Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 5728042, 11 pages
https://doi.org/10.1155/2018/5728042

Research Article

High-Speed Data-Driven Methodology for Real-Time Traffic Flow Predictions: Practical Applications of ITS

Hyun-ho Chang[1] **and Byoung-jo Yoon**[2]

[1]*School of Environmental Studies, Seoul National University, Seoul, Republic of Korea*
[2]*Department of Urban Engineering, Incheon National University, Incheon, Republic of Korea*

Correspondence should be addressed to Byoung-jo Yoon; bjyoon63@inu.ac.kr

Received 18 August 2017; Revised 28 February 2018; Accepted 8 March 2018; Published 24 April 2018

Academic Editor: Anastasios Kouvelas

Despite the achievements of academic research on data-driven *k*-nearest neighbour nonparametric regression (KNN-NPR), the low-speed computational capability of the KNN-NPR method, which can occur during searches involving enormous amounts of historical data, remains a major obstacle to improvements of real-system applications. To overcome this critical issue successfully, a high-speed KNN-NPR framework, capable of generating short-term traffic volume predictions, is proposed in this study. The proposed method is based on a two-step search algorithm, which has the two roles of building promising candidates for input data during nonprediction times and identifying decision-making input data for instantaneous predictions at the prediction point. To prove the efficacy of the proposed model, an experimental test was conducted with large-size traffic volume data. It was found that the performance of the model not only at least equals that of linear-search-based KNN-NPR in terms of prediction accuracy, but also shows a substantially reduced execution time in approximating real-time applications. This result suggests that the proposed algorithm can be also effectively employed as a preprocess to select useful past cases for advanced learning-based forecasting models.

1. Introduction

In intelligent transport systems (ITS), vehicular traffic variables such as volumes, speeds, travel times, and occupancies collected by traffic monitoring devices are not current but are data which were temporally collected previously. To produce accurate real-time or future traffic information, a prediction system is essential as one of the subsystems of ITS, and it is natural for the core of the system to be a reliable prediction model. As such, short-term prediction modelling is a crucial and attractive topic in ITS research.

In modern ITS, the popular introduction of advanced data management systems has made tremendous quantities of traffic variable data available. The wealth of traffic data, in turn, has rendered promising opportunities to data-driven forecasting approaches, one of which is data-driven *k*-nearest neighbour nonparametric regression (KNN-NPR), as KNN-NPR has its roots in strong pattern recognition, that is, the diversity of the patterns included in rich historical data [1]. KNN-NPR is also called the *k*-nearest neighbours (KNN) method [2], as it essentially employs the *k*-NN strategy to mine the neighbourhoods directly, that is, *k* cases similar to a current case, which in turn is used to understand the temporal evolution of current states [1–3]. Despite the academic and practical advantages [3] of KNN predictions as discussed in the literature review here, a chronic weakness of data-driven KNN in time-critical systems has been the long execution time [4, 5], mainly required to search for past observations, that is, KNN, which are similar to current observations. This computational issue of KNN should therefore be addressed thoroughly before real-world applications can be realized.

To overcome the shortcomings of data-driven KNN as mentioned above, a high-speed KNN framework for predicting short-term traffic flows is proposed in this article. The method is developed based on a two-step data search algorithm, which is designated for two separate roles to find candidate input data similar to the current state included in historical data during nonprediction times and then

instantaneously to identify decision-making input data for predictions from the candidate input data at the forecasting point. The performances of the prediction algorithm are proven through a prediction simulation with real-world motorway traffic volume data, and certain findings are discussed from academic and practical perspectives.

2. Backgrounds

There is a predominant notion that the evolutions of time-series traffic flow states naturally are chaotic systems [1, 6] in which the development of states is sensitively determined given certain initial conditions. KNN has its theoretical foundation in chaos theory in the area of time-series pattern recognition [1, 2], as follows: the decision with regard to future states made by a KNN predictor has the asymptotically minimum risk level, as $n \to \infty$, $k \to \infty$ with $k/n \to 0$, where n and k are possible patterns included in historical data and nearest patterns (similar) to the current time-series state, respectively. Particularly, the KNN approach basically relies on the bulk of knowledge [3] included in the historical data to understand the relationship between the input and the output variables [2, 4] without any statistical assumptions [2], as opposed to statistical knowledge inferred by man-made artificial formulae [2–4].

Due to its theoretical and practical advantages, KNN has developed into a promising approach for ITS forecast modelling. (Note that research on traffic volume predictions based on data-driven KNN is the focus here; several wide-ranging reviews [7, 8] of various forecasting approaches for traffic variables are recommended for more interested readers.) The KNN strategy for nonlinear clustering was extended to a time-series analysis as a data-driven KNN method [2], after which it was also extended to multi-interval forecasting [1, 3, 9, 10] and improved for multivariate estimations [11, 12]. To enhance the prediction accuracy, academic efforts are still realized with sophisticated similarity measures [9, 13] and/or updated forecasting functions [1, 2, 13]. It was also reported in comparative studies [1–3, 9, 12] that the performances of KNN-based methods in terms of prediction reliability are at least comparable to those of parametric and/or nonlinear models.

Despite these achievements, the weakness of data-driven KNN has been its slow execution time [4, 5] from the perspective of time-critical systems such as dynamic ITS information systems. A majority of the run time is spent searching for past cases (similar to the current case) included in rich historical data, as a linear search is essential to build the best group of past cases. To address this problem, several techniques to reduce the search time can be categorized into two approaches: advanced search technology [4] and data-segmentation methods [3, 5, 10, 12]. As for advanced search technology, an imprecise computational method based on approximate nearest neighbour (ANN) searching supported by an advanced data management system (ADMS) was reported in [4], where searching time was reduced to the degree of 44.0~67% with an acceptable level of +1% prediction error. However, additional academic studies supported by ADMS have not been reported in the area of ITS forecasting, since ADMS that is composed of advanced hardware and software devices is deeply related to tight budgets, which is another hindrance in practice. The data-segmentation method narrows the entire historical data down to useful data with the assumption that the temporal variation of traffic flow is recurrent within a time span of a day, an hour, or even several minutes. This can be efficacy, as searching time is proportional to the size of searched data in the case of a linear search. A useful sector of a historical database is time-dependently constrained within a fixed window of the present time interval $\pm\beta$ [3, 12] or the type of day [10] to reduce the load when searching. The β-based methods reduce a size of historical data to $(2\beta + 1)/S_n$, where S_n is the number of time sequences in a day (see Figure 1(a)). These fixed-window methods are useful for searching time but they cannot yield reliable predictions. To handle this problem effectively, a flexible segmentation method to preset a useful segment of the database was proposed in [5]. The flexible segmentation is preset through a sort of KNN prediction simulation using similarity ranking and prediction-error ranking, and each segment for a time interval has an s-size array that includes a ranking score (z, 0.0~1.0). If a ranking score (z_i) for s_i ($s_i \in S$) is more than α value (0.0~1.0), then each historical data corresponding to the s_i sequence is searched in the searching step of a standard KNN (SKNN) (see Figure 1(b)). In addition, the procedure to predetermine flexible segmentation is very time-expensive and should be executed offline to update segmentation information on a monthly or weekly basis. Most importantly, the data-segmentation approach cannot guarantee the reliability of prediction, when the temporal variation of traffic volume is nonrecurrent (see Figure 2(b)). The temporal pattern of traffic flow deviates from the notions of periodicity (monthly, daily, or hourly) [14]. Despite these efforts, it is clear that the execution time of the data-driven KNN forecasting algorithm is not comparable to those of high-speed real-time models. In addition, a KNN algorithm has still not been reported, which can dynamically and effectively predetermine a small member of promising past cases by reflecting current traffic flow states under the conditions of conventional ITS systems.

