}\right]=0.\end{aligned} \tag{47}$$

Substituting (15) into (45) yields

$$R_{k+1}=\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)\left(z_{k+1}-G_{k+1}\widetilde{x}_{k+1}\right)^{T}-G_{k+1}\widetilde{P}_{k+1}G_{k+1}^{T}. \tag{48}$$

Formula (24) is obtained. □

Table 1: Mapping from individual to decision variable.

| $j$ | $y_j$ | $\pi_j$ |
|---|---|---|
| 1 | −0.99 | 2 |
| 2 | 1.80 | 5 |
| 3 | 3.01 | 4 |
| 4 | −0.72 | 1 |
| 5 | −1.20 | 6 |
| 6 | 2.16 | 3 |

## 3. Cultural Algorithm-Based Cuckoo Search

*3.1. Encoding Method.* Construct the mapping relationship from the individual to the solution decision variable $y$. $y$ is composed of randomly generated $(N+K-1)$-dimensional real numbers. Take Table 1 for example. Suppose $N$ is 4 and $K$ is 3. $N+K-1=6$.

$y_4$ is the largest dimension, so $\pi_4 = 1$, $y_1$ is the second largest dimension, so $\pi_1 = 2$, and so on. Finally, the whole mapped individual is $\pi = [2, 5, 4, 1, 6, 3]$.

Then, $\pi_j\,(N+1 \le \pi_j \le N+K-1)$ are regarded as the boundary lines of different vehicles. Customers within the same boundary lines are assigned to the same vehicle in turn. So $\pi_2 = 5$ and $\pi_5 = 6$ are regarded as boundary lines. So the mapped solution decision variable can be written as $\pi_i^t = [2|4, 1|3]$. This means that the second customer is assigned to the first vehicle, the fourth and the first customers are assigned to the second vehicle, and the third customer is assigned to the third vehicle.

*3.2. Cuckoo Search.* In CS, the position value of each cuckoo nest corresponds to each solution to VRP. Its goal is to search for the best cuckoo nest.

Lévy flight is a random walk strategy, whose step length can be determined by the Lévy distribution. When the new $y^{k+1}$ is discovered to be solution-suitable, a Lévy flight can be used as follows:

$$y_i^{k+1} = y_i^k + \sigma \oplus \operatorname{levy}(\beta), \tag{49}$$

where $i$ represents the $i$th cuckoo. $\sigma$ represents the step size related to the scales of the problem. $\beta$ is the index coefficient. An adaptive method can also be used:

$$\sigma = \sigma_0\left(y_j^{(t)} - y_i^{(t)}\right), \tag{50}$$

where $\sigma_0$ is a constant. The product $\oplus$ means entry-wise multiplications. The step length of a random walk by the Lévy flight is computed from a Lévy distribution:

$$\operatorname{levy}(\beta) = t^{-1-\beta},\ (0<\beta\le 2). \tag{51}$$

The steps essentially form a random walk process with a power-law step-length distribution with a heavy tail. Some new solutions should be generated by Lévy walk around the best solution obtained so far to speed up the local search. The locations of a new solution may be far enough from the current best solution; this will make sure the system will not be trapped in a local optimum. When the length of random step is determined by the maximum step length in the Lévy

distribution, the Lévy flight can effectively provide a random walk model.

To avoid parameters dependence, an adaptive mechanism is introduced. The step size of Lévy flight and discovery probability are computed as follows:

$$\sigma_i = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \frac{f_{\max} - f_i}{f_{\max} - f_{\min}}, \tag{52}$$

$$p_a = p_{a,\min} + \left(p_{a,\max} - p_{a,\min}\right)\left(1 - \frac{it}{IT}\right), \tag{53}$$

where $\sigma_{max}$ denotes the maximum step size of Lévy flight and $\sigma_{\min}$ denotes the minimum step size of Lévy flight. $f_{max}$ denotes the maximum fitness of all nests, and $f_{\min}$ denotes the minimum fitness of all nests. $p_{a,\max}$ denotes the maximum discovery probability, and $p_{a,\min}$ denotes the minimum discovery probability. $it$ denotes the current iteration, and $IT$ denotes the maximum iterations number. As the fitness of a nest decreases, its step size of Lévy flight will increase. As the time of iteration increases, the discovery probability of nests will decrease.

*3.3. Cultural Algorithm.* CA is a dual evolution mechanism proposed by Reynolds in 1994. For VRP, situation knowledge represents the optimization goal, and normative knowledge represents constraint conditions. The individuals gained in the evolution process are passed to Belief space by Accept function. Then the Belief space is updated through Update function. The Influence function uses the useful information of Belief space to guide the evolution process of population space. The main structure of the cultural algorithm is shown in Figure 2.

In the Belief space, $u$ denotes the upper bound of the individual, and $l$ denotes the lower bound of the individual. $L_j$ is lower limit of the fitness value of $l_j$, and $U_j$ is upper limit of the fitness value of $u_j$. The updating rules of situation knowledge are as follows:

$$S^{k+1} = \begin{cases} y^k_{\text{best}}, & f\left(y^k_{\text{best}}\right) < f\left(S^k\right), \\ S^t, & \text{else}, \end{cases} \tag{54}$$

where $S^k$ denotes the optimal individual in $k$ generation and $f$ denotes the fitness function. The updating rules of standard knowledge are

$$l_j^{t+1} = \begin{cases} y^t_{i,j}, & y^t_{i,j} \le l^t_j \text{or} f\left(y^t_i\right) < L^t_j, \\ l^t_j, & \text{else}, \end{cases} \tag{55}$$

$$L_j^{t+1} = \begin{cases} f\left(y^t_i\right), & y^t_{i,j} \le l^t_j \text{or} f\left(y^t_i\right) < L^t_j, \\ L^t_j, & \text{else}, \end{cases} \tag{56}$$

$$u_j^{t+1} = \begin{cases} y^t_{k,j}, & y^t_{k,j} \ge u^t_j \text{or} f\left(y^t_k\right) < U^t_j, \\ u^t_j, & \text{else}, \end{cases} \tag{57}$$

$$U_j^{t+1} = \begin{cases} f\left(y^t_k\right), & y^t_{k,j} \ge u^t_j \text{or} f\left(y^t_k\right) < U^t_j, \\ U^t_j, & \text{else}. \end{cases} \tag{58}$$

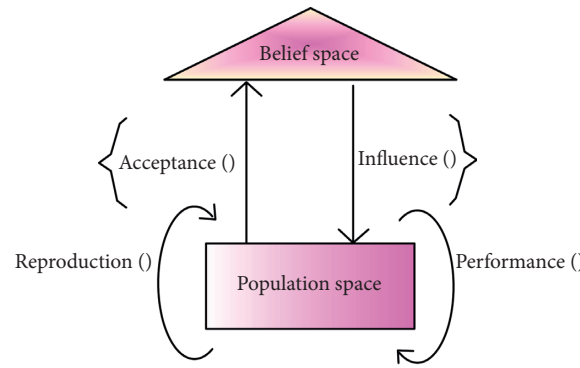


Figure 2: Structure of the cultural algorithm.

*3.4. CACS Process.* To accelerate the convergence speed of CS, this paper adopts the double mechanism of the CA. The individuals with high fitness form the culture in the Belief space through acceptance rules. The representative individuals further influence the evolution of next population through the outstanding individuals adjusted. The process of CACS with vehicle speed estimation is shown in Figure 3.

