Dynamic travel time prediction using data clustering and genetic programming

Mohammed Elhenawy a, Hao Chen b,1, Hesham A. Rakha b,*

a Department of Electrical and Computer Engineering, 3500 Transportation Research Plaza, Blacksburg, VA 24061, United States
b Charles E. Via, Jr. Department of Civil and Environmental Engineering, Virginia Polytechnic Institute and State University, 3500 Transportation Research Plaza, Blacksburg, VA 24061, United States

A R T I C L E   I N F O

Article history:
Received 28 March 2013
Received in revised form 9 February 2014
Accepted 13 February 2014

Keywords:
Travel time prediction
Clustering
Genetic programming
Sampling with replacement
Probe data

A B S T R A C T

The current state-of-practice for predicting travel times assumes that the speeds along the various roadway segments remain constant over the duration of the trip. This approach produces large prediction errors, especially when the segment speeds vary temporally. In this paper, we develop a data clustering and genetic programming approach for modeling and predicting the expected, lower, and upper bounds of dynamic travel times along freeways. The models obtained from the genetic programming approach are algebraic expressions that provide insights into the spatiotemporal interactions. The use of an algebraic equation also means that the approach is computationally efficient and suitable for real-time applications. Our algorithm is tested on a 37-mile freeway section encompassing several bottlenecks. The prediction error is demonstrated to be significantly lower than that produced by the instantaneous algorithm and the historical average averaged over seven weekdays (p-value <0.0001). Specifically, the proposed algorithm achieves more than a 25% and 76% reduction in the prediction error over the instantaneous and historical average, respectively on congested days. When bagging is used in addition to the genetic programming, the results show that the mean width of the travel time interval is less than 5 min for the 60–80 min trip.

1. Introduction

Congestion has proven to be a serious problem in most urban areas in the United States. In 2011, it caused urban Americans to spend 5.5 billion hours more in traveling and cost them an extra 2.9 billion gallons of fuel, for a congestion cost of $121 billion. Congestion has also environmental side effects because of the CO₂ produced during congested periods, which was estimated to be as much as 56 billion pounds in 2011 (Schrank et al., 2012). Adding capacity has been the traditional solution to the congestion problem, but this has become impractical given the financial, environmental, and social constraints. Consequently, highway agencies are seeking new solutions to overcome recurrent and non-recurrent congestion.

Thanks to advanced new technologies that enable continuous monitoring and dissemination of traffic information, it is possible to manage the transportation system more efficiently. The minimum that can be accomplished is to inform the potential users of a road what they can expect during their trip. Such information helps travelers compare alternative routes...
and make better routing and departure time decisions. This is the essence of Advanced Traveler Information Systems (ATISs),
such as the 511 systems that have been implemented nationwide. In many states relevant traffic information is also posted
on Variable Message Signs (VMSs) that are strategically positioned along highways. Because the infrastructure is already
available, we can assist travelers in making better decisions by providing accurate travel time predictions. In case of congest-
tion, many road users may change their routes of travel based on displayed travel time information.

Recently, various traffic-sensing technologies, such as point-to-point travel time measurement systems (e.g., license plate
recognition systems, automatic vehicle identification systems, mobile devices, Bluetooth tracking systems, and probe vehi-
cles, etc.) and station-based traffic-state-measuring devices (e.g., loop detectors, video cameras, remote traffic microwave
sensors, etc.), have been used to collect traffic data. The data collected using these technologies are used in several applica-
tions, including computing travel times. Private companies such as INRIX integrate different sources of measured data to
provide section-based traffic state data (speed, average travel time), which are used in our study to develop algorithms
for predicting travel times. The benefit of using section-based traffic state data is that travel times can be easily calculated.
More importantly, the section-based data provide the flexibility for scalable applications on traffic networks.

Travel time prediction algorithms that use section-based traffic state data can be categorized into two broad categories
depending on the trip experience: dynamic and instantaneous travel time (Mazare et al., 2012; Tu, 2008). Dynamic travel
time reflects the actual, realized travel time that a vehicle experiences during a trip. Dynamic travel time algorithms account
for speed changes over both space and time. Consequently, some algorithms predict future speed patterns and use them to
predict travel times. Instantaneous travel time usually computes travel time using the current speed along the entire road-
way; in other words, the speed distribution is assumed to remain constant for the duration of the trip. As long as the change
of speed with time is not significant, both approaches provide comparable travel time estimates. However, instantaneous
approaches may deviate substantially from the actual, experienced travel time under transient states during which congestion
is forming or dissipating during a trip (Chen et al., 2012).

Some attempts have been conducted using macroscopic traffic modeling to predict short-term traffic states; however,
this approach is computationally expensive and the accuracy degrades rapidly with the increase in the prediction temporal
horizon (Chen et al., 2011, 2012). For long trips, traffic states may change significantly, and the traffic state in the near future
usually cannot provide enough information to cover the entire trip. For example, in the case of a 100-mile trip, if the driver
departs at the time \( t_0 \) and the trip would take 1 h or more depending on traffic conditions, then the traffic state for the follow-
ing 1 h or more would need to be predicted in order to compute dynamic travel times.

An alternative approach to solving this problem is to assume that these states are hidden variables and are function of the
current and previous traffic states. This function can be derived from historical data. The historical dataset provides a pool of
past experienced traffic conditions and shows how traffic status changes over time and space. The key issue is how to devel-
op this function (model) and its parameters and then use it to predict travel times. The purpose of this study is to develop a
simple and fast algorithm to predict dynamic travel times. The proposed method searches the model space guided by the
historical data set to construct a model that best describes the relationship between traffic states along time. A freeway
stretch from Newport News to Virginia Beach is selected to test the proposed algorithm using 5-min aggregated traffic data
for 2010 provided by INRIX. The travel time prediction results from April to August demonstrate that the proposed method
produces higher prediction accuracies compared to the state-of-practice instantaneous algorithm.