Based on the literature review, the performance of KNN has reached an acceptable level of prediction accuracy in virtue of academic efforts toward refined modelling. However, it appears that the execution time of the data-driven KNN algorithm had yet to progress sufficiently. Most importantly, the framework of the KNN forecasting algorithm inevitably includes a search process, which mainly contributes to its long execution time. For this reason, the KNN forecasting procedure can be a bottleneck in dynamic information flows in conventional ITS systems that are not supported with any advanced data management or search technologies.

Unquestionably, the "bigger data and slower running" problem associated with KNN-based forecasting remains an ongoing issue to be addressed successfully and urgently. To make matters worse, the volume of historical data available continues to grow in modern ITS. Hence, a high-speed framework for KNN algorithms is necessary, and

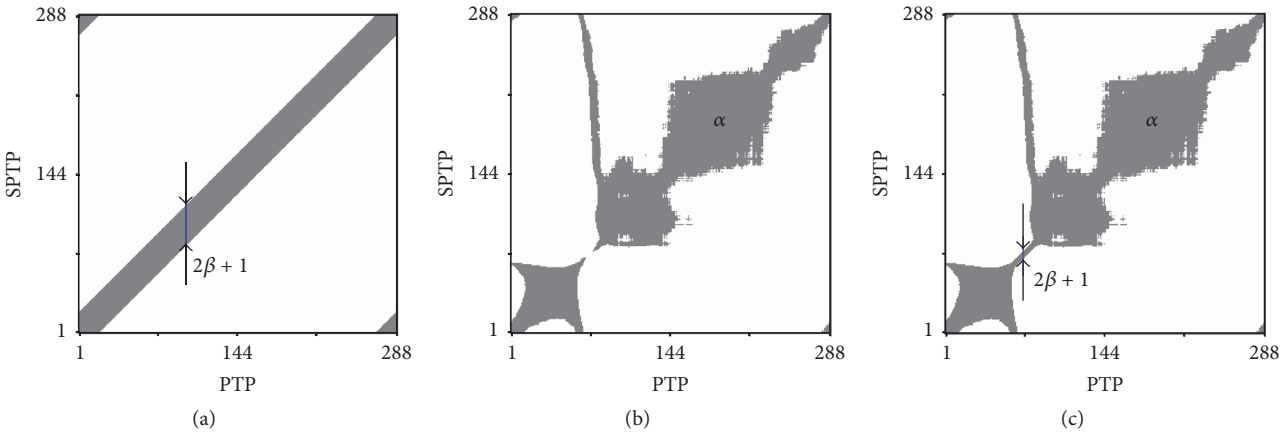

FIGURE 1: Static data segments used for forecasting. (a) KNN-D, (b) KNN-U, and (c) KNN-UD. Note: gray area is used segment, PTP (prediction time point, 5 min), and SPTP (selected past time point, 5 min).

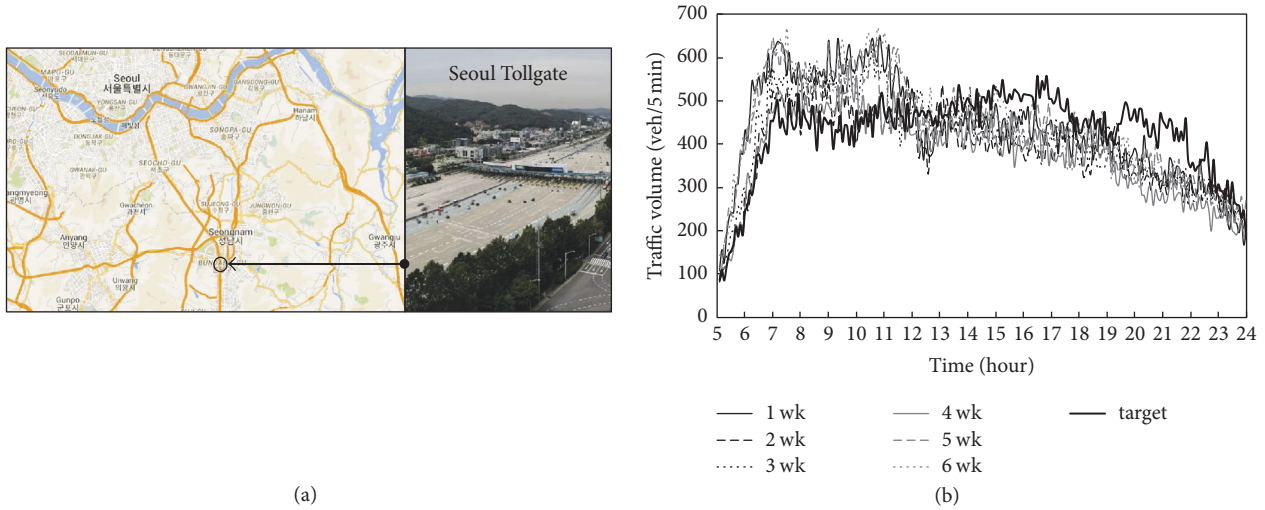

FIGURE 2: Information of target traffic volume data. (a) Test bed (source: Google Maps) and (b) temporal state evolution of the target data.

this represents another challenge. In addition, a high-speed process to search and identify similar cases from tremendous historical data is necessary as a learning step of advanced forecasting models such as a support vector machine or deep learning.

3. High-Speed Framework for KNN Prediction

3.1. Concept. The aim of this study is to develop a high-speed framework of KNN-based prediction while considering partial current traffic flow states in order to extremely speed up a KNN prediction method and guarantee its prediction accuracy. The slow running issue of KNN algorithms can be surmounted simply by excluding the search process of KNN from the prediction algorithm. However, a new problem would be how to find and determine the potential NN to generate the forecast within a given time limit, $\varepsilon_r \rightarrow +0.0$, at prediction point t_p. Note that the structure of a KNN prediction algorithm is divided into two main parts: a search process to find KNN similar to the current states from the historical database and a forecast function to generate future states with the information of KNN.

