

國立交通大學

資訊科學與工程研究所

碩士論文

用於交通預測之二層資料分群法

Clustering Traffic Sensing Data for Traffic Prediction: A
Two-Phase Clustering Approach

研究生：榮芊菡

指導教授：彭文志 教授

中華民國 一 百 零 一 年 一 月

用於交通預測之二層資料分群法
Clustering Traffic Sensing Data for Traffic Prediction: A Two-Phase
Clustering Approach

研究生：榮芊菡

Student : Han-Chien Jung

指導教授：彭文志

Advisor : Wen-Chih Peng

國立交通大學
資訊科學與工程研究所
碩士論文



Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

January 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年一月

用於交通預測之二層資料分群法

學生：榮芊菡

指導教授：彭文志

國立交通大學資訊科學與工程研究所

摘 要

現代城市的運輸系統幾乎已資訊化，而具備交通路網資訊的資料庫經常累積了相當豐富的歷史交通資料。在目前的系統中，即時的交通資料常被用於估算當時路況以及預測短期未來的交通路況；然而，隨時間累積的歷史資料實際上隱含了其路段上路況變化的習性與特徵。若能發掘出這些資訊，我們便能用於推測未來長期的交通路況，進而增進許多交通資訊服務系統的效率與準確性。我們在此研究中提出一個二層資料分群方法，從一路段的歷史交通資料探勘出其道路速度的變化習性。我們的方法是，首先估算該道路速度變化的基礎樣式，稱之為巨觀估算，接著再針對交通尖峰時刻的速度樣式做細部估算，我們稱為微觀估算。其中，此方法的輸入資料格式為時間序列資料，每條序列是連續具時間值的道路速度紀錄。為了將眾多條時間序列資料分群，我們採用了專門用於量測時間序列資料間相似度的方法。因此，在本研究實驗部分，我們分析了使用不同的時間序列量測法於本二層分群法中效能之差異，以及使用不同分群演算法所帶來效能的差異。另外，我們還提出了三種基於二層分群結果之交通預測函式，並分析了這三種函式的預測效果。最後，在實驗中，我們的二層分群方法與統計領域中回歸式的預測方法做比較，數據顯示我們的方法確實能帶來更好的準確度。

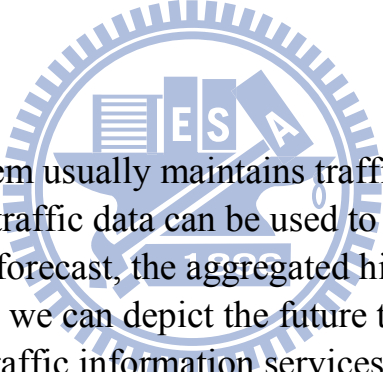
Clustering Traffic Sensing Data for Traffic Prediction: A Two-Phase Clustering Approach

student : Chien-Han Jung

Advisors : Dr. Wen-Chih Peng

Institute of Computer Science
National Chiao Tung University

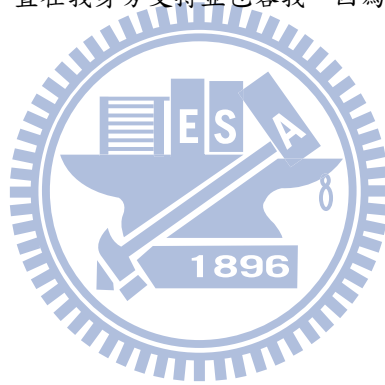
ABSTRACT



A modernized transport system usually maintains traffic databases with sufficient historical data. While real-time traffic data can be used to estimate the present traffic states and the short-term traffic forecast, the aggregated historical data actually imply some traffic behaviors by which we can depict the future traffic patterns over a long period and support the on-line traffic information services. In this work, we propose a two-phase mining method to explore the speed patterns given the historical driving data of one road segment. Generally, we estimate the speed patterns on a *macroscopic scale* in the first phase, and then in the second phase we explore more *peak-time patterns* on a *microscopic scale* from their macroscopic appearances. Additionally, the input of our method consists of sequences of time series data recorded over numerous days, and clustering on the sequences is performed based on the similarity measuring of the time series data. Hence, in this work, we analyze the availability of several frequently-used time series similarity measuring methods combined with the clustering methods, and furthermore develop a traffic prediction model with three kinds of predicting functions to examine our two-phase mining method. Finally, in the experiment section, we analyze the performance of our two-phase mining method adopting different selections of similarity measuring method with clustering method, as well as the accuracy of the proposed three prediction functions.