The remainder of this paper is organized as follows. A literature review of previous travel time prediction methods is pro-
vided. Subsequently, the proposed genetic programming-based approach is presented. This is followed by a description of
the test data used for the case study and the results of a comparison of the proposed approach to traditional instantaneous
algorithms. The last section provides the conclusions of the research and some recommendations for future research.

2. Literature review

During the past decades, many studies have been conducted to predict travel times. Some of the reviews of different
methods can be found in earlier publications (Du et al., 2012; Lint et al., 2005; Myung et al., 2011; Vlahogianni et al.,
2004). According to the manner of modeling, these methods can be classified into time series models or data-driven meth-
ods. Time series models include the Kalman filter (Yang, 2005; Fei et al., 2011) and Auto-Regressive Integrated Moving Aver-
age (ARIMA) models (Chen and Chien, 2001; Xia et al., 2011; Yang, 2005). Data-driven methods include neural networks (Fei
et al., 2011; Hinsbergen et al., 2011; Lint et al., 2005; Vlahogianni et al., 2004; Xia and Chen, 2009; Xia et al., 2011; Yang,
2005); support vector regression (SVR) (Vanajakshi and Rilett, 2007; Wu et al., 2004), and k-nearest-neighbor (k-NN) (Bus-
tillos and Chiu, 2011; Myung et al., 2011; Qiao et al., 2012) models. These techniques are implemented through direct and
indirect procedures to predict travel times using different types of state variables. Travel time is directly used as the state
variable in model-based or data-driven methods to predict travel times. Indirect procedures are performed using other vari-
ables (such as traffic speed, density, flow and occupancy) as the state variable to predict the traffic status from which future
travel times can be calculated based on some transition function.

Time series models construct the time series relationship of travel time or traffic state, and then current and/or past traffic
data are used in the constructed models to predict travel times in the next time interval (Yang et al., 2010). Kalman filters
were proposed to predict travel times using Global Positioning System (GPS) information and probe vehicle data (Yang,
2005; Nanthawichit et al., 2003). A Kalman filter (KF) is a popular method for data estimation and tracking, in which the time
update and measurement update processes are included. A time series equation is used to predict state variables, and then state values are corrected according to the new measurement data. The main advantage of a Kalman filter (KF) is that the recursive framework ensures traffic data is efficiently updated using only data from previous states and not the entire history (Chen et al., 2012). The state transient parameter in the time series equation is defined from average historical data to calculate future travel times.

A similar idea was used in the Bayesian dynamic linear model for real-time, short-term travel time prediction (Fei et al., 2011). The system noise can be adjusted for unforeseen events (e.g., incidents, accidents, or bad weather) and integrated into the recursive Bayesian filter framework to quantify random variations on travel times. Experimental results based on loop detector data from a segment of I-66 demonstrated that this method produced higher prediction accuracy under both recurrent and non-recurrent traffic conditions. However, a problem existed with these methods in that the travel time in the previous time interval was needed to calculate the future travel time. For real-time applications, the travel time is usually greater than the time interval step size. Hence, the actual travel time from the previous time interval is not available to apply in the algorithms used to predict travel times for the next time interval.

A seasonal ARIMA model was proposed to quantify the seasonal recurrent pattern of traffic conditions (occupancy) (Xia and Chen, 2009; Xia et al., 2011). Moreover, an embedded adaptive Kalman filter was developed in order to update the occupancy estimate in real-time using new traffic volume measurements. Consequently, multi-step, look-ahead occupancy information was estimated to obtain a data matrix representing the spatiotemporal traffic condition for the future trip. Since travel time cannot be directly computed through traffic conditions (occupancy), future traffic speed can be calculated using occupancy data by assuming an average vehicle length and using a constant conversion factor, known as the g-factor in the literature. Consequently, dynamic freeway corridor travel times are predicted with the consideration of traffic state evolution along the corridor. However, this approach may be difficult to implement since the described recurrent pattern of traffic conditions may not be found everywhere.

Data-driven methods usually predict travel times using a large amount of historical traffic data. Time series models are not specified in the data-driven methods, considering the complexity and randomness in the system. Neural networks can be trained using historical data to identify hidden dependencies that can be used for predicting future states. A state space neural network (SSNN) method was proposed to predict freeway travel times for missing data (Lint et al., 2005). The missing data problem was tackled by simple imputation schemes, such as exponential forecasts and spatial interpolation. Travel time was the direct state variable used for the training process, and the experimental results demonstrated that the SSNN methods produced accurate travel time predictions on inductive loop detector data. Supported vector machine (SVM) is a successor to artificial neural networks (ANNs). SVM has greater generalization ability than the ANNs and a superior empirical risk minimization principle (Wu et al., 2004). The application of SVM to time series forecasting is called SVR. The SVR predictor was demonstrated to perform well for travel time prediction. The point-to-point travel time is usually used as the input to ANNs and SVRs. However, both methods require long training processes and are nontransferable to other sites (Myung et al., 2011).