A key clue to solving the problem exists in the basic premise of forecast modelling, which holds that the current temporal evolution of states is closely related to subsequent states in some way. Similarly, the relationship between the previous temporal development of states and the present state also follows this premise. Based on this self-evident relationship, incomplete current temporal states which do not include the present state can be effectively used to find promising instances of NN from a historical database with a similarity measure during nonprediction times (i.e., the length of the time interval (t_l)—ε_r), after which the exact NN to the complete current temporal states with acceptable differences, when the present state is collected, can then be instantly identified from a dataset of promising instances of NN during the process of the prediction algorithm within ε_r at t_p.

In this study, each of two-step search procedures above is combined with the framework of the KNN prediction algorithm. The algorithm minus the forecast generation

aspect is used as the first step of the search process to discern promising k-nearest neighbours (k_p-NN) from a historical database during the nonprediction time, after which the algorithm is again employed to select instantly the optimal k-nearest neighbours (k_o-NN) from the updated k_p-NN dataset using the second step of the search and then to generate future states at t_p, where $0 < k_o \le k_p$, $k_p = f \times k_o$, and $f \ge 1.0$. Additionally, the k_p-NN dataset is updated simultaneously when the present state is collected. In this way, the search process of k-NN, the main cause of the long execution time of KNN, is significantly excluded from the algorithm, whereas the SKNN algorithm can be ensured as k_o-NN.

3.2. Fundamental Components. A KNN prediction model is typically composed of three fundamental components: (a) three state vectors (current, input, and output), (b) a similarity measure, and (c) a forecast function (FF) or algorithm. The high-speed framework proposed in this study was developed based on the general framework of the SKNN prediction algorithm. Under this consideration, various KNN models can be easily integrated into the high-speed framework. Each of the three components is defined as shown below, and the defined components are combined into the forecasting algorithm in Section 3.3.

In discrete systems, continuous time is divided by a fixed length of the time interval t_l; this is represented in the form of a time series, $[(t), (t-1), \ldots, (t-d)]$, at the present time interval (t) toward the past, where d is a suitable number of lags, that is, the embedding size. In our case, system states are traffic volume measurements, and the temporal development of the traffic volume states corresponding to the form of the time series is defined as the current state vector, $x_c(t)$, with

$$x_c(t) = \left[q(t), q(t-1), \ldots, q(t-d)\right]. \tag{1}$$

In this equation, $q(t)$ is the measurement during the time interval (t), $q(t-1)$ is the measurement during the time interval $(t-1)$, and so on. A one-dimensional univariate state vector is used in this study despite the fact that several types of state vectors are available for various purposes in KNN studies [1, 2].

To find and discern past cases similar to $x_c(t)$ from the historical database that consists of n past cases, the input state vector and the output state vector are necessary. The jth input state vector, $x_j(\tau)$, and output vector, o_j, at the past time interval (τ) are defined as (2) and (3), respectively, with $j \in n$. Additionally, the dimension of $x_j(\tau)$ is equal to that of $x_c(t)$.

$$x_j(\tau) = \left[q_j(\tau), q_j(\tau-1), \ldots, q_j(\tau-d)\right] \tag{2}$$

$$o_j = \left[u_j, q_j(\tau+1)\right]. \tag{3}$$

Here, τ is the (past) running-time index ($\tau < t-1$); o_j is associated with $x_j(\tau)$; u_j is the state distance, calculated with the Euclidean metric, between $x_c(t)$ and $x_j(\tau)$; and $q_j(\tau+1)$ is the past traffic volume record at time interval $(\tau+1)$.

To update and determine k-NN during the search process of the algorithm, various similarity measures can be used. In our case, the Euclidean distance is employed to calculate the degree of nearness between $x_c(t)$ and $x_j(\tau)$. Given the time-series dimension (l, d), $l = \{0, 1\}$, the state distance is defined as $u_j(l, d)$ with (4). In this way, the two types of state distances, that is, $u_j(1, d)$ and $u_j(0, d)$, are correspondingly utilized to determine the k_p-NN and k_o-NN datasets in the two-step search procedure of the proposed algorithm.

$$u_j(l, d) = \left[\sum_{s=l}^{d} \left|q_c(t-s) - q_j(\tau-s)\right|^2\right]^{1/2}. \tag{4}$$

The three defined vectors and the distance metric, which are combined with the search function, play a role in the building of the information of k-NN, which is composed of k [input → output] objects, where $k = \{k_p, k_o\}$, $0 < k_o \le k_p$, $k_p = \alpha \times k_o$, and $\alpha \ge 1.0$. The k-objects are $x_i(\tau)$ and $o_i = [u_i(l, d), q_i(\tau+1)]$, where $i = 1, 2, \ldots, k$. To record the k-object information, a data structure is required. In this study, the data structure of k-NN is therefore defined for two functions: to record the k_p objects discerned in the historical database with $u_j(1, d)$ during nonprediction time and then instantaneously to determine the k_o-objects from the k_p-objects with $u_j(0, d)$ at $t_p$shown as follows.

Bilevel Data Structure for the Input and Output Vectors

$$\begin{array}{lccc} & \text{Input} & & \text{Output} \\ [i] & [k\text{-nearest neighbors}, x_i(\tau)] & & [k\text{-output vectors}, o_i] \\ 1 & [q_1(\tau), q_1(\tau-1), \ldots, q_1(\tau-d)] & \longrightarrow & [q_1(\tau+1), u_1(l,d)] \\ 2 & [q_2(\tau), q_2(\tau-1), \ldots, q_2(\tau-d)] & \longrightarrow & [q_2(\tau+1), u_2(l,d)] \\ \cdots & \cdots & & \cdots \\ k_o & [q_{k_o}(\tau), q_{k_o}(\tau-1), \ldots, q_{k_o}(\tau-d)] & \longrightarrow & [q_{k_o}(\tau+1), u_{k_o}(l,d)] \\ \cdots & \cdots & & \cdots \\ k_p & [q_{k_p}(\tau), q_{k_p}(\tau-1), \ldots, q_{k_p}(\tau-d)] & \longrightarrow & [q_{k_p}(\tau+1), u_{k_p}(l,d)]. \end{array} \tag{5}$$

Once the dataset of the k_o-objects has been selected, the last step is to estimate the future states with FF. A weighted function by the inverse of the state distance that was applied in previous studies [1–3, 10, 12] is improved and used as (6) with $l = 0$ in this work. This approach is based on the assumption that a more similar experience can lead to more reliable decisions about future uncertain states. Additionally, the FF with $l = 1$ can generate predictions when the present state, that is, $q(t)$, is not available at t_p due to a temporary failure or delay in the transmission of data from field-detector stations to a data center.

$$\widehat{q}(t+1) = \frac{\sum_{i=1}^{k_o} \left(q(\tau+1)/u_i(l,d)\right)}{\sum_{i=1}^{k_o} \left(1/u_i(l,d)\right)}. \tag{6}$$

Here, $\widehat{q}(t+1)$ is the estimated traffic volume for the future time interval $(t+1)$, $u_i(l,d) > 0.0$, and $l = \{0, 1\}$.