In this paper, the adaptive factor strategy is introduced. Through the dynamic adjustment of the factor to the optimization step size, the algorithm can search with adaptive step size in the early stage of evolution and improve the convergence speed of the algorithm. In the late stage of evolution, it can quickly tend to the optimal solution and further improve the convergence speed of the algorithm. This reduces the computational cost of the algorithm.

*3.5. Convergence of CACS.* The solution generated in the $k^{\text{th}}$ iteration is denoted as $y_k$. The optimal group state set is denoted as $H$. The optimal state set is denoted as $R$. All the states of all the cuckoos form a state space for the group denoted as $Q$.

**Lemma 1** (see [26]). *Assume that the objective function $f$ is a measurable function. The feasible solution space $A$ is a measurable subset on $\mathfrak{R}^n$. $Y_\varepsilon$ is a set of global optimal solutions. $P(y^k \in Y_\varepsilon)$ denotes the probability that the solution $y^k$ belongs to the optimal solutions set $Y_\varepsilon$. For any Borei subset $B$, its Lebesgue measure $\nu(B) > 0$. $\mu_k(B)$ is the probability of obtaining $B$ by measuring $\mu_k$. $\prod_{k=0}^{\infty}(1 - \mu_k(B)) = 0$. Thus,*

$$\lim_{k \longrightarrow +\infty} P\left(y^k \in R_\varepsilon\right) = 1. \tag{59}$$

**Lemma 2.** *The finite homogeneous Markov chain starting from any nonrecurrent states will reach its recurrent state with probability of 1.*

**Lemma 3.** *Suppose Markov chain has a nonempty set $C$ and there is not any nonempty closed set $D$ satisfying $C \cap D = \phi$. Then when $j \notin C$, $\lim_{n \longrightarrow +\infty} P(x_n = j) = 0$.*

**Theorem 2.** *The Markov chain represented by CS is finite and homogeneous.*

Algorithm 1 CACS with Kalman estimation of vehicle speed

Input: VRP description
Output: optimal solution
1: function CACS ()
2: Initialize population space;
3:Randomly generate an initial population of nest positions;
4: Initialize the Belief Space;
5: it = 0;
6: while it < IT
7: Get a cuckoo randomly by Lévy flights using (49)–(53);
8: Estimated vehicle speed $v_{ij}$ based on AFKF using (23)–(27);
9: Caculate $t_{ij}$ according to Figure 1
10: Calculate the goal function with constraint conditions using (1)–(8);
11: Evaluate the fitness *Fi*;
12: Choose a nest *j* randomly;
13: if *Fi* > *Fj* then
14: replace cuckoo with the new solution;
15: end if
16: Abandon part of worse nests and build new ones;
17: Evaluate the population and rank the solutions;
18: Select the best feasible solution as the next generation;
19: if receiving condition is satisfied
20: Deliver best individuals from the main population space to the belief space through the Accept function;
21: end if
22: Update the situation knowledge in the belief space through the Update function using (54);
23: Update the standard knowledge in the belief space through the Update function using (55)–(58);
24: Compute the fitness in the belief space using (1)–(8);
25: Update optimal individuals;
26: if the influence condition is satisfied
27: Replace the worst individual in the population space with the optimal individual in belief space through the Influence function;
28: end if
29: it = it + 1;
30: end while
31: end function

Figure 3: Pseudocode of CACS.

*Proof.* The search space of the optimization algorithm is limited, and the cuckoo nest state is limited. So the state space of nest position is limited. Since the population state of nest positions is composed of $m$ nest positions and $m$ is a finite positive integer, the population state of nest position is also limited.

From the process of CS, the transfer probability $P(T(y^{k-1}) = y^k)$ of any nest position is only related to the state of the previous state $y^{k-1}$. $T(\cdot)$ denotes the transfer operation. Therefore, the population states of nest position have Markov property.

The transfer probability $P(T(y^{k-1}) = y^k)$ of any nest position is not related to the time instant $k$. That is to say, the population states sequence of nest location is homogeneous. Therefore, the population states sequence of cuckoo nest location is a finite homogeneous Markov chain. □

**Theorem 3.** *When the number of iterations approaches becoming sufficiently large, the group state sequence will converge to the optimal state solution set H.*

*Proof.* $\forall y_i \in R, \forall y_j \notin R$, $f(y_i) < f(y_j)$. From the process of CS, it is obtained that $P(T(y_j) = y_i) = 0$. For the state sequence $\{y(k); k > 0\}$, the state set $R$ of optimal nest position is a closed set on the state space $Y$.

Since $R$ is a closed set on $Y$, according to the definition of closed set, the state set $H$ of the optimal nest position group is a closed set on the group state space $Q$.

Assume that there exists a closed set $B$ in the group space $Q$, such that $B \cap H = \phi$. $\forall y_i \in R, \forall y_j \notin R$, $f(y_i) < f(y_j)$. Then, $\forall y_i \in B \cap H \subseteq H$, $f(y_j) \leq f(y_i)$. Thus, it is obtained that $P(T(y_j) = y_i) \neq 0$. So $B$ is not a closed set, which contradicts the hypothesis. Therefore, there is no nonempty closed set outside $H$ in the nest position state space $Q$ such that $B \cap H = \phi$.

From Lemma 3, when the number of iterations approaches becoming sufficiently large, the group state sequence will converge to the optimal state solution set $H$. □

Each iteration of CS algorithm preserves the optimal position of the population and guarantees the nonincremental fitness. According to Theorem 3, the population sequence of cuckoo nest position will enter the optimal state after continuous and infinite iterations. Therefore, the possibility that the global optimal solution cannot be searched continuously and infinitely is 0. So the CS algorithm converges to the global optimum.

**Theorem 4.** *The random variation of the state in Belief space of CACS belongs to finite homogeneous Markov chain.*

*Proof.* In CACS, the Belief space is used to record the evolutionary knowledge in the search process, and CACS searches for the optimal value in discrete and limited space. So the Belief space used for recording the evolutionary knowledge is also limited. Meanwhile, the state transition probability in the Belief space is only related to its current state and has nothing to do with time instant $k$. Therefore, the random variation of the state in the Belief space of CACS belongs to the finite homogeneous Markov chain. □

**Theorem 5.** *CACS has global convergence with probability of 1.*

*Proof.* Denote the global knowledge state space in the Belief space as $\Omega_g$. Denote the local knowledge state space as $\Omega_l$. The whole state space of Belief space is $\Pi = \{(I_g, I_s)\}$, in which $I_g$ denotes the global knowledge set and $I_s$ denotes the local set of population space.

$\Pi$ is divided into two subspaces, namely, recurrent state space $\Pi_1$ and nonrecurrent state space $\Pi_2$. The recurrent state space can be expressed as follows:

$$\Pi_1 = \{(I_g, I_s) | I_g \in \Omega_g, I_s \in \Omega_l\}, \tag{60}$$

and thus $\Pi_2 \cap \Pi_2 = \phi$ and $\Pi_2 \cup \Pi_2 \subseteq \Pi$.