The k-Nearest Neighbor (k-NN) method can be used to find several candidate sequences from historical data by matching current to short, past data sequences. Travel time and occupancy sequences were used to predict dynamic travel times using the k-NN method with data combined from vehicle detectors and automatic toll collection systems in an earlier study (Myung et al., 2011). The occupancy was used since the travel time sequence was collected for the recent past time intervals. The results from the case study demonstrated an improvement in the prediction accuracy by combining two types of sequences in the matching process. Moreover, a k-NN method was proposed by selecting candidates through the Euclidean distance and data trend measures to predict freeway travel times under different weather conditions (Qiao et al., 2012). Unlike ANNs and SVRs, k-NN methods are easy to implement and transfer to different sites without data training.

Genetic programming (GP) has two major advantages. The first advantage is the ability of a GP to find a model to solve a problem without any pre-specified structure of the model. Based on the training data, the GP selects the best model to capture the underlying behavior. The second advantage is that the GP solution is interpretable, which means it defines a logical relationship between the explanatory variables and the response variable. GP has been used successfully for regression and clustering (Oltean and Diosoan, 2009; Bezdek et al., 1994). GP is used in different applications, including curve fitting, data modeling, image and signal processing, financial trading, time series prediction, and economic modeling (Langdon et al., 2008). In the field of transportation, GP is used to build models for real-time crash prediction (Xu et al., 2013). It is used to perform new and complex tasks needed for vehicle guidance and control (Marko and Hampo, 1992). GP is applied to evaluate the performance of the pavement, where it is used to develop models to predict pavement rutting (Jia-Ruey et al., 2008). To the best of our knowledge, we are the first to propose GP for dynamic travel time prediction.

In summary, existing methods are either insufficient or have limitations for predicting dynamic travel times for departures at the current time or future times. The proposed approach used in this study is a data-driven method, yet outperforms the previous methods by fully utilizing the relationship between traffic states and travel times. Moreover, other than previous studies using travel time sequences as input, the proposed method uses spatiotemporal traffic data to build a model using genetic programming and then uses it to predict future travel times.

3. Background

The problem can be stated as follows: given the current traffic states (at time of departure), the goal is to accurately predict the traveler’s travel time. In our case, the speed measurements along the roadway segments are available. Our task is,
given the speeds at the time of departure, to predict the expected travel time. The hypothesis we assume is that the speed along each segment changes gradually, displaying a relationship between the current and future speeds. This relationship can be described using a mathematical function, which we will attempt to derive. Another important point is that we know the relationship between speed, distance, and time so that given the speed and distance measurements we can accurately calculate the travel time. The proposed approach does not entail estimating the speed to compute the travel time; instead, we develop a model that links both the speeds before and at the time of departure to the actual experienced travel time. In this model, the future speeds are hidden and the output of the model is the travel time. This problem can be viewed as developing a model that relates the travel time to particular speed inputs using a functional form similar to Eq. (1).

\[ \hat{y} = \mathcal{F}(x, \beta); \]  
\[ y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \beta_{n+1} x_1 \cdots x_j + \cdots + \beta_p x_l^2; \]  

When we look at our problem this way, we discover that it becomes equivalent to searching for the model within the program space.

The proposed algorithm in this paper is developed using GP (Koza, 1992). GP is an effective paradigm that provides a way to search a solution space for the optimum computer program. In a GP paradigm, the models are represented in a tree structure, and populations of these trees are genetically bred using the principle of survival of the fittest to select and generate new generations from current generations.

There are three main reasons to use the GP to solve this regression problem. First, we cannot satisfy the required assumption for the regression such as the equal variance assumption. Second, conventional regression seeks to estimate the regression coefficient for a pre-specified model. Alternatively, by using GP, there is no pre-specified model structure; instead the GP determines the model structure and regression coefficients simultaneously. The third advantage is the interpretability of the model obtained from the GP. Unlike other machine learning techniques like neural networks, GP generates analytical models that can be interpreted. In the context of travel time prediction, GP models give information about the interaction between the different road segments and which segments are more important for travel time prediction. In the following sections, we will briefly describe the GP approach and how we use it in our problem domain.

When the idea of automatic programming was first considered in the late 1940s and early 1950s, scientists attempted to develop programs without specifying how to carry out the tasks. In GP, the individuals of a generation are a tree-like structure built from genes. There are two types of genes: functional genes and terminal genes. Functional genes are tree nodes with children arguments. These functions depend on the problem domain and have to be defined and described. The other gene types are terminal genes, which are nodes of the tree without branches. In general, GP searches the space of computer programs to find the best fit program by executing three simple steps: (1) generate an initial random population; (2) conduct a fitness test; and (3) execute genetic operations. These three steps are briefly discussed in this section.

3.1. Generate initial random population

Based on the problem domain, we determine our function set \( F = \{f_1, f_2, \ldots, f_g\} \), then we use the set \( F \) and the terminal set \( T = \{t_1, t_2, \ldots, t_l\} \), which includes the input attributes, constants, and random numbers to create individuals of the first generation. If the node is a function, we randomly select one of the functions from the set \( F \) using Eq. (3).

\[
\text{Index} = \text{INT}(g * \eta) + 1;
\]  

where \( g \) is the number of functions in the set \( F \) and \( \eta \) is a random number drawn from a standard uniform distribution \( \text{unif}(0, 1) \). If a tree node is labeled with a function \( f \) from the set \( F \), then \( \phi(f) \) branches are created to radiate out from that tree node, where \( \phi(f) \) is the number of arguments taken by the function \( f \). For each such radiating branch, a random number is drawn from a standard uniform probability distribution and an element from the combined set \( C = F \cup T \) is chosen using Eq. (4).

\[
\text{Index} = \text{INT}((g + s) * \zeta) + 1;
\]  

where \( \zeta \) is another random number. If the branch tree node is chosen from \( T \), this node becomes terminal and does not have branches.