3.3. Prediction Algorithm. The three aforementioned components and the input-and-output data structure are integrated into the two-step search (S2S) KNN algorithm as shown in Algorithm 1. The KNN-S2S algorithm is designed for two main purposes: to build k_p-NN (i.e., k_p-objects) during the nonprediction time between the previous $t_p + \varepsilon_r$ and t_p and then to determine k_o-NN (i.e., k_o-objects) and generate $\widehat{q}(t+1)$ during the given prediction time between t_p and $t_p + \varepsilon_r$.

The first step consists of three substeps: (1) initialization, (2) searching and building k_p-NN, and (3) predetermining the potential k_o-NN. In substep (2), various techniques from a linear to an advanced search can be employed to discern object candidates (i.e., $x_j(\tau)$ and o_j) from the historical database, which is beyond the scope of this research. Substep (3) plays a role in predetermining the potential k_o-NN based on the selected k_p-NN, which also reduces the execution time used to determine k_o-NN in the second step.

The second step is composed of four substeps: (1) updating $x_c(t)$, (2) recalculating the state distance, (3) selecting k_o-NN from k_p-NN, and (4) prediction generation. The additional information is updated through substeps (1)-(2), after which (promising) k_o-NN instances are determined in substep (3). These processes are carried out almost instantly without any computational work for accessing and searching through vast amounts of historical data. Finally, the prediction generation is conducted based on the selected k_o-NN using FF.

4. Evaluation and Findings

4.1. Design of Experiments. The efficacy of the proposed KNN-S2S algorithm was examined through an experimental study that had three aspects: the features of the test data, the selection of benchmark models, and the selection of performance measures.

The one-year traffic volume data used in this study were collected from a toll collection system with a five-minute aggregation in 2016. The test bed was a tollgate near Seoul, South Korea, located on motorway #1 (Figure 2(a)). The data array size was 105,408 measurements (288 intervals per day for one year = 288 × 366). The target day for the prediction simulation was the last day of the year (a Saturday); thus, the states of the target data exhibit different temporal developments compared to those of the six previous Saturdays (Figure 2(b)). Noticeably, the lower and upper states during two time periods (05:00–12:00 and 12:00–24:00) deviate from the corresponding recurrent patterns. Due to these nonrecurrent conditions, the target state can be the worst case for KNN-based methods, the prediction accuracy of which depends strongly on the selection of similar patterns.

If the data segment is correct, the accuracy performances of a KNN model either using the data segment or using entire data should be at least comparable. To examine this, the SKNN algorithm widely applied in related works [1–5] is selected as a basic model. The SKNN algorithm, which consists of three steps (initialization, the search for and the building of k-NN, and forecast generation), is executed at t_p. In our study, steps (1) and (2) of SKNN are identical to substeps (1) and (2) of KNN-S2S step (1), respectively, with $x_c(t)$, $l = 0$, and $k = k_o$. Then, the future estimation is generated with the FF defined as (6). A linear search technique that checks each case of a list sequentially until all cases have been searched was used to find the best k-NN in this study. In this way, the performance outcomes of SKNN with the best k-NN information can provide a baseline for the evaluation of the potentialities of KNN-S2S. Note that substep (3) of KNN-S2S step (1) for k_o objects was added to SKNN and KNN-S2S before the forecast generation step for the respective case for analysis purposes, although it is not necessary in actual use. Unfortunately, a dynamic segmentation method similar to the method proposed in this study has not been reported in ITS forecasting research. Therefore, three static segmentation methods (i.e., fixed-window (KNN-D) and flexible segmentation (KNN-U) methods and a combination of KNN-D and U (KNN-UD)) were selected to demonstrate the robustness of the proposed method. Note that advanced readers can refer to the three compared methods in [5].

Based on the features of the target system and the analytical necessities of the five models, the following five performance measures were carefully chosen. The mean absolute percentage error (MAPE) and the root-mean-squared error (RMSE) defined in (7) and (8), respectively, were used for an analysis of the prediction accuracy. The relative percentage error (RPE) was selected as two types, the mean (MPRE) and standard deviation (SDRPE) of RPE [$= (\widehat{q}_i - q_i)/q_i \times 100.0$], to check the conformity of predictions between entire data and segmented data. To examine the searching speed, data usage [= used data/entire data × 100.0] was also used as soft computing.

$$\text{MAPE}(\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\widehat{q}_i - q_i\right|}{q_i} \times 100, \quad q_i > 0 \tag{7}$$

$$\text{RMSE}(\text{veh}) = \left(\frac{1}{N}\sum_{i=1}^{N}\left|\widehat{q}_i - q_i\right|^2\right)^{0.5}. \tag{8}$$

Here, N is the number of test samples and q_i and $\widehat{q}_i$ represent the actual volume and the predicted volume of sample i, respectively, where $i \in n$.

Given the $x_c(t)$ without $q(t)$, k_o and k_p values, d value:
(where $1 < k_o < k_p$; $k_p/n \to 0.0$)
Step 1: Build k_p-NN
If (the previous $t_p + \varepsilon_r$) $< \tau_{nw} < t_p$ then l-value = 1
(where τ_{nw} is the present running time; ε_r is a run-time limit)
(1) Initialize the list of the k_p-objects shown as (5).
(2) For each $x_j(\tau)$ and o_j in the past database
(where $j = 1, 2, \ldots, n$; $o_j = [q_j(\tau + 1), u_j(l, d)]$)
(2-1) Calculate $u_j(l, d)$ between $x_c(t)$ and $x_j(\tau)$ by eq. (4)
(2-2) If $u_j(l, d) < u_{\max}$ then
(where $u_{\max} = \max\{u_1(l, d), u_2(l, d), \ldots, u_{k_p}(l, d)\}$)
(2-2-1) Delete $x_i(\tau)$ and o_i from the k_p-object list
(where o_i is associated with $u_{\max}$, $1 \le i \le k_p$)
(2-2-2) Update $x_j(\tau)$ and o_j into the k_p-object list
(2-2-3) Search $u_{\max}$ in the updated k_p-object list
(3) Sort the select k_p-objects ascending based on $u_i(l, d)$
Step 2: Decide k_o-NN and estimate $\widehat{q}(t + 1)$
If $t_p \le \tau_{nw} \le (t_p + \varepsilon_r)$ then
If $q(t)$ is available then l-value = 0
(1) Add $q(t)$ to $x_c(t)$
(2) Recalculate all $u_j(l, d)$ in the k_p-object list as follows
(where $j = 1, 2, \ldots, k_p$)

$$u_j(l, d) = \left[\left\{u_j(1, d)\right\}^2 + \left\{q(t) - q_j(\tau)\right\}^2\right]^{1/2}$$