If there is a knowledge state $y \in \Pi_1$, according to the definition of global knowledge, it will not transfer to $\Pi_2$. So it is a closed set. The state in $\Pi_1$ is interlinked. So the state in $\Pi_1$ is a recurrent state. If $y \in \Pi_2$, each individual will move throughout the whole solution space due to evolution. So the convergence probability of the evolutionary population to

the global optimal solution is greater than 0. That means the transition probability of $y$ from $\Pi_2$ to $\Pi_1$ is greater than 0. Therefore, the state in $\Pi_2$ is a nonrecurrent state. According to Lemma 2, the evolution of knowledge in Belief space from any nonrecurrent state will always transfer to the recurrent state with probability of 1, that is, converge to the global optimal solution according to probability of 1. □

## 4. Example and Analysis

*4.1. Experimental Results with Deterministic Demand Probability.* The CACS algorithm is tested on Intel® Core™ i3-4150T CPU @3.00 GHz, 64-bit operating system with memory of 4.00 GB, and x64-based processor. The VRP library used is published on the Networking and Emerging Optimization homepage [27]. The capacitated VRP instance library was selected. The first layer index is the first letter of word in the instance library. The second layer index is the number of customers, and the third layer index is the number of vehicles. Take A, B, P, and E sets as examples. Set A has 27 VRP instances from A-n32-k5 to A-n80-k10 proposed by Augerat et al. Set B has 23 VRP instances from B-n31-k5 to B-n78-k10 proposed by Augerat et al. Set P has 24 VRP instances from P-n16-k8 to P-n101-k4 proposed by Augerat et al. Set E has 15 VRP instances from E-n13-k4 to E-n101-k14 proposed by Augerat et al.

Consider the case when the arrival probability is 1. Table 2 gives the shortest path length, average iteration times, and average CPU computing time of each instance.

For instance, B-n31-k5 algorithm has found a better solution than the published optimal solution, whose scheme is shown in Table 3.

Figure 4 shows the shortest path of 70 nodes when demand probability is 1. The horizontal axis represents the horizontal axis of a city map with the unit of meter. The vertical axis represents the longitudinal axis of a city map with the unit of meter.

Figure 5 shows the objective function convergence curves of 70 nodes when demand probability is 1. The horizontal axis represents iteration times. The vertical axis represents the best individual in each iteration.

*4.2. Parameter Influence Analysis.* Figure 6 shows the average results of E-n22-k4, B-n31-k5, and B-n64-k9 after 20 times calculation under different demand probability $p$ with IT 100. The updating rate of situational knowledge $\alpha$ is 0.3. The updating rate of standard knowledge $\beta$ is 0.5. The population size $m$ is 50. $\tau_{il}$ is 1.

Figure 7 shows average results of 20 times calculation under different $\alpha$ with IT 100. $p$ is 0.45. $\beta$ is 0.5 and $m$ is 50.

Figure 8 shows average results of 20 times calculation under different $\beta$ with IT 100. $p$ is 0.45. $\alpha$ is 0.45 and $m$ is 50.

Figure 9 shows average results of 20 times calculation under different $m$ with IT 100. $p$ is 0.45. $\alpha$ is 0.3 and $\beta$ is 0.5.

From the above figures, when $m$ is larger, the optimal solution is better. Increasing $m$ is advantageous to searching for better solutions. However, when $m$ reaches a certain value, the optimal solution is almost unchanged as $m$ increases. In this instances, the optimal parameters are $p = 0.45$, $\alpha = 0.3$, $\beta = 0.5$, and $m = 50$.

The sensitivity analysis of iterations times is shown in Table 4. In the case of E-n22-k4, when IT is too small, the calculation results still need to be improved and $\Delta f/\Delta N_{\max}$is large. When IT reaches 130, $\Delta f/\Delta N_{\max}$ remains close to zero as IT increases. The results are almost stable.

We have also tried to vary the number of host nests and the discovery probability $p_a$ in Cuckoo Search algorithm. We have used $n = 5$, 10, 15, 20, 50, 100, 150, 250, and 500 and $p_a = 0.01$, 0.05, 0.1, 0.15, 0.2, 0.25, 0.4, and 0.5. We found that $n = 15$ and $p_a = 0.25$ are sufficient for most optimization problems. Results and analysis also imply that the convergence rate of Cuckoo Search algorithm is not sensitive to the parameters used.

Apart from the number of host nests, there is one parameter $p_a$. This also means that we do not have to fine-tune these parameters for a specific problem. Subsequently, Cuckoo Search algorithm is more generic and robust for many optimization problems. There are fewer parameters to be fine-tuned in Cuckoo Search than in other metaheuristic algorithms.

*4.3. VRPSC Experiments with Fixed Demand Probability.* $p$ is set as different real numbers from 0.1 to 0.9. The parameters are set as follows: $\alpha = 0.3$, $\beta = 0.5$, $m = 50$, and $\tau_{il} = 1$. The initial vehicle speeds $v$ are set as stochastic real numbers within the range of [1, 6]. Table 5 gives the average results of shortest expected total travelling time and servicing time for P-n16-k8 case with different IT and different $p$ under 20 times calculation.

Table 5 shows that, with the increasing of $p$, the objective function also increases. With the increasing of IT, the objective function reduces. However, when IT reaches 300, even if IT continues to increase, the shortest expected total travelling time and servicing time will not be reduced and will remain almost the same.

*4.4. VRPSC Experiments with Distinguishing Customer Characteristics.* Customers can be divided into large customers, small and medium customers, and retail customers according to the weight of the goods they order. Consider the influence of customer property in these three cases. In the case of B-n31-k5, the customers who order goods weighing in the range of [19, 25] belong to large customers. The customers who order goods weighing in the range of [9, 18] belong to small and medium customers. The customers who order goods weighing in the range of [2, 8] belong to small and retail customers. The parameters are set as follows: $p = 0.45$, $\alpha = 0.3$, $\beta = 0.5$, $m = 50$, and $\tau_{il} = 1$. The initial vehicle speeds $v$ are set as stochastic real numbers within the range of [1, 6].

Table 6 shows the optimal solution, average solution, and standard deviation in three cases of different customer types after 20 times calculation, when the demand probability is 0.8 or 0.9. In Table 6, 1st represents the result for the first calculation, 2nd denotes the results for the second calculation, and so on.

Table 2: Calculation result of different instances when the demand probability is 1.

| Instance | A-n32-k5 | A-n39-k6 | B-n31-k5 | P-n16-k8 | P-n19-k2 |
|---|---|---|---|---|---|
| $N$ | 32 | 39 | 31 | 16 | 19 |
| $K$ | 5 | 6 | 5 | 8 | 2 |
| Optimal solution (m) | 784.612 | 831.2696 | 662.3275 | 451.9471 | 212.6569 |
| Average iteration number | 427 | 648 | 329 | 193 | 308 |
| Average CPU computing time (s) | 18.9688 | 27.1563 | 6.8125 | 3.2969 | 6.7188 |
| Published optimal solution (m) | 784 | 831 | 672 | 450 | 212 |
| Improvement ratio (%) | 0 | 0 | 1.43936 | 0 | 0 |

Table 6 shows when the large customers have least objective function. The small and medium customers have second shortest objective function. The retail customers have the longest objective function. This is because when the vehicle does not need to serve large customers, it can freely and flexibly arrange its routes. When vehicles need to serve large customers, with the limit of maximum carrying capacity, the choice of driving routes has more constraints. With the increase of $p$, the objective function also increases. The objective function with $p$ of 0.9 is greater than that with $p$ of 0.8.