There are several other methods that can be used to randomly create the first generation of programs, as illustrated in Fig. 1. Grow, full, and ramped-half-and-half are the most common techniques to create individuals in the first generation. In the grow method, starting from the root of the tree, the type of node is chosen randomly. If the new node is functional, the function is chosen randomly from a function set \( F \) and children nodes are created for this node. If the node is terminal, a random terminal is chosen from the terminal set \( T \). This process continues until the depth of the tree reaches a maximum depth. In the full method, every node starting from the root node that has a depth less than a certain threshold is considered a functional node. If the node reaches the depth threshold, then the node type is selected randomly. In ramped-half-and-half,
the population is divided into parts. Half of each part is generated using the full mechanism and the other half of each part is generated using the grow mechanism.

### 3.2. Fitness test

Once we have the initial population, we test the fitness of each individual by executing each program in the population and assigning it a fitness value using the fitness measure. The fitness function could be a problem-specific issue. In many practical problems, the fitness of an individual is measured by the mean of absolute difference between the predicted value using that individual and the true value. The closer this mean is to zero, the better the individual.

### 3.3. Genetic operations

Once the fitness of each individual is evaluated, the evolutionary process starts and individuals in the next generation are created using the following techniques: reproduction, crossover, and mutation. Reproduction is simply copying individuals from the old generation to the next generation. Usually 10% of the old generation is copied to the next generation. Crossover is used to provide new and hopefully better individuals. It chooses two existing computer programs and randomly chooses a crossover point within each program to switch the chosen parts of the two programs. For example, consider two individuals $J_1$ and $J_2$ shown in Eq. (5) where $[ ... ]$ shows the crossover points in both individuals

$$
\begin{align*}
J_1 &= [G_{11}, G_{12}, G_{13}, G_{14}, G_{15}] \\
J_2 &= [G_{21}, G_{22}, G_{23}, G_{24}] \\
o_1 &= [G_{11}, G_{22}, G_{23}, G_{15}] \\
o_2 &= [G_{21}, G_{12}, G_{13}, G_{14}, G_{24}] \\
\end{align*}
$$

In genetic programming, there are many mutation operators in use. Sub-tree mutation is the simplest mutation operator, which replaces a randomly selected sub-tree with another randomly created sub-tree. The process of creating generations continues until we reach a stop criterion (e.g., the limit on the generation number). The final result of genetic programming is the best-so-far individual.

### 4. Methodology

#### 4.1. Travel time prediction

Our problem entails predicting travel times given the current spatiotemporal variation in segment speeds. We hypothesize that speeds vary gradually in time and space unless a shockwave propagates through the system. The approach looks back $L$ minutes into the past in predicting future travel times.

##### 4.1.1. Travel time ground truth calculation

The calculation of the travel time ground truth is based on trajectory construction and the known speed through the trajectory’s cells. A simple example of travel time ground truth calculation based on trajectory construction is demonstrated in Fig. 2. In this example the roadway is divided into four sections using segments of length $\Delta x$ and a time interval of $\Delta t$. We assume that the speed is homogenous within each cell. The average speed of the red-dotted cell ($i = 2, n = 3$) in the figure is $u(x_2,t_3)$. Consequently, the trajectory slope represents the speed in each cell. Once the vehicle enters a new cell, the trajectory within this cell can be drawn as the straight blue line in Fig. 2 using the cell speed as the slope. Finally, the ground truth

---

2 For interpretation of color in Figs. 2 and 10, the reader is referred to the web version of this article.
travel time can be calculated when the trip reaches the downstream boundary of the last freeway section. It should be noted that the ground truth travel times were computed using the same INRIX dataset.

4.1.2. Instantaneous travel time

The instantaneous method is very simple where it assumes the segment speed does not change during the entire trip time. The travel time using the instantaneous approach is shown in Eq. (6)

$$\text{Instantaneous travel time} = \sum_{i=1}^{h} \frac{L_i}{v_i^{t0}};$$  \hspace{1cm} (6)

where $L_i$ is the length of segment $i$, $v_i^{t0}$ is the speed at segment $i$ at the departure time $t0$ and $h$ is the total number of segments.

4.1.3. Historical average method

If the spatiotemporal speed matrices are known for several previous days, we can calculate the ground truth travel time at each time interval for each day. The historical average at any time $t0$ is calculated using Eq. (7)

$$\text{Historical average travel time} = \sum_{i=1}^{Z} \frac{\text{GTTT}_{i}^{t0}}{Z};$$  \hspace{1cm} (7)

where GTTT$_{i}^{t0}$ is the ground truth travel time at departure time $t0$ at historical day $i$ and $Z$ is number of days included in the average. In other words the historical average travel time at departure time $t0$ and current day is the average of the ground truth travel times at $t0$ for the previous $Z$ days.

The historical average was calculated considering different $Z$ values ranging from 5 to 30 days, where $Z$ is number of days included in the average shown in Eq. (7). As shown in Fig. 3 there is no significance impact of $z$ on the algorithm performance.