(3) For each $x_j(\tau)$ and o_j in the k_p- object list
(where $j = k_o + 1, k_o + 2, \ldots, k_p$)
If $u_j(l, d) < u_{\max}$ then
(where $u_{\max} = \max\{u_1(l, d), u_2(l, d), \ldots, u_{k_o}(l, d)\}$)
(3-1) Delete $x_i(\tau)$ and o_i from the k_o-object list
(where o_i is associated with $u_{\max}$, $1 \le i \le k_o$)
(3-2) Update $x_j(\tau)$ and o_j into the k_o- object list
(3-3) Search $u_{\max}$ in the updated k_o-object list
Else l-value = 1
(4) Estimate $\widehat{q}(t + 1)$ by eq. (6)

Algorithm 1: Pseudocode for the KNN-S2S prediction algorithm.

4.2. Parameter Analysis and Findings. The efficacy of SKNN is achieved through pattern recognition, which is wholly reliant on both the d value of state space and the k value of the nearest neighbours. With regard to the d value (i.e., the embedding size) to construct the temporal properties of $x_c(t)$, Takens' definition of $d \ge 2D+1$ with a given D-dimension state space value [1–3] is widely referenced as a minimal embedding size. Thus far, determining a suitable d value mainly depends on the properties of the temporal states of the target system. With regard to the k value, the theoretical condition of KNN (i.e., $n \to \infty, k \to \infty$ with $k/n \to 0$) is restricted in actuality. Past cases (n) included in the available historical data are limited, and a suitable k value should therefore be finite to meet the condition of $k/n \to 0$. The two parameters are also closely related to each other. In this context, the best or optimal values of the two parameters to ensure reliable estimations should be concurrently analysed [1, 3] and determined in advance. As such, a prediction simulation [1–3, 5, 10, 12, 13] was also used in this work in order to determine suitable parameter values for the two algorithms (SKNN and KNN-S2S).

The SKNN algorithm was experimentally executed to estimate the target data with all combinations of possible parameter values (1,000 cases = d-values [1–10] × k-values [1–100]), and the prediction error of each case was analysed with MAPE. The effects of the two parameters on the prediction errors are shown in Figure 3(a). The best d and k values, $d_o = 5$ and $k_o = 41$, were determined under the condition in which MAPE is minimized to 6.172.

With regard to the d value, the overall MAPEs decrease to the minimal error space and then increase as the d value increases, and the k value exceeds 30. The values of $d_o \pm 2$ are acceptable as optimal d values within the minimal error rate of +0.15%. Specifically, the optimal d values to guarantee reliable estimations satisfy Takens' definition (i.e., a minimum threshold), whereas a maximal threshold also exists. This fact indicates that the temporal state evolution of the optimal d-size is closely related to the pattern recognition process of KNN for future states in some way, and therefore this state of the optimal $d - 1$ size can, at least, play a role comparable to that of the optimal d-size with an acceptable margin of error.

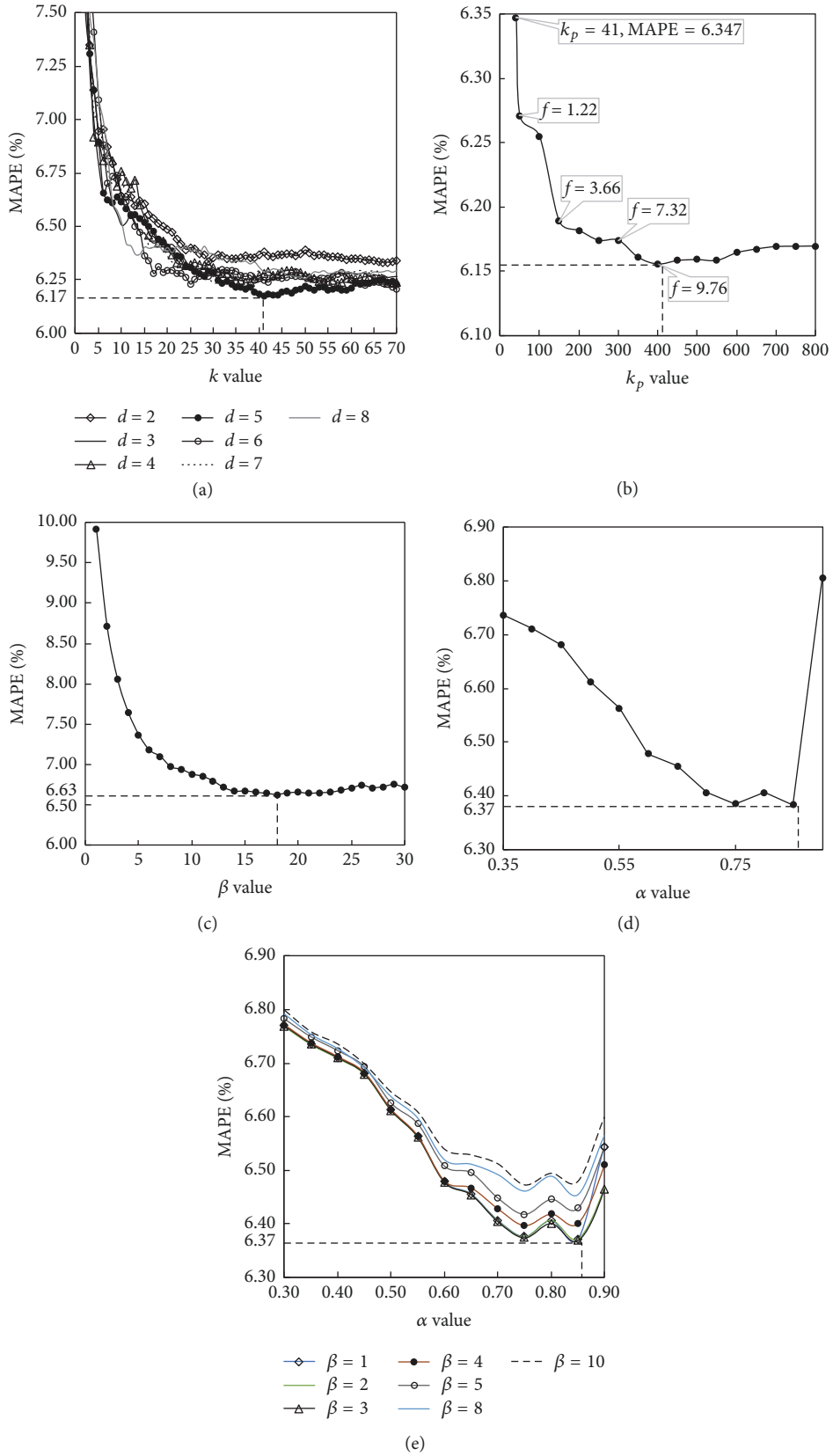

FIGURE 3: Effects of model parameters on prediction errors. (a) Effects of d and k values; (b) effects of the k_p value with d_o and k_o values; (c) effects of the β value; (d) effects of the α value; and (e) effects of α and β values.