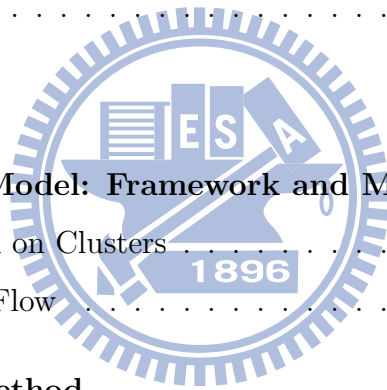
誌 謝

回想這段研究的過程，其實充滿了很好的回憶--受到許多幫助及關懷的回憶。歷經大三開始做專題到研究所這兩年，感謝彭文志老師的指導與關照，讓我參與專題製作、國際研討會MDM的舉辦以及demo paper展示、與國科會合作的計畫、教育部舉辦的軟體比賽，以及碩士論文研究，真的得到了很多寶貴的經驗與知識。在我們的前瞻資料庫系統實驗室，老師對我們的要求，如做學問態度需積極並謹慎，對己身任務需負責，與實驗室學長姐弟妹多加交流與合作等，都是將來出社會仍然非常受用的訓練；因此覺得自己相當幸運，能成為這個實驗室的一員，尤其是，實驗室的同伴們真的都非常友善，跟大家相處真的很開心。感謝有01學姊從大學專題以來的帶領，我才能順利完成許多任務；還有感謝Barry學長，已畢業的Oshin學長，Dimension學長，Young學姊這些高強的博班學長姊適時給予我指點建議；以及已畢業的碩士學長姐，至雯學姊，zvn學長，vcore學長總是很熱心給我討論研究問題及分享經驗；還有一起奮鬥度過歡笑淚水的acrt，kp張，Luc，廷威，我也總是受到你們的幫助，多虧有你們這群熱血的好夥伴，我的碩士兩年生活變得很熱鬧有趣；還有貼心的學弟妹們，拍拍，kerker，雅婷，wallman，堃偉，感謝你們總是在我慌亂時幫我一把，還有在最後的幾個月的鼓勵與關心對我而言都是很重要的動力。最後感謝我的家人們，一直在我身旁支持並包容我，因為你們的陪伴，就是我的力量泉源。



Contents

1	Introduction	1
2	Related Work	6
2.1	Traffic Estimation and Prediction	6
2.2	Time Series Forecasting	7
2.3	Time Series Distance Function	8
2.4	Fastest Path Searching	8
3	Preliminary	10
4	Overview of Prediction Model: Framework and Method	13
4.1	Speed Prediction Based on Clusters	13
4.2	Two-Phase Clustering Flow	14
5	Two-phase Clustering Method	17
5.1	The Distance Measuring of Speed-Time Series Data	17
5.2	Clustering By Time Series Data Similarities	19
5.3	Speed Pattern Peak Finding	22
5.3.1	Sliding-average smoothing	23
5.3.2	First derivative sequence	24
5.3.3	Two-way sliding window	24
6	Speed Pattern Prediction Function	27
7	Experiments	30
7.1	Datasets and Setting	31

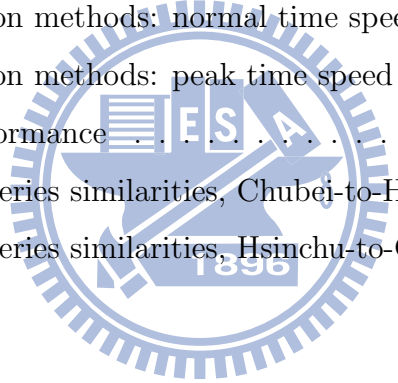


7.1.1	Data Preparation	31
7.1.2	Evaluation Measure	32
7.2	Clustering Method Performance	34
7.3	Evaluation of Prediction Models	36
7.4	Evaluation of Time Series Distance Functions	37
7.4.1	The Accuracy in Normal Time Prediction	37
7.4.2	Accuracy in Peak Time Prediction	37
8	Conclusion	44



List of Figures

1.1	Real data example on the Chubei-to-Hsinchu highway segment	4
4.1	Speed prediction framework	14
4.2	Two phase clustering flow	16
5.1	Chubei-to-Hsinchu highway traffic records	21
5.2	Peak finding sample	25
7.1	Clustering and regression methods: normal time speed error	35
7.2	Clustering and regression methods: peak time speed error	39
7.3	Conditions of best performance	40
7.4	Best case of four time series similarities, Chubei-to-Hsinchu data set	41
7.5	Best case of four time series similarities, Hsinchu-to-Chubei data set	43



Chapter 1

Introduction

With more traffic data sources available and so more sufficient traffic data that can be analyzed, mining this data to get meaningful information is of great help to our transport systems. The most important information is the driving speed on the road. This speed directly reflects the traffic conditions on the road, and it is also the key to routing services such as fastest path finding. The driving speed data can be either real-time or historical, and can be obtained from different types of traffic detecting devices. For example, static sensors such as inductive loop sensors and camera sensors are deployed on the roads, and usually we can get speed data directly from the inductive loop sensors. In recent years, GPS sensors have become commonly embedded in portable devices like smart phones, PDAs and mobile navigators (TomTom, Garmin, etc.). Drivers use the applications on these portable devices while traveling on the road, while at the same time the applications also record and send their GPS trajectories, with speed information, to some traffic databases. In recent years, the estimation of real-time traffic [15, 3, 25, 20], and short-term traffic forecasting [16, 18, 21, 27, 5, 7], or on-line fastest path searching with real-time traffic conditions [22, 19, 26], have been the focus of numerous research works. However, we believe that mining traffic patterns from off-line historical data is also both meaningful and necessary. Our motivation comes from both the applications in the real world and research problems.