*4.5. Comparative Analysis.* Table 6 gives performance of CACS, CA, and CS with different $p$. Since CACS uses CA and CS, it is more complex than these two algorithms, while its time cost and storage cost are slightly higher compared to the latter two. However, the rapid development of computer hardware technology makes this difference for solving problems of general scale not apparent. From the experimental results, CACS performs better than CA, CS, and ACO.

To avoid sensitivity to parameters selection, which is one of the main limitations in swarm intelligence algorithms, adaptive mechanism is adopted into CACS. ACO [28] used for comparison also proposed machine learning strategies to control the parameter adaptation. The objective function by different $p$ and different algorithms is listed in Table 7.

*4.6. TDVRPSC Using AFKF Speed Prediction.* Based on the technology of AFKF, the highway data analysis platform is established. After extracting the information and constructing appropriate traffic prediction model, the future vehicle speed in different business districts can be effectively estimated. When the speed of a road is detected to be below the threshold, it can be inferred that this road is congested due to accidents or other reasons. Rapid identification of congested roads is completed. This is also useful for emergencies. When traffic congestion is detected in a certain road, decision support and vehicle dispatching optimization can adjust VRP scheme in time and avoid congested roads, so as to reduce transportation time and cost. In the field of real-time traffic forecasting, the fast information processing and mining ability of large data have very high reliability for real-time VRP scheduling. The accuracy of VRP scheduling is improved for the time-varying actual urban traffic road.

Table 8 lists the traffic speed and congestion delay index in different business districts of Xiamen at 16:25 on February 17, 2019 [29]. Congestion delay index is used to evaluate the degree of urban congestion. It is the ratio of the actual travel time to the free travel time of a trip.

Table 9 lists the traffic speed in different business districts of Xiamen from 16:30 to 20:00 on February 17, 2019. The unit of speed is km/h.

Figure 10 shows Xiamen city map at 16:25 on February 17, 2019. Different business districts are marked with their congestion delay indexes.

Several branches of Yuantong Express in Xiamen city distributed in different business districts are used. Table 9 lists their business district, latitude, longitude, and the Mercator coordination converted. The No. 0 branch is the head office. VRP task is to distribute goods to all other branches.

*4.7. Accuracy of Prediction Model.* Figure 11 shows the traffic speed prediction in Huli district using AFKF prediction from 16:30 to 20:00 on February 17, 2019. The horizontal axis represents time. The vertical axis represents the traffic speed in km/h. The star marking connected with solid wire represents the real speed. The joint spider connected with dotted lines represents the predicted speed.

Table 10 shows the prediction mean square error (MSE) and mean absolute percentage error (MAPE) by AFKF, fractional Kalman filter (FKF), and Kalman filter (KF). Table 11 shows AFKF has smaller prediction error and better prediction accuracy than FKF and KF.

Figure 12 shows that the prediction errors vary with different fractional orders of Kalman filter. The horizontal axis represents fractional order. The vertical axis represents the prediction error of traffic speed in km/h. The five-pointed star represents the mean relative error. The asterisk represents the mean square error. For certain business districts, there exists a most suitable fractional order for reducing its prediction error. Compared with integer-order filters, fractional-order Kalman filters have wider parameter selection range and higher prediction accuracy.

If the prediction model is not accurate, wrong route selection may be made. For example, the traffic flow is busy in Siming district and its vehicle speed is relatively slow. If the vehicle selects Siming district with no accurate speed prediction, it will take more travelling time, even if the distance is short.

*4.8. Vehicle Routing Solution.* With four vehicles, TDVRPSC schemes including the route number, customer visiting sequence, total route length, travelling starting time,

Table 3: Comparison of new solution and published optimal solution for B-n31-k5.

| Route | Published optimal solution | | | New solution | | |
|---|---|---|---|---|---|---|
| | Travelling sequence | Distance | Capacity | Travelling sequence | Distance | Capacity |
| 1 | 30 ⟶ 23 ⟶ 8 ⟶ 12 ⟶ 28 ⟶ 26 | 109.6811 | 97 | 7 ⟶ 30 ⟶ 5 ⟶ 25 ⟶ 18 ⟶ 16 ⟶ 21 | 119.2372893 | 97 |
| 2 | 21 ⟶ 16 ⟶ 18 ⟶ 25 ⟶ 5 ⟶ 4 ⟶ 29 | 112.9915 | 86 | 12 ⟶ 24 ⟶ 19 ⟶ 1 ⟶ 3 ⟶ 6 ⟶ 9 | 102.3659802 | 99 |
| 3 | 7 ⟶ 17 ⟶ 13 ⟶ 6 ⟶ 9 ⟶ 22 | 110.8287 | 96 | 20 ⟶ 27 ⟶ 10 ⟶ 2 ⟶ 14 ⟶ 15 ⟶ 11 ⟶ 28 ⟶ 8 | 240.5088518 | 95 |
| 4 | 20 ⟶ 27 ⟶ 10 ⟶ 2 | 183.5752 | 38 | 4 ⟶ 29 ⟶ 22 ⟶ 13 ⟶ 17 ⟶ 23 | 108.7156422 | 99 |
| 5 | 14 ⟶ 15 ⟶ 11 ⟶ 24 ⟶ 19 ⟶ 1 ⟶ 3 | 155.4298 | 95 | 26 | 98.06120538 | 22 |
| Total | | 672.506 | 412 | | 668.889 | 412 |

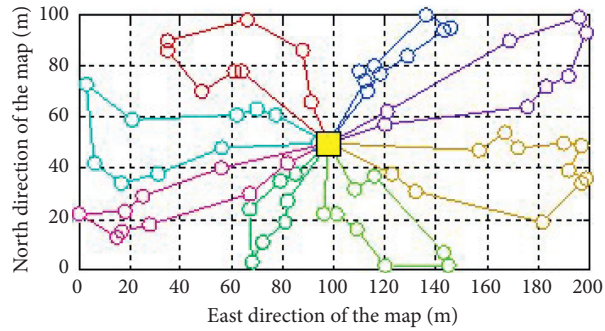


Figure 4: Optimal path of 70 nodes when demand probability is 1.

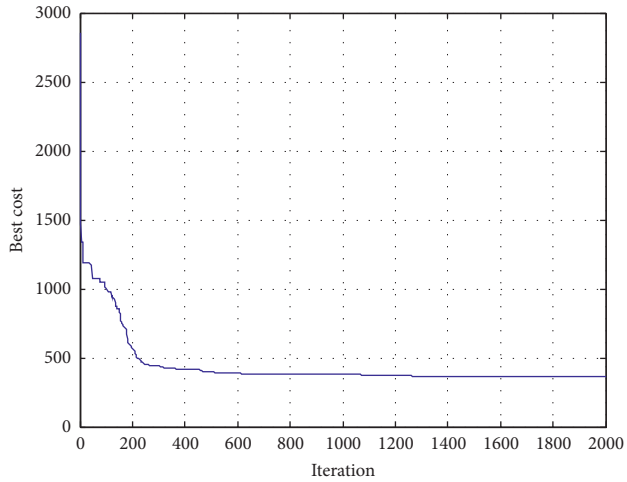


Figure 5: Objective function convergence curves of 70 nodes when demand probability is 1.