4.1.4. Data reshaping and clustering

In general, data clustering is used to group objects of similar behavior into respective categories. These discovered categories have meaningful structures, within which the data are grouped in a way that the degree of association between two data vectors is maximal if they belong to the same group and minimal otherwise. By visually inspecting the spatiotemporal speed matrix we found that the speed patterns at different time intervals are different because of the activations of different

![Fig. 2. Illustration of travel time ground truth calculation (Chen et al., 2012).](image)

![Fig. 3. MAPE and MAE for the historical average at different number of days included in the average.](image)
bottlenecks. During these time periods the variance of the speed are different and we use clustering to group similar data vectors into clusters. The discovered clusters have similar structure and the GP can find better models for each cluster compared to only using a single model to predict the travel time. In this proposed algorithm to achieve more accurate results, we cluster the dataset into several partitions and build a model for each cluster. Building a model, also known as a program in genetic programming, is done through a training process. The training process is done using historical data that are arranged as shown in Fig. 4. For each day \((D_{58 \times 288} \text{ speed matrix})\) from the historical dataset, we shift a window of size 58 for a step, where 58 is number of road segments and 288 is the number of time intervals. For each sub-matrix \(D_{58 \times L}\) within the (solid black rectangular), we reshape it into a vector as in Eq. (8).

\[
D_{58 \times L} \overset{\text{yield}}{=} \begin{bmatrix} D_1 & D_2 & \ldots & D_L \end{bmatrix};
\]

where \(D_n\) is the transpose of the \(n\)th column. The resulting vectors for each day are stacked together to construct the explanatory variables matrix \(X\). The response vector \(Y\) is the ground truth travel time.

Before using matrix \(X\) and vector \(Y\) as inputs to the GP, we use \(k\)-means to cluster (partition) the \(X\) matrix. Given the rows of \(X = (x_1; x_2; \ldots; x_m)\) as our set of observations, where \(m\) is the number of rows in the matrix \(X\) and each row \(x\) is the reshaped sub matrix \(D_{58 \times L}\), we use the \(k\)-means to partition the \(m\) observations into \(k\) partitions \((k \ll m) S = \{S_1, S_2, \ldots, S_k\}\) so that the objective function in Eq. (9) is minimized:

\[
\arg \min_{\mathbf{c}} \sum_{i=1}^{k} \sum_{x_j \in S_i} d(x_j, C_i);
\]

where \(d\) is the distance measure and \(C_i\) is the mean of the partition \(S_i\). To minimize the objective function, \(k\)-mean sets are randomly selected as an initial set of \(k\) means \((C_1^{(0)}, C_2^{(0)}, \ldots, C_k^{(0)})\), then \(k\)-means alternate between the assignment step shown in Eq. (10) and the update step shown in Eq. (11) until the \(k\)-means converge.

\[
S_i^{(t)} = \{x_q : d(x_q, C_i^{(t)}) \leq d(x_q, C_j^{(t)})\} \quad \forall \ 1 \leq j \leq k \text{ and } 1 \leq q \leq m;
\]

\[
C_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j.
\]

4.1.5. Model building

The output of the \(k\)-means is the partitions set \(S = \{S_1, S_2, \ldots, S_k\}\) and the codebook set \(C = \{C_1, C_2, \ldots, C_k\}\). The next step entails building a model for each partition in \(S\). We use GP to search the model space and find the best possible model. The inputs to the GP algorithm for partition \(S_i\) are \(x_j \in S_i\) for \(\forall j\) and its corresponding ground truth travel time \(y_r\).

In our approach we use a multi-individual GP. Multi-individual GP finds several models for each partition in \(S\). The final model for a partition \(S_i\) is the linear combination of the models found by the GP for partition \(S_i\). To find the linear combination coefficient, we regres the output of these models against the response, as demonstrated in Eq. (12).

\[
y_j = a_0 + a_{11}y_{j1} + \cdots + a_{n}y_{jn};
\]

where \(y_{jr}\) is the predicted travel time from individual \(r\) in partition \(S_i\). The values of the coefficients are found using the ordinary least squares estimation shown in its matrix form in Eq. (13).

\[
a = (Y'Y)^{-1}Y'Y;
\]
where $\hat{Y}$ is the matrix form of the predicted travel time from individuals in $S_i$.

After building all the models for all partitions, we are ready to estimate the travel time for any incoming new data. In estimating the travel time at a specific departure time, we use all the speeds across all segments at the current time and back in time $L$ minutes. This matrix is reshaped into a vector using Eq. (8), then the Euclidian distance between the incoming data and each code vector of the codebook is computed using Eq. (14).

$$\text{arg min}_{C_i} \| x_{\text{test}} - C_i \|^2, \quad \forall 1 \leq i \leq k.$$  

Based on the distance, the new data is assigned a partition, and the model for estimating the travel time of this cluster is used to estimate travel time as shown in Fig. 5. The estimation process is very fast because we only substitute the values of the input variables into the model, which is very simple.

### 4.2. Estimation of travel time lower and upper bounds

In the context of our problem, the training dataset $\pi$ consists of $\{(x_i, y_i), \ i = 1, \ldots, m\}$, where $y_i$ is the ground truth travel time for speed pattern $x_i$. In the above sections, we developed a predictor $\phi(x, \pi)$, such that when new input is received the model predicts $y$ as $\phi(x, \pi)$. Let us assume that we have multiple training datasets $\{\pi_i\}$ each coming from the same underlying distribution. Using multiple training datasets is typically expensive and time consuming. Consequently, we propose the use of bagging to generate multiple datasets. We use sampling from $\pi$ with replacement to generate $B$ new data sets $\{\pi_B\}$ each of the same size $m$. Each observation pair $(x_i, y_i)$ may exist in the bootstrapped dataset several times, once or not at all. Each data set in $\{\pi_B\}$ is a unique sample from the original dataset. Using the $B$ datasets, $K \times B$ models are built using the GP algorithm, where $K$ is the number of partitions. The use of bagging allows for the estimation of a travel time distribution as opposed to estimating a single travel time.

### 5. Case study

The performance of the proposed GP algorithm was tested on a study 37-mile freeway section. A description of the test data is first introduced and then followed by a comparison of the proposed approach to the instantaneous method.