With regard to the k value when $d_o = 5$, MAPE decreases exponentially to the minimum point at $k_o = 41$ and then gradually increases with little variation as the k value increases. In this way, the k_o value of 41 satisfies $k/n \to 0$ with 41/105,120 = 0.00039. This essentially implies that a clustered pattern exists in past cases regardless of whether the boundaries between patterns are explicit. The steep decrement of MAPE also provides evidence that the convergence process quickly reaches the geometric centroid of past future states associated with the selected pattern. Additionally, it was noted that suitable values of the two parameters can easily be recalibrated and updated on a periodic time basis (daily, weekly, or even monthly) in advance [1–3, 10]. Furthermore, the d_o value can be fixed after it has been determined through experimental tests [1, 3].

The KNN-S2S algorithm was also conducted with the identified d_o and k_o values, and its performance according to the k_p values ranging from 50 to 800 in increments of 50 is shown in Figure 3(b). When $k_p = k_o$, the prediction error of KNN-S2S guarantees the best case of SKNN within +0.2%. This directly indicates that $x_c(t)$ with $l = 1$ is closely linked to future states and can therefore play an effective role in the selection of k_p-NN, which includes the best k-NN according to $x_c(t)$ with $l = 0$. The MAPEs decrease steeply ($k_p = 41 \to 150$), then gradually decrease ($k_p = 150 \to 400$) to the SKNN performance baseline level of −0.02, and then gradually increase to the baseline of −0.002, as the k_p value increases. In this manner, the performances of SKNN and KNN-S2S are at least comparable in terms of the prediction accuracy when $1.22 \leq f$, whereas KNN-S2S clearly outperforms SKNN when $7.32 < f$, where $f = k_p/k_o$. In the case of $7.32 < f$, it can be seen that the double decision-making process of KNN-S2S is more reliable than the one-time decision of SKNN, as few NNs that are included in the k_o-NN of SKNN according to the rank of $u(0, d)$ with $|q(t) - q(\tau)|^2 \to 0.0$ are excluded from the k_p-NN by the rank of $u(1, d)$.

With regard to a suitable k_p value, if the k_p value is sufficiently larger than the k_o value, reliable prediction accuracy is at least guaranteed by approximating the best case of SKNN within an acceptable error margin, which is insensitive to the prediction accuracy. In this case, the insensitivity is also one of the strong advantages gained during actual use. In addition, the optimal value of k_p can be analysed and updated on a weekly or monthly basis in advance or even fixed at $k_p = f \times k_o$ with $f > 1.0$, through experimental tests.

The three benchmark segmentation methods (i.e., KNN-D, KNN-U, and KNN-UD) were also experimentally evaluated to predict the target data, and all of the past data associated with the previous same day type was used. As for KNN-D, prediction errors decrease exponentially and gradually to MAPE 6.63, when the β value increases (Figure 3(c)). In addition, the prediction error of KNN-D is exactly the same as that of SKNN with $\beta \to 0.5 \times S_n$. This fact indicates that the time dependence exists to a certain extent, even though it is not high. With regard to KNN-U, average prediction error decreases to an optimal error space, goes through the error space area, and then increases steeply according to the increment of the α value (Figure 3(d)). This fact implies that if a high α value is used to reduce searching time, undesirable prediction results (that are not comparable to those of KNN-D) can occur inevitably. In spite of this, KNN-U surely outperforms KNN-D in terms of MAPE, when the span of the α value is between 0.5 and 0.85. In the case of KNN-UD, a distinguished improvement of prediction accuracy does not show, even when the β value increases (Figure 3(e)). This is due to the fact that the temporal variation of the target day deviates from the recurrent pattern of the same day type. This fact directly reveals that a flexible segmentation method combined with a fixed-window method can fail in the improvement of prediction accuracy and can reduce the accuracy performance of KNN-U at least in the case of a nonrecurrent pattern. In addition, KNN-UD is more reliable than KNN-U, when α values are greater than 0.85.

Based on the results of a parameter analysis, the optimal values of the model parameters for SKNN and KNN-S2S were determined to be $[d_o = 5, k_o = 41, k_p^o = 400]$ for result analysis. The optimal values of the model parameters were also identified as $\beta_o = 18$, $\alpha_o = 0.85$, and $[\beta_o = 3, \alpha_o = 0.85]$ for KNN-D, KNN-U, and KNN-UD, respectively, as shown in Figure 1.

4.3. Results and Findings. The test results are summarized in Table 1. The two best performers were KNN-S2S and SKNN, which are followed by the two second-best performers, KNN-UD and KNN-U. This indirectly indicates that the S2S search method, a sort of a dynamic targeting search with an incomplete current time series of traffic flow states, is more reliable in building k-nearest past observations than predetermined static segmentation methods. KNN-S2S is very similar to SKNN in terms of four performance measures, except for data usage. This desirable result directly indicates that the prediction reliability of the high-speed framework proposed in this paper is at least comparable to that of a KNN method, even though the 1.34% of past data was only used at a prediction time point. On the other hand, KNN-S2S slightly outperforms SKNN in terms of prediction-error measures. These results imply that the select pattern built through the pattern recognition process of SKNN, which includes the biased contribution of $|q(t) - q(\tau)|^2 \to 0.0$, could not be the best case for the decision of future states, despite the fact that it would be the best case for the reconstruction of the current temporal state evolution. Therefore, it can be seen at least in our case that the pattern-selection capability of KNN-S2S with the two-step search strategy based on the intrinsic relationship of temporal state evolution is more reliable than that of SKNN from the perspective of the decision-making process.

The conformity of predictions between SKNN using a whole dataset and a KNN model combined with a segmentation method is very crucial. It should be noted that the proposed high-speed KNN framework is not to improve the prediction accuracy but to extremely reduce the execution time of a KNN model, thus ensuring the prediction accuracy of the KNN method combined with the KNN framework. The conformity of SKNN and the two models (KNN-S2S and KNN-UD) through a point-to-point analysis of the

Table 1: Summary of the analysis results.