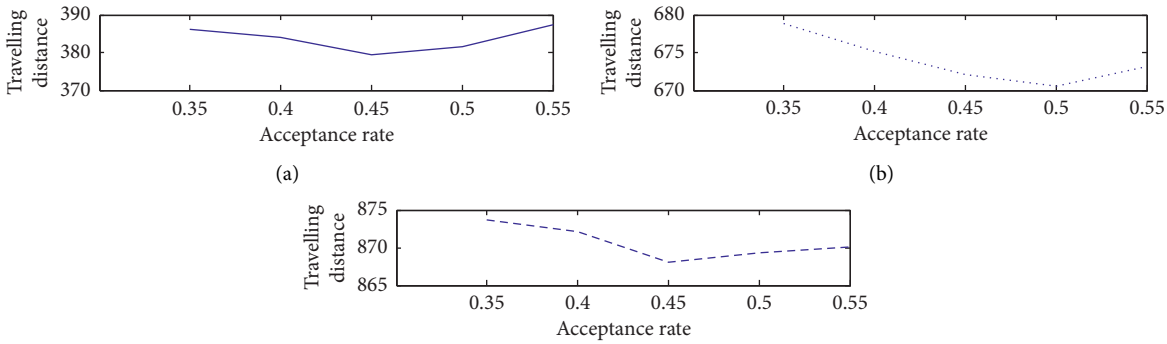


Figure 6: Calculation results of different $p$ with IT 100. (a) E-n22-k4, (b) B-n31-k5, and (c) B-n64-k9.

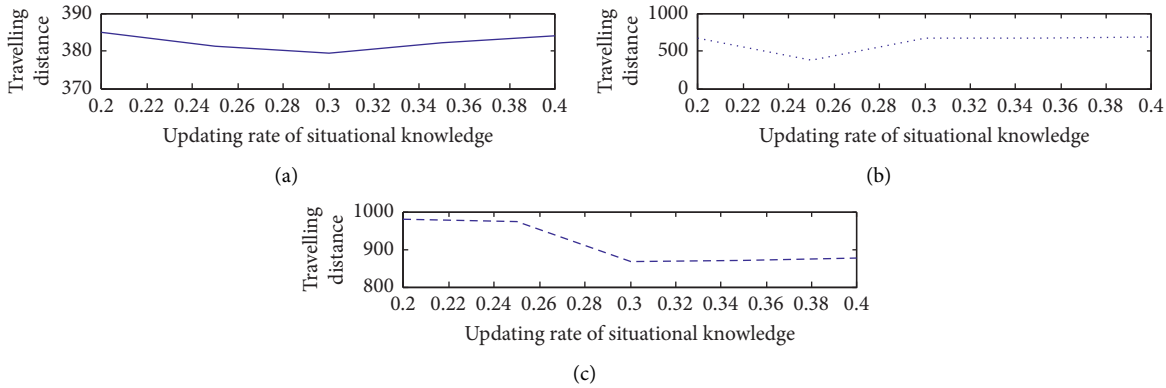


Figure 7: Calculation results of different $\alpha$ with IT 100. (a) E-n22-k4, (b) B-n31-k5, and (c) B-n64-k9.

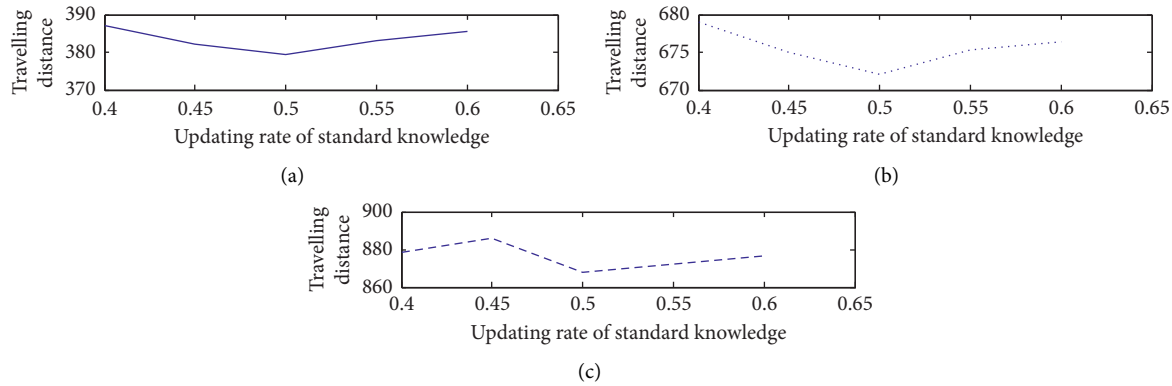


Figure 8: Calculation results of different $\beta$ with IT 100. (a) E-n22-k4, (b) B-n31-k5, and (c) B-n64-k9.

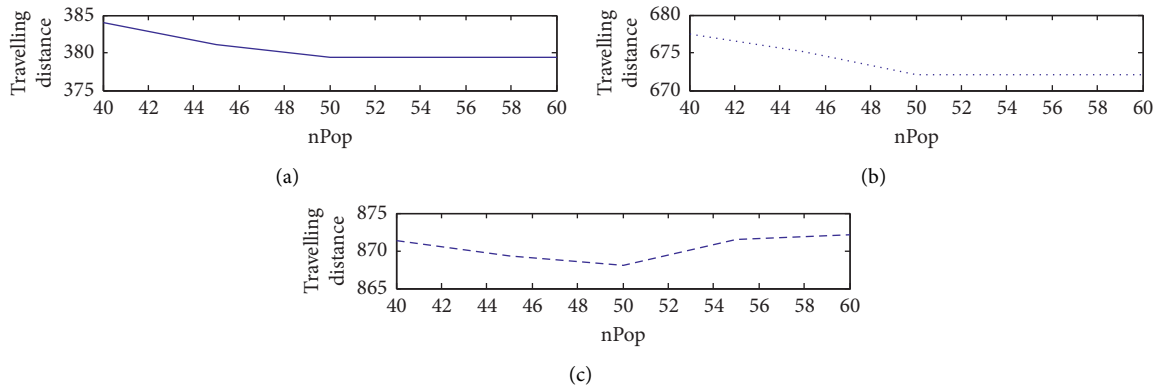


Figure 9: Calculation results of different m with IT 100. (a) E-n22-k4, (b) B-n31-k5, and (c) B-n64-k9.

Table 4: Calculation results of E-n22-k4 with different iteration times.

| $N_{\max}$ | 100 | 110 | 120 | 130 | 140 |
|---|---|---|---|---|---|
| Mean | 379.3766 | 377.4962 | 375.8675 | 375.1023 | 375.1023 |
| Std. | 0.7625 | 0.2166 | 0.1026 | 0.0125 | 0.0063 |
| $\Delta N_{\max}$ | 10 | 10 | 10 | 10 | 10 |
| $\Delta$Mean | 1.8804 | 1.6287 | 0.7652 | 0 | 0 |
| $\Delta$Mean/$\Delta N_{\max}$ | 0.18804 | 0.11287 | 0.07652 | 0 | 0 |

Table 5: Results with different IT under different $p$.

| $p$ | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| 0.1 | 47.016647 | 47.016647 | 47.016647 | 47.016647 | 47.016647 |
| 0.2 | 65.356696 | 65.356680 | 65.356680 | 65.356680 | 65.356681 |
| 0.3 | 74.817290 | 74.816978 | 74.816975 | 74.816977 | 74.816984 |
| 0.4 | 81.356627 | 81.354022 | 81.354010 | 81.354017 | 81.354064 |
| 0.5 | 86.766766 | 86.753328 | 86.753220 | 86.753164 | 86.753328 |
| 0.6 | 91.635135 | 91.584978 | 91.583532 | 91.583322 | 91.583318 |
| 0.7 | 96.183111 | 96.040084 | 96.028954 | 96.026950 | 96.026981 |
| 0.8 | 100.469282 | 100.163211 | 100.110735 | 100.106147 | 100.104906 |
| 0.9 | 104.306277 | 103.887785 | 103.744337 | 103.739400 | 103.737179 |

travelling ending time, and total travelling time are shown in Table 12. The parameter settings are the same as those in Section 4.5.