#### 5.1. Data description

The case study is conducted using privately developed INRIX traffic data, which are mainly collected using GPS-equipped probe vehicles. The collected probe data are supplemented with traditional road sensor data, as well as mobile devices and other sources (INRIX, 2012). As a result, the traffic data are the average speed of a roadway segment and aggregated at 5-min intervals.

INRIX data are subject to continuous quality monitoring and improvement process. This process consists of five step process, which are described in detail in the literature (Trepanier, 2013). Moreover, the quality of INRIX data was investigated.
and shown to be good for travel time prediction (Rakha et al., 2013). Finally, it should be noted that the ground truth travel times are computed using trajectory construction using the same INRIX data.

The INRIX data on the main segments along I-64 and I-264 in 2010 are used to construct the travel database. Since heavy traffic volumes are usually observed along I-64 and I-264 heading to Virginia Beach during the summer season and especially during the weekends, efficient and accurate travel time prediction can be helpful to travelers in planning their trips and reducing traffic congestion around the area. A 37-mile freeway stretch is selected to test the prediction algorithm, which includes most of the congested areas heading towards Virginia Beach from Richmond. The selected freeway stretch travels from Newport News to Virginia Beach along I-64 and I-264 and includes 58 sections as shown in Fig. 6. The average length of all the sections is 0.65 miles and the longest section is the 3.7-mile segment located at the Hampton Roads Bridge-Tunnel (HRBT). Typically major congestion forms upstream of the HRBT and thus the freeway section include several congested locations with various backward forming shockwaves upstream of these bottlenecks.

A procedure of data reduction is conducted on the raw data to obtain daily traffic data, which is a spatiotemporal speed matrix. The data samples for typical weekday and weekend traffic occurring in June 2010 are presented in Fig. 7. It should be noted that the traffic flow moves upwards. The figure illustrates a significant amount of missing data, especially for June 5 and 6, 2010 (Saturday and Sunday). It appears from inspection of the data that the weekends involve more missing data than weekdays, which may pose a problem especially when making travel time predictions for weekends. According to the speed map of Fig. 7(a), most missing data (white areas) for a typical weekday occur between 21:00 p.m. and 5:00 a.m. (i.e., during the night and early morning hours). Normally there are few traffic volumes during this time period and free-flow speed could be assumed. However, sometimes the missing data also occur around a congested area (e.g., Fig. 7(c) and (e)). Consequently, free-flow speed cannot be simply assumed for all missing data. A detailed description of the data imputation is provided in (Chen et al., 2012) and briefly described here.

As demonstrated earlier, various traffic data estimation algorithms have been developed for different data sources. Since ramp traffic data are not available, large errors will be introduced if macroscopic traffic models are used to estimate missing data. Alternatively, a statistical approach of data imputation is employed here that utilizes neighboring speed data over temporal and spatial conditions to estimate missing data. Here, the average value of eight neighboring cells is used to estimate the missing speed data in our dataset. Advanced approaches such as using kernel regression over temporal and spatial coordinates can be considered in the future. The samples of estimated speed maps for typical weekday and weekend traffic in June 2010 are presented in the right-hand column of Fig. 7. Consequently, the full coverage daily temporal–spatial traffic data on the selected freeway stretch is estimated and can be used in the proposed travel time prediction algorithm. Note-worthy is the fact that the full matrix of data is used for training and testing purposes, as will be described later.

![Fig. 6. 37-Mile test site.](image-url)
5.2. Travel time prediction results

Because congestion on the freeway stretch usually occurs during the summer months between 5:00 a.m. and 10:00 p.m., the evaluation of the prediction algorithm focuses on travel times from April to August of 2010 during this period (5:00 a.m. to 10:00 p.m.). Since the length of each section and the corresponding average speed for every time interval are known, the instantaneous travel time is calculated for each departure time. Here, we used the fivefold cross validation technique to test the proposed algorithm as shown in Fig. 8 (Efron and Tibshirani, 1997). In the fivefold cross validation, the entire training and testing process is repeated five times (folds). In each fold, four months (clear cells) are used as the training dataset to build

Fig. 7. Samples of daily temporal–spatial traffic state variation.
the models, and the remaining months (yellow cells) are used to test the built models. The overall performance of the proposed algorithm is the average of the five folds’ tests.

The code for the proposed algorithm is written in Matlab using the Genetic Programming toolbox, which is available for free online (searson, 2009). The toolbox returns several models for each training cluster, and the estimated travel time is the weighted sum of individual models. We used the default settings of the toolbox, which are shown in Table 1. It is recommended that further experimentation with the various input parameters be conducted.

5.2.1. Selecting the model parameters

The proposed algorithm uses two parameters, namely: \( L \), which is the temporal look back duration that we use to construct the model, and \( K \), which is the number of clusters used in the algorithm. Optimizing these two parameters is a challenging task. In doing so, we ran the algorithm several times at different values of \( K \) and \( L \). Both relative and absolute prediction errors are calculated for each set of parameters. The relative error is computed as the Mean Absolute Percentage Error (MAPE) using Eq. (15). This error is the average absolute percentage change between the predicted and the true values. The corresponding absolute error is presented by the Mean Absolute Error (MAE) using Eq. (16). This error is the absolute difference between the predicted and the true values.

\[
\text{MAPE} = \frac{100}{IJ} \sum_{j=1}^{J} \sum_{i=1}^{I} \frac{|y_{ij} - \hat{y}_{ij}|}{y_{ij}};
\]

\[
\text{MAE} = \frac{1}{IJ} \sum_{j=1}^{J} \sum_{i=1}^{I} |y_{ij} - \hat{y}_{ij}|.
\]

Here \( J \) is the total number of days in the testing data set in each fold (i.e., 30 days or 31 days); \( I \) is the total number of time intervals in a single day; and \( y \) and \( \hat{y} \) denote the ground truth and the predicted value, respectively, of the dynamic travel time for the \( i \)th time interval on the \( j \)th day. The relative and absolute errors calculated by the proposed method across various values of \( L \) and \( K \) are shown in Table 1 (Tables 2 and 3). The \( L \) parameter was varied from 4 to 10 at increments of 2, while the \( K \) parameter was varied from 2 to 8 at increments of 1.