Model	Performance measures				
	MAPE	MRPE	SDRPE	RMSE	Data usage (%)
SKNN	6.17	0.36	8.13	25.84	100.00
KNN-S2S	6.16	0.36	8.10	25.79	1.34
	0.07*	0.00*	0.19*		0.38**
KNN-D	6.63	0.57	8.80	28.89	12.85
	3.51*	0.29*	4.72*		
KNN-U	6.37	−0.01	8.43	26.24	22.52
	1.68*	−0.40*	2.36*		
KNN-UD	6.37	−0.01	8.43	26.24	22.61
	1.68*	−0.40*	2.36*		

Note. *Measures between elements of SKNN and that of a compared model, and **data usage in the case of one-year data.

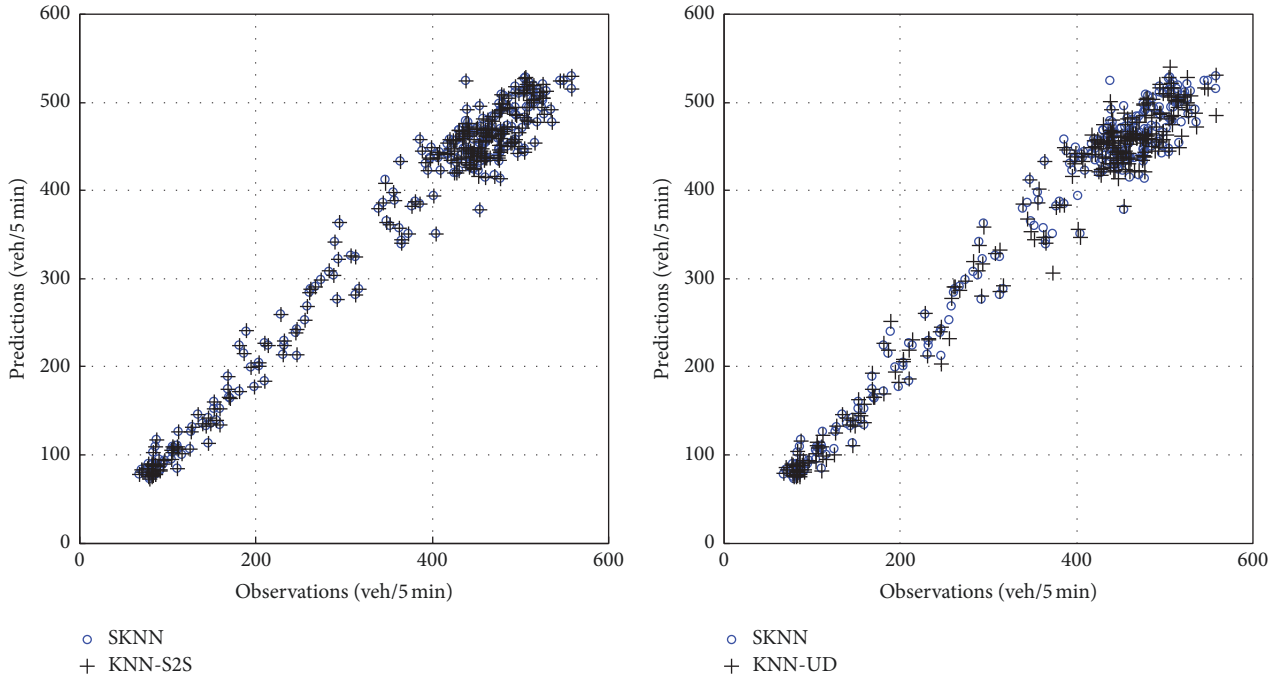

Figure 4: Conformity of predictions between two models.

predictability is shown in Figure 4. The conformity of KNN-S2S is more reliable than that of KNN-UD in terms of MRPE* and SDRPE* (Table 1). The MRPE* and SDRPE* of KNN-S2S are 0.00* and 0.19*, respectively. Otherwise, they are −0.40* and 2.36*. Therefore, it appears that the capability of S2S for dynamic pattern selection is at least comparable to that of the entire data searching and is more reliable than that of a predetermined static segmentation method. This fact provides clear evidence that $x_c(t)$ with $l = 1$ is closely linked to $x_c(t)$ with $l = 0$ in some way from the perspective of the temporal development of traffic volume states.

To inspect the potential of KNN-S2S for selecting promising nearest neighbours, an analysis of the past time points of the selected neighbours was conducted based on time dependency for the target day. The selected past time points (SPTP) by KNN-S2S are compared according to prediction time points (PTP) on the data segment used for the analysis for KNN-UD in Figure 5. Note that SPTP by KNN-UD is included in the data segment. The SPTP (at PTP between 72 and 264) mainly range between 144 and 264. Especially for nonrecurrent cases, SPTP values are highly biased from the space of the data segment. SPTP densely exists between 144 and 264 when PTP ranges from 80 to 144. SPTP also is concentrated in an area between 144 and 228 in the case that PTP ranges from 240 to 264. These results directly indicate that a static data-segmentation method can inevitably fail to ensure the finding of the first- or second-best patterns, even if the data segment might be analysed and updated on a daily or hourly basis with overcoming the time-extensive execution time of the segmentation method.

To examine the potential of KNN-S2S to shorten the execution time, an analysis of data usage (%) according to

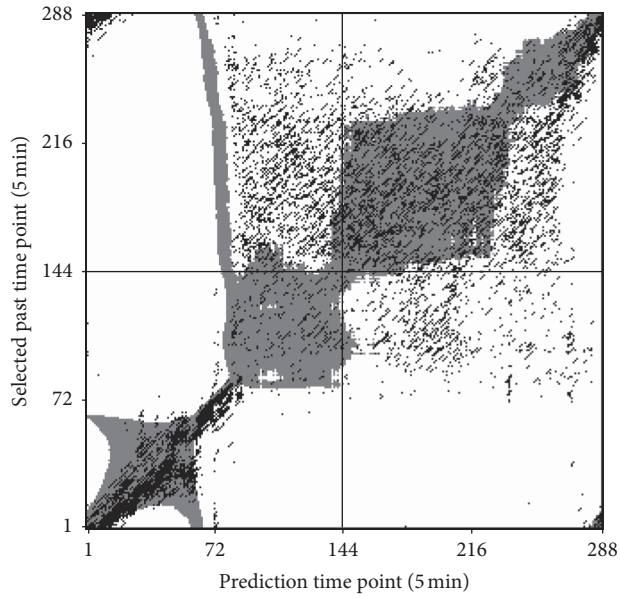

Figure 5: Comparison of a fixed data segment and S2S-based selected time points.