Table 12 gives the minimal expected total travelling time with different $p$, different departure time, and different algorithms. Its performance is compared with CA, CS, and

Table 6: Objective function with different customer types and different demand probability.

| Probability | 0.9 | | | 0.8 | | |
|---|---|---|---|---|---|---|
| Customer | Large | Small | Retail | Large | Small | Retail |
| 1st | 596.155 | 638.2462 | 656.2919 | 585.8801 | 590.4435 | 638.0412 |
| 2nd | 594.4813 | 637.9749 | 651.5232 | 585.6425 | 594.5347 | 636.5034 |
| 3rd | 590.7583 | 635.9657 | 653.6638 | 585.8801 | 594.5347 | 637.8991 |
| 4th | 596.155 | 638.7321 | 659.7856 | 587.4265 | 594.5347 | 636.5034 |
| 5th | 590.7583 | 639.7165 | 653.4783 | 582.6757 | 590.4435 | 634.8529 |
| 6th | 590.7583 | 637.8215 | 655.5595 | 588.2996 | 590.4435 | 636.5034 |
| 7th | 590.7583 | 637.9749 | 657.6003 | 582.6757 | 590.4435 | 634.8529 |
| 8th | 596.155 | 637.8215 | 656.7167 | 588.2996 | 594.5347 | 636.8692 |
| 9th | 590.7583 | 637.9749 | 651.375 | 582.6757 | 590.4435 | 636.5034 |
| 10th | 596.155 | 638.6415 | 654.8217 | 588.2996 | 594.5347 | 636.5034 |
| Ave. | 593.2893 | 638.087 | 655.0816 | 585.7755 | 592.4891 | 636.5032 |
| Std. | 2.714153 | 0.947677 | 2.664324 | 2.373579 | 2.156252 | 1.04854 |

Table 7: Objective function with different $p$ and different algorithms.

| $p$ | CACS | CA | | CS | | ACO | |
|---|---|---|---|---|---|---|---|
| | | Results | Rate (%) | Results | Rate (%) | Results | Rate (%) |
| 0.1 | 47.016647 | 47.105654 | 0.18895 | 47.134136 | 0.24927 | 47.091061 | 0.15802 |
| 0.2 | 65.356681 | 65.374587 | 0.02739 | 65.394374 | 0.05764 | 65.364734 | 0.01232 |
| 0.3 | 74.816984 | 74.834867 | 0.02390 | 74.921210 | 0.13911 | 74.901423 | 0.11273 |
| 0.4 | 81.354064 | 81.504864 | 0.18502 | 81.431457 | 0.09504 | 81.401763 | 0.05860 |
| 0.5 | 86.753328 | 86.891211 | 0.15868 | 86.793134 | 0.04586 | 86.801475 | 0.05547 |
| 0.6 | 91.583318 | 91.716301 | 0.14499 | 91.613706 | 0.03317 | 91.654601 | 0.07777 |
| 0.7 | 96.026981 | 96.105346 | 0.08154 | 96.071341 | 0.04617 | 96.057820 | 0.03210 |
| 0.8 | 100.104906 | 100.115963 | 0.01104 | 100.110036 | 0.00512 | 100.113467 | 0.00855 |
| 0.9 | 103.737179 | 103.921347 | 0.17722 | 103.864136 | 0.12223 | 103.916316 | 0.12223 |

Table 8: Traffic speed and congestion delay index in Xiamen (2019.2.17, 16:25).

| Ranking | Business district | Congestion delay index | Speed (km/h) |
|---|---|---|---|
| 1 | Siming | 1.58 | 27.21 |
| 2 | Huli | 1.33 | 36.46 |
| 3 | Haicang | 1.25 | 44.82 |
| 4 | Jimei | 1.24 | 41.80 |
| 5 | Tongan | 1.13 | 47.74 |

Table 9: Traffic speed in Xiamen during 2019.2.17, 16:30–20:00, in km/h.

| Time | Siming | Huli | Haicang | Jimei | Tongan |
|---|---|---|---|---|---|
| 16:30 | 27.25 | 36.88 | 45.13 | 42.03 | 46.89 |
| 16:40 | 27.33 | 36.38 | 42.01 | 46.19 | 46.78 |
| 16:50 | 27.85 | 36.25 | 45.39 | 42.34 | 46.05 |
| 17:00 | 28.07 | 36.53 | 45.72 | 42.52 | 45.55 |
| 17:10 | 28.83 | 36.69 | 42.38 | 45.99 | 45.89 |
| 17:20 | 28.46 | 36.46 | 41.70 | 45.78 | 44.90 |
| 17:30 | 28.27 | 36.71 | 40.93 | 42.92 | 45.64 |
| 17:40 | 27.57 | 36.63 | 40.92 | 42.82 | 44.83 |
| 17:50 | 26.88 | 41.81 | 36.33 | 41.18 | 43.01 |
| 18:00 | 26.41 | 40.56 | 36.09 | 42.46 | 41.58 |
| 18:10 | 26.68 | 40.13 | 35.68 | 42.14 | 41.38 |
| 18:20 | 26.81 | 40.25 | 35.41 | 40.84 | 44.56 |
| 18:30 | 26.59 | 35.37 | 41.61 | 40.76 | 45.95 |
| 18:40 | 27.04 | 41.43 | 36.14 | 41.13 | 46.18 |
| 18:50 | 27.54 | 36.52 | 43.29 | 41.66 | 46.65 |
| 19:00 | 28.43 | 37.06 | 45.60 | 42.47 | 47.07 |
| 19:10 | 29.07 | 37.07 | 46.23 | 43.47 | 47.92 |
| 19:20 | 29.94 | 37.78 | 46.93 | 43.5 | 47.93 |
| 19:30 | 30.21 | 38.06 | 46.67 | 43.83 | 48.18 |
| 19:40 | 31.04 | 38.21 | 47.01 | 44.22 | 48.78 |
| 19:50 | 31.22 | 38.46 | 47.02 | 43.95 | 48.50 |
| 20:00 | 31.69 | 38.94 | 47.40 | 44.02 | 49.10 |

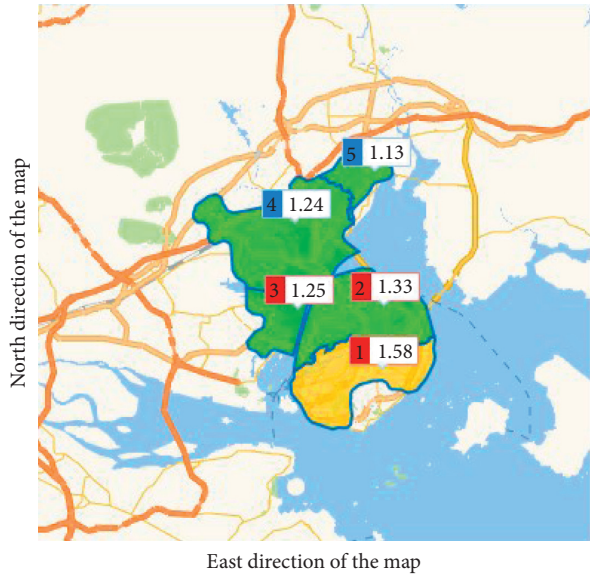


Figure 10: Xiamen map for real-time experiment (2019.2.17, 16:25).