The results demonstrate that the algorithm is not impacted significantly by changes in the \( L \) and \( K \) values and that the optimum performance occurs when \( K \) ranges from 3 to 7 and \( L \) equals 4.

5.2.2. Testing the significance of the proposed algorithm

In comparing the proposed algorithm (using \( K = 5 \) and \( L = 4 \)) with the instantaneous algorithm and the historical average of seven days, we compute the MAPE and MAE for each day in the data set using the proposed algorithm and the pair methods, as presented in Table 3. Then, we apply the Wilcoxon Signed Rank Test to the MAE measures for the algorithms, as summarized in Table 4. We also applied the same test to the MAPE.

Wilcoxon Signed Rank Test is a non-parametric statistical hypothesis test. This test is equivalent to the paired \( t \)-test. It assumes that magnitudes of the differences between paired observations and the signs of differences carry information about the population. The test takes the paired observations and calculates the differences, where one pair is the error...
measure from the compared algorithm for the same day. The test then ranks the differences from smallest to largest by absolute value. The test statistic $W_{stat}$ is calculated by adding all the ranks associated with positive differences. Finally, we use special tables of the Wilcoxon Signed Rank test to find the $p$-value associated with $W_{stat}$.

In both tests, our null hypothesis is that the distribution of differences is symmetrical around zero. The alternative hypothesis is that the differences tend to be smaller than zero.

The experimental results show that our algorithm has a statistically lower MAPE and MAP compared to the instantaneous algorithm and the historical average method for 7 days. The $p$-value for the Wilcoxon Signed Rank Test is less than 0.0001. Consequently, the null hypothesis is rejected. We conclude that there is significant evidence that the MAE and MAPE for the proposed GP algorithm is less than the MAE and MAPE of both the instantaneous algorithm and the historical average method.

For individual days that have significant congestion and rapid temporal speed changes, the GP algorithm is significantly superior to the instantaneous algorithm. For these days, the MAPE and MAE are 25% less than those for the instantaneous algorithm and 76% less than those for historical average. As shown in Fig. 9, the instantaneous algorithm underestimates the travel time as congestion builds up. Another drawback of the instantaneous algorithm is that it overestimates the travel time as the peak period recedes. Alternatively, the GP algorithm responds quickly to changes in speeds at the shoulders of the peak period.

### 5.2.3. Model Interpretability

One of the advantages of using GP in predicting travel times is that the model is simple and interpretable. By analyzing the model variables, critical segments that affect the predicted travel time along a roadway can be identified. To illustrate this concept, Fig. 10 visualizes the codebook at $K = 4$ and $L = 4$. This codebook illustrates the four clusters that the model identified. In the figure, the red color represents low speeds. The first two clusters show a red color at the tunnel location where congestion occurs. The third code vector shows light congestion at two different locations, one of them at the tunnel. The fourth code vector shows free-flow conditions. These clusters clearly demonstrate that for congested conditions the tunnel is the most critical location in predicting travel times given that it serves as a recurring bottleneck.

In order to interpret the GP models for the dataset partitions obtained using the above codebook, we visualize the coefficients of the first order terms in the models. As shown in Fig. 11, the most important coefficients are those at the tunnel or...
near the tunnel area. Also, most of the important coefficients are those related to the speeds immediately prior to departure (i.e., at \( t_0 \)). The algebraic equations (models) built using the GP are presented in Appendix A.

### 5.3. Bagging and genetic programming results

The other important information are the upper and lower limits on the travel time. The travel time interval estimation is based on bootstrapping the original dataset. Given the dataset \((X, Y)\) where each row in the \( X \) matrix is the explanatory variables and the corresponding element in \( Y \) is the ground truth travel time, we build 100 datasets \( S = \{(\tilde{X}_1, \tilde{Y}_1), (\tilde{X}_2, \tilde{Y}_2), \ldots, (\tilde{X}_{100}, \tilde{Y}_{100})\} \), where \((\tilde{X}_n, \tilde{Y}_n)\) is the data set number \( n \). The dimensions of \((\tilde{X}_n, \tilde{Y}_n)\) are the same as \((X, Y)\). The \((\tilde{X}_n, \tilde{Y}_n)\) should be about 66% of the original training \((X, Y)\) and the other cases are repeated cases. For each dataset \((\tilde{X}_n, \tilde{Y}_n)\), we apply our proposed algorithm using the same configuration shown in Table 1 except the population size is set to 100 to derive the models for this data set.

The travel time interval for an unseen new data case (instance) can be defined by the maximum and minimum travel times estimated from the models from all 100 datasets. The histograms of the predicted travel time width for the five-month analysis period are shown in Fig. 12 and Table 5 demonstrate that the width of the travel time interval is less than 5 min in

---

**Fig. 9.** Comparison of proposed algorithm and the instantaneous algorithm for two days.
Fig. 10. Visualization of codebook at $K = 4$ and $L = 4$.