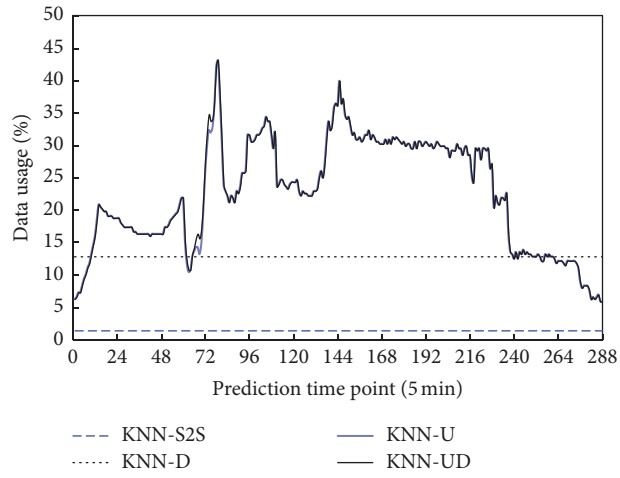

Figure 6: Data usage according to prediction time points.

prediction time points was conducted as shown in Figure 6. This type of analysis is very useful, as the execution speed of the KNN predictor relies considerably on the search time for the amount of historical data. The data usage of KNN-UD varies from 27.78 to 43.06 for the time period (72–240) when estimations generated by a prediction model are heavily used. It should be noted that a volume of searched past data by static segmentation methods increases in proportion to the increment of used past data. For this reason, KNN-UD cannot satisfy the limited running time required in time-critical ITS systems. Contrarily, the data usage of KNN-S2S at t_p is only 1.34% with no computational work related to the access and/or search of historical data at t_p. However, computational loads equal to those of SKNN to search through all of the historical data are inevitable during nonprediction time (i.e., time interval (t)). The computational speed of KNN-S2S was also at least 20 times faster than KNN-UD in the case of daytime. The data usage is also at most 0.38 ($= k_p^o/n \times 100.0 = 400/105{,}120 \times 100.0$) in the case of one-year data, as the useful data is predetermined with the k_p^o value during nonprediction time. This means that the execution time of KNN-S2S at t_p can be 263($= 100.0/0.38$) (i.e., n/k_p^o) times faster than that of SKNN. Therefore, the execution performance of KNN-S2S is instantaneous at t_p. Importantly, the execution time of KNN-S2S does not slow down when the quantity of past data increases, which means that KNN-S2S can meet the required time of ITS data flow consistently. This is one of the strong capabilities of KNN-S2S in real-life applications. Moreover, advanced search methods and data structures can easily be combined into the first step of the KNN-S2S search algorithm to manage the search time more effectively.

5. Conclusions

The slow computing problem of data-driven KNN-NPR remains an ongoing issue as the amount of historical data available grows ever larger. To address this problem, an online framework for a high-speed KNN-S2S algorithm was proposed to reduce the execution time greatly while also improving the prediction accuracy. The algorithm framework was developed based on the deeply linked features of temporal state evolution, sparing artificial approaches to scale down the amount of useful past data and the complexities of advanced search methods.

The analysis results showed that KNN-S2S, despite its extremely high-speed execution time, is at least comparable to the standard KNN method in terms of prediction accuracy. It was also found that the two-step decision-making process of KNN-S2S based on the causal relationship of temporal state development in the pattern recognition process can efficiently diminish the uncertainties of future states from the viewpoint of forecast modelling. This fact can also be indirect evidence that the temporal evolution of traffic volume states is close to the initial deterministic condition, due to the fact that the theoretical foundation of KNN-NPR is based on chaos theory. Specifically, in the case of nonrecurrent conditions in which useful neighbours are not located in a narrow past time window or segment around the prediction time, KNN-S2S is clearly superior to static data-segmentation approaches when used to reduce candidate neighbours.

With regard to the actual feasibility of the KNN-S2S algorithm, it was demonstrated that the algorithm ensures instantaneous and stable execution times with no burden of accessing and retrieving historical data at prediction points, which is in turn another strength over the synchronization and communication of information without any delay in the data flow of time-critical systems. The algorithm was also designed to handle considerable amounts of historical data efficiently without the support of cutting-edge search engines and/or data structures. For this reason, the algorithm can be instantly installed and employed in conventional ITS systems without raising budget issues.

To achieve the objective of high-speed performance, the proposed framework of KNN-S2S was presented with basic

components which are widely used in KNN-based forecast modelling. There are, hence, still promising opportunities to improve on its prediction performance with a combination of different types of state vectors, similarity measures, and forecast functions. Furthermore, the framework can be also used as a preprocess to determine similar learning cases in advanced ITS forecasting models based on support vector machine or deep learning.

Conflicts of Interest

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

Acknowledgments

This work was supported by the University of Incheon (International Cooperative) Research Grant in 2013.

References

[1] B. Yoon and H. Chang, "Potentialities of data-driven nonparametric regression in urban signalized traffic flow forecasting," *Journal of Transportation Engineering*, vol. 140, no. 7, pp. 143–158, 2014.

[2] B. L. Smith, B. M. Williams, and R. Keith Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 4, pp. 303–321, 2002.

[3] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *IET Intelligent Transport Systems*, vol. 6, no. 3, pp. 292–305, 2012.

[4] B. L. Smith and R. K. Oswald, "Meeting real-time traffic flow forecasting requirements with imprecise computations," *Computer-Aided Civil and Infrastructure Engineering*, vol. 18, no. 3, pp. 201–213, 2003.

[5] M. Bernas, B. Placzek, P. Porwik, and T. Pamuła, "Segmentation of vehicle detector data for improved k-nearest neighbours-based traffic flow prediction," *IET Intelligent Transport Systems*, vol. 9, no. 3, pp. 264–274, 2015.

[6] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Statistical methods for detecting nonlinearity and non-stationarity in univariate short-term time-series of traffic volume," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 5, pp. 351–367, 2006.

[7] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: overview of objectives and methods," *Transport Reviews*, vol. 24, no. 5, pp. 533–557, 2004.

[8] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, no. 1, pp. 3–19, 2014.

[9] B. Yu, X. Song, F. Guan, Z. Yang, and B. Yao, "k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition," *Journal of Transportation Engineering*, vol. 142, no. 6, Article ID 04016018, 2016.

[10] H. Chang, D. Park, S. Lee, H. Lee, and S. Baek, "Dynamic multi-interval bus travel time prediction using bus transit data," *Transportmetrica*, vol. 6, no. 1, pp. 19–38, 2010.

[11] S. Clark, "Traffic prediction using multivariate nonparametric regression," *Journal of Transportation Engineering*, vol. 129, no. 2, pp. 161–168, 2003.

[12] H. Chang, D. Park, Y. Lee, and B. Yoon, "Multiple time period imputation technique for multiple missing traffic variables: Nonparametric regression approach," *Canadian Journal of Civil Engineering*, vol. 39, no. 4, pp. 448–459, 2012.

[13] R. E. Turochy, "Enhancing short-term traffic forecasting with traffic condition information," *Journal of Transportation Engineering*, vol. 132, no. 6, pp. 469–474, 2006.

[14] A. Stathopoulos and M. Karlaftis, "Temporal and spatial variations of real-time traffic data in urban areas," *Transportation Research Record*, no. 1768, pp. 135–140, 2001.