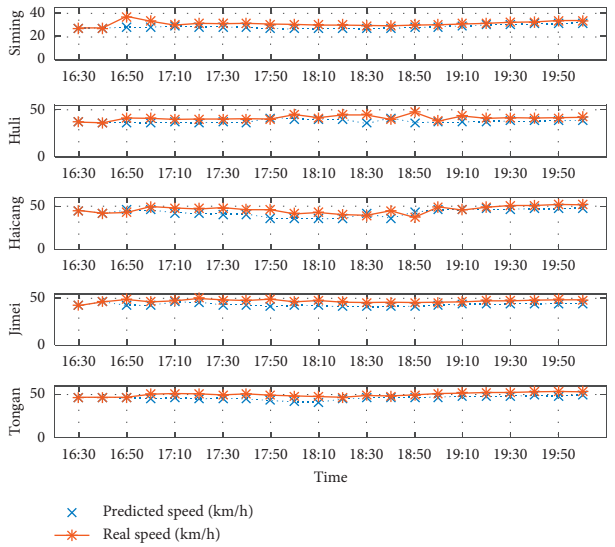


Figure 11: Traffic speed prediction in Huli using adaptive fractional Kalman prediction.

Table 10: Prediction error with different algorithms (km/h).

| Business district | AFKF | | FKF | | KF | |
|---|---|---|---|---|---|---|
| | MSE | MAPE | MSE | MAPE | MSE | MAPE |
| Siming | 1.4443 | 3.8773 | 1.9888 | 4.5995 | 2.6275 | 5.2535 |
| Huli | 2.7632 | 3.4929 | 2.7823 | 3.6189 | 3.1027 | 3.6632 |
| Haicang | 2.2912 | 3.4245 | 2.3334 | 3.4322 | 2.5653 | 3.4551 |
| Jimei | 1.7556 | 2.2908 | 1.7663 | 2.4931 | 2.6664 | 2.736 |
| Tongan | 1.4951 | 2.0732 | 1.9425 | 2.2198 | 1.9284 | 2.4390 |

ACO. The parameter settings of the algorithms are the same as those in Section 4.5.

Table 13 shows CACS is better than CA, CS, and ACO algorithms in solving TDVRPSC. With the increase of $p$, the minimal total expected time also increases. The total time also depends on the departure time. When the departure time is not the rush hour with traffic congestion, the total time is lower. When the departure time is the rush hour with

Table 11: Geographical location of Yuantong Express.

| No. | Area | Longitude (degree) | Latitude (degree) | Vertical coordinate (m) | Horizontal ordinate (m) |
|---|---|---|---|---|---|
| 0 | Siming | 118.076548 | 24.449664 | 00614776 | 2739976 |
| 1 | Siming | 118.086074 | 24.483964 | 00616305 | 2745670 |
| 2 | Siming | 118.138108 | 24.477468 | 00625298 | 2744930 |
| 3 | Siming | 118.17016 | 24.482423 | 00629818 | 2745249 |
| 4 | Huli | 118.142822 | 24.514657 | 00625437 | 2751505 |
| 5 | Huli | 118.178887 | 24.504187 | 00632221 | 2749519 |
| 6 | Huli | 118.147297 | 24.488977 | 00626737 | 2747251 |
| 7 | Huli | 118.144676 | 24.521974 | 00625959 | 2752476 |
| 8 | Jimei | 118.108407 | 24.610044 | 00620122 | 2768477 |
| 9 | Jimei | 118.10464 | 24.608536 | 00619048 | 2769188 |
| 10 | Tongan | 118.156155 | 24.726039 | 00627708 | 2790657 |
| 11 | Tongan | 118.146635 | 24.697452 | 00626189 | 2785614 |
| 12 | Tongan | 118.138162 | 24.708646 | 00624935 | 2787738 |
| 13 | Haicang | 118.017619 | 24.531057 | 00604864 | 2753869 |
| 14 | Haicang | 118.042758 | 24.498528 | 00608609 | 2748789 |
| 15 | Haicang | 118.015899 | 24.530205 | 00604389 | 2753588 |

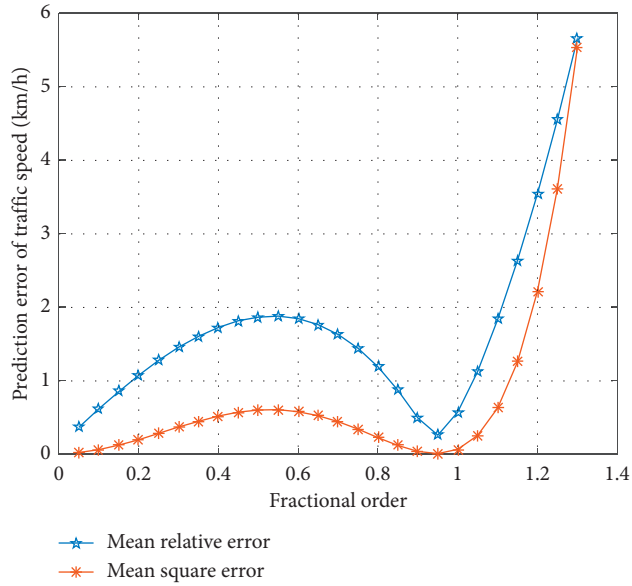


Figure 12: Prediction error of traffic speed with different fractional orders.

Table 12: Vehicle routing solution schemes for TDVRPSC.

| Route | Customer visiting sequence | Length (km) | Start | End | Time (min) |
|---|---|---|---|---|---|
| 1 | $0 \longrightarrow 8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 12 \longrightarrow 9 \longrightarrow 14 \longrightarrow 0$ | 109.6033 | 16:30 | 18:40 | 130 |
| 2 | $0 \longrightarrow 6 \longrightarrow 4 \longrightarrow 7 \longrightarrow 5 \longrightarrow 3 \longrightarrow 0$ | 47.3146 | 16:30 | 17:57 | 87 |
| 3 | $0 \longrightarrow 2 \longrightarrow 1 \longrightarrow 0$ | 26.5490 | 16:30 | 17:26 | 56 |
| 4 | $0 \longrightarrow 13 \longrightarrow 15 \longrightarrow 0$ | 34.7407 | 16:30 | 17:27 | 57 |

Table 13: Results with different $p$, different departure time, and different algorithms (min).

| $p$ (%) | Departure time | CACS | CA | CS | ACO |
|---|---|---|---|---|---|
| 25 | 16:30 | 223 | 228 | 225 | 226 |
| | 17:30 | 231 | 237 | 234 | 236 |
| 50 | 16:30 | 276 | 282 | 279 | 280 |
| | 17:30 | 286 | 281 | 277 | 279 |
| 75 | 16:30 | 312 | 318 | 314 | 317 |
| | 17:30 | 324 | 329 | 326 | 328 |
| 100 | 16:30 | 330 | 337 | 333 | 336 |
| | 17:30 | 343 | 349 | 345 | 347 |

traffic congestion, the minimal total expected time will increase.