Fig. 11. Coefficients of linear term models.
Furthermore, as illustrated in Fig. 13, most of the ground truth travel time experiences are within the estimated travel time interval. Furthermore, points that are outside the interval are close to the interval borderlines. This accurate information, when provided to travelers, gives them a better idea about the expected conditions on the road. We also note that using the travel time interval is more robust than using a single value for the predicted travel time. When the speed pattern along the road varies, the difference between the predicted travel time and the ground truth is larger than the difference between the upper bound of the travel interval and ground truth. This makes travelers more confident when using the travel time interval.

6. Conclusions and future work

The research presented in this paper develops a genetic programming algorithm to predict dynamic travel times. The proposed algorithm uses the \( k \)-means approach to partition the data into similar clusters in the training phase and to act as a simple classifier during the testing phase. The genetic programming approach is then used to build a model for each data partition. The proposed algorithm has two main advantages. The first advantage is the simplicity of the model and its computational efficiency. The second advantage of the model is that it is interpretable and provides insight into critical segments.

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Monthly variation in predicted travel time interval.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April</td>
</tr>
<tr>
<td>Mean</td>
<td>4.19</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>2.368</td>
</tr>
<tr>
<td>Std. err mean</td>
<td>0.03027</td>
</tr>
</tbody>
</table>

Fig. 13. Temporal variation in travel time estimates relative to ground truth.
that significantly impact future travel times. The results show the superior performance of the proposed algorithm when compared to the state-of-the-practice instantaneous algorithm. The proposed algorithm is tested on a 37-mile freeway section. The prediction error is demonstrated to be significantly lower than that produced by the instantaneous algorithm or the historical average ($p$-value < 0.0001). Specifically, the proposed algorithm achieves more than a 25% and 76% reduction in the prediction error over the instantaneous and historical average, respectively on congested days. When bagging is combined with GP, the results show that the mean width of the travel time interval is less than 5 min for the 37-mile trip.

Further research is proposed to identify the minimum data requirements to train the model to the specific roadway and traffic conditions. Further testing is also required to test the transferability of the model. As such, the researchers recommend that further testing of the model be conducted on different roadway segments.

Appendix A

Table A1 below shows the algebraic program for each cluster. As shown in the table, the equations have linear terms and interaction terms. From the equation, we can see that most of the dominant terms are at time $t_0$ or near $t_0$. The $v(s,t)$ in the equation represents the speed at segment number $s$ and time $t$.

Table A1

| Model | Algebraic equations (models). | \[
\text{Model } \#1 \quad y_{\text{pred}} = 78.74 + 0.002495 \times v(t_0,46^2) - 0.2911 \times v(t_0,46) - 0.05495 \times v(t_0-1.42) + 0.002495 \times v(t_0(1.39) - 0.07538 \times v(t_0,48) - 0.05495 \times v(t_0,45) - 0.05495 \times v(t_0,44) - 0.1185 \times v(t_0,36) - 0.07583 \times v(t_0,15) - 0.07334 \times v(t_0-3.22) - 0.05246 \times v(t_0-2.36) + 0.002495 \times v(t_0,28)\\
\text{Model } \#2 \quad y_{\text{pred}} = 92.82 + 0.05417 \times v(t_0-3.49) + 0.00487 \times v(t_0-2.2) - 0.05417 \times v(t_0-1.45) + 0.00487 \times v(t_0-3.45) + 0.05417 \times v(t_0-1.34) - 0.1419 + 0.08775 \times v(t_0,48) - 0.08775 \times v(t_0,45) + 0.05417 \times v(t_0-3.40) - 0.08775 \times v(t_0,29) - 0.05417 \times v(t_0,23) + 0.00487 \times v(t_0,15) - 0.1419 \times v(t_0,4) - 0.1755 \times v(t_0-3.3) + 0.00487 \times v(t_0-2.57) - 0.002273 \times v(t_0,46)\\
\text{Model } \#3 \quad y_{\text{pred}} = 75.92 + 0.01868 \times v(t_0-3.16) - 0.05009 \times v(t_0-3.44) - 0.00311 \times v(t_0,54) - 0.01868 \times v(t_0,53) - 0.09342 \times v(t_0,47) - 0.1124 \times v(t_0,46) + 0.05009 \times v(t_0,41) - 0.1124 \times v(t_0,31) - 0.05009 \times v(t_0,29) - 0.05009 \times v(t_0,26) - 0.07592 \times v(t_0,14) + 0.05009 \times v(t_0-3.28) - 0.06907 \times v(t_0-3.7) - 0.05009 \times v(t_0-2.47) - 0.001875 \times v(t_0,44) + 0.00311 \times v(t_0-3.25) + v(t_0-2.40)\\
\text{Model } \#4 \quad y_{\text{pred}} = 67.03 + 0.1929 \times 10^{-3} \times v(t_0,38^2) + v(t_0,37) - 0.03861 \times v(t_0,57) - 0.03861 \times v(t_0,43) - 0.07254 \times v(t_0,42) - 0.1111 \times v(t_0,38) - 0.1111 \times v(t_0,37) - 0.03861 \times v(t_0,21) + 1.929 \times 10^{-3} \times v(t_0-3.2) - 0.03393 \times v(t_0-3.27) - 1.292 \times 10^{-3} \times v(t_0,43) - 0.07254 \times v(t_0,58) - 0.07254 \times (v(t_0-1.10) + v(t_0,43) + v(t_0,38) - v(t_0,80)) - (v(t_0,43) + v(t_0-1.19) + v(t_0,54) + v(t_0,19) + 7.378)\]

References


Trepanier, T., 2013. INRIX Data Services – Arterial System Performance Assessment and Management.