Compared with other kinds of cultural algorithms, such as adaptive immune clonal selection cultural algorithm, the convergence rate of Cuckoo Search algorithm is not sensitive to the parameters used. Therefore, adaptive Cultural Algorithm Based Cuckoo Search has fewer parameters to be fine-tuned in Cuckoo Search and is more robust for optimization problems.

## 5. Conclusion

The main contributions of this paper are summarized as follows:

(1) An adaptive CACS has been proposed and its convergence is proved.

(2) AFKF for traffic speed prediction is proposed. An adaptive mechanism is adopted with the covariance of prediction error. Its mathematical process is proved.

(3) New solutions of VRP are discovered, which are better than the published solutions.

(4) CACS and AFKF are used for TDVRPSC with real-time urban vehicle data. Traffic speed is predicted and vehicle routing can be adjusted in time to avoid congested roads to reduce transportation time and cost.

The next step is to study better TDVRPSC models and design new methods to deal with uncertainties.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The author declares that there are no conflicts of interest.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 51579114, the Natural Science Foundation of Fujian Province under Grant 2018J05085, and High Level Scientific Research Project of Transportation Engineering (202003).

## References

[1] Y.-N. Guo, J. Cheng, S. Luo, D. Gong, and Y. Xue, "Robust dynamic multi-objective vehicle routing optimization method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1891–1903, 2018.

[2] Y.-n. Guo, H. Wang, and J. Cheng, "Adaptive immune clonal selection cultural algorithm," *Acta Electronica Sinica*, vol. 38, no. 4, pp. 966–972, 2010.

[3] Y.-n. Guo, J. Cheng, Y.-y. Cao, and Y. Lin, "A novel multi-population cultural algorithm adopting knowledge migration," *Soft Computing*, vol. 15, no. 5, pp. 897–905, 2011.

[4] N. Geng, Q. Meng, D. Gong, P. W. H. Chung, and H. Chung, "How good are distributed allocation algorithms for solving urban search and rescue problems? a comparative study with centralized algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 478–485, 2019.

[5] W. R. Stewart and B. L. Golden, "Stochastic vehicle routing: a comprehensive approach," *European Journal of Operational Research*, vol. 14, no. 4, pp. 371–385, 1983.

[6] D. J. Bertsimas, "A vehicle routing problem with stochastic demand," *Operations Research*, vol. 40, no. 3, pp. 574–585, 1992.

[7] L. Bianchi, M. Birattari, M. Chiarandini et al., "Hybrid metaheuristics for the vehicle routing problem with stochastic demands," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, pp. 91–110, 2006.

[8] A. Bozorgi-Amiri, M. S. Jabalameli, and S. M. J. Mirzapour Al-e-Hashem, "A multi-objective robust stochastic programming model for disaster relief logistics under uncertainty," *OR Spectrum*, vol. 35, no. 4, pp. 905–933, 2013.

[9] I. Sungur, F. Ordóñez, and M. Dessouky, "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty," *IIE Transactions*, vol. 40, no. 5, pp. 509–523, 2008.

[10] K. C. Tan, C. Y. Cheong, and C. K. Goh, "Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation," *European Journal of Operational Research*, vol. 177, no. 2, pp. 813–839, 2007.

[11] O. J. Ibarra-Rojas, L. Hernandez, and L. Ozuna, "The accessibility vehicle routing problem," *Journal of Cleaner Production*, vol. 172, pp. 1514–1528, 2018.

[12] R. D. Wollmer, "Two stage linear programming under uncertainty with 0-1 integer first stage variables," *Mathematical Programming*, vol. 19, no. 1, pp. 279–288, 1980.

[13] A. S. Kenyon and D. P. Morton, "Stochastic vehicle routing with random travel times," *Transportation Science*, vol. 37, no. 1, pp. 69–82, 2003.

[14] D. Alpago, M. Zorzi, and A. Ferrante, "Identification of sparse reciprocal graphical models," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 2475–1456, 2018.

[15] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," *World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 210–214, IEEE Publications, New York, NY, USA, 2009.

[16] M. Mareli and B. Twala, "An adaptive Cuckoo search algorithm for optimisation," *Applied Computing and Informatics*, vol. 14, no. 2, pp. 107–115, 2018.

[17] S. Ishak Boushaki, N. Kamel, and O. Bendjeghaba, "A new Quantum Chaotic Cuckoo search algorithm for data clustering," *Expert Systems with Applications*, vol. 96, pp. 358–372, 2018.

[18] M. Ayoubi and R.-A. Hooshmand, "A new fuzzy optimal allocation of detuned passive filters based on a Nonhomogeneous Cuckoo search algorithm considering resonance constraint," *ISA Transactions*, vol. 89, pp. 186–197, 2019.

[19] M. Z. Ali, N. H. Awad, R. G. Reynolds, and P. N. Suganthan, "A balanced fuzzy Cultural Algorithm with a modified Levy flight search for real parameter optimization," *Information Sciences*, vol. 447, pp. 12–35, 2018.

[20] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[21] X. Liu, H. Qu, J. Zhao, and P. Yue, "Maximum correntropy square-root cubature Kalman filter with application to SINS/GPS integrated systems," *ISA Transactions*, vol. 80, pp. 195–202, 2018.

[22] S. San Gultekin and J. Paisley, "Nonlinear Kalman filtering with divergence minimization," *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6319–6331, 2017.
[23] J. E. Solís-Pérez, J. F. Gómez-Aguilar, L. Torres, R. F. Escobar-Jiménez, and J. Reyes-Reyes, "Fitting of experimental data using a fractional Kalman-like observer," *ISA Transactions*, vol. 88, pp. 153–169, 2019.
[24] Y. Sun, X. Wu, J. Cao, Z. Wei, and G. Sun, "Fractional extended Kalman filtering for non-linear fractional system with Lévy noises," *IET Control Theory & Applications*, vol. 11, no. 3, pp. 349–358, 2017.
[25] T. Liu, S. Cheng, Y. Wei, A. Li, and Y. Wang, "Fractional central difference Kalman filter with unknown prior information," *Signal Processing*, vol. 154, pp. 294–303, 2019.
[26] X.-S. Yang, *Cuckoo Search and Firefly Algorithm: Theory and Applications*, Springer, London, UK, 2014.
[27] Augerat, *Known Best Results, [EB/OL]*, 2013, http://neo.lcc.uma.es/vrp/known-best-results/.
[28] R. Sagban, K. Ruhana Ku-Mahamud, and M. Shahbani Abu Bakar, "Nature-inspired parameter controllers for ACO-based reactive search," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 11, no. 1, pp. 109–117, 2015.
[29] Auto Navi Map, *Real-time City Details, [EB/OL]*, 2019, https://report.amap.com/detail.do?city=350200.