Some real world applications, like Google map in some countries, have provided the function of traffic state prediction according to different weekdays and times. For this function, the

historical traffic data can not only support instant traffic data but can also provide information about the common traffic behavior according to different weekdays. This function would be even more valuable if it was able to predict the traffic states in the u upcoming week, based on the recent historical traffic data. Sometimes drivers not only want to know the current traffic state, but also the predicted traffic state two or more days in advance. Hence, for example, drivers can get their traveling route several hours and even several days before starting off, so as to plan their journey well. On the other hand, the commuter can overview the traffic state each weekday and plan more routes to his/her workplace according to the different weekday traffic situations. In the above examples, we mention the scenario of route planning, which is also a common function of digital maps and portable devices. Route planning is often viewed as the *fastest path searching* problem in research works [10, 14, 17, 22, 19, 26]. Besides the fact that off-line path searching problems need historical traffic data [10, 14, 17], even on-line fastest path searching cannot just depend on the real-time traffic information [22, 19, 26]. One reason is that the system may sometimes lack real-time data; another is that the pre-computation of historical data can reduce on-line traffic forecasting costs.

The above mentioned scenarios imply that the traffic prediction based on historical data is more likely to find a *pattern* of traffic states during a long-term period, such as one day or one week, than to predict one traffic state value in the short-term. In this work, we consider the pattern of traffic states during one period as a time series of speed values, and define the predicted time series of speed values in one specific period as a *speed pattern*. If every road has its historical data, then our goal is, that given a target road and a queried future time period, we can predict a *speed pattern* during the period for this road. If we can predict the speed pattern during one future period, we can also provide one predicted speed value at a specific time point in that period. Since the prediction is based on off-line historical data mining, we can infer a long-term future speed pattern without real-time traffic data, which provides solutions to the above mentioned scenarios and existing research problems. First, we can look ahead to future traffic states such as two or more days ahead or all of the days in the upcoming week. Second, speed patterns created off-line can reduce the on-line computation cost of fastest path searching. Still there exists one more benefit of mining speed patterns. The speed pattern in fact stands for the most likely traffic behavior during a given time period based on past experience. Consequently, once we have real-time traffic data on-line that can

be compared with the off-line computed speed pattern, we can easily detect the *abnormal traffic behavior* if the real-time data differs greatly from the pattern. With the speed pattern, we can reduce the on-line computation cost when detecting *abnormal traffic behavior*.

The speed pattern basically represents the common traffic behavior during a time period. Take Figure 1.1 for example; we can roughly identify some periodic behavior appearing in the historical speed record over several days. Assuming that there are no traffic accidents that will cause abnormal traffic behavior in the future, we can expect that the periodic behaviors will be repeated in the future. This is our concept of mining the speed patterns, that is, to find the most frequently appearing periodic pattern from the historical data. However, with detailed observation of historical data as in Figure 1.1, we can find that although the valley wave happens in a certain periodic way, the shapes of the valley wave are not always the same, and the lengths of the intervals between the waves are not equal. This observation leads to two objectives. First, there may exist more than one kind of periodic behavior, so we should find an adequate number of patterns that frequently happened and are believed to appear in the future. Second, after we find one or more than one kind of pattern, such patterns can be separated into sub-patterns according to some parts of their detailed wave shape. To achieve these two objectives, our general concept is to develop a clustering method on a set of time series data of historical speed records. In this work, we define that a sequence of time series data is the accumulated speed-time records during one day, that is, during a period of twenty-four hours. The reason why we segment the time series data this way is intrinsic. Since traffic flow is generated by human activities on the road, and people schedule their lives by the unit of days, i.e. 24 hours, cutting traffic patterns into days is natural and so an intuitive truth. Figure 1.1 is a visualization example, where we delimit the continuous two-week time series by day length (24 hours). To cluster the similar sequences of time series data, we need to measure the distance between each of them. Hence, before the clustering step, we utilize the *time series distance function* to measure the distance between time series data. Generally, we can get speed patterns from the output clusters. However, we have some issues to deal with while developing an effective method to achieve the above mentioned two objectives. Since we utilize the time series distance function combined with the clustering method, we should choose an adequate selection for the two steps. There are some existing works of time series distance functions, such as Euclidean distance, DTW[4], and many other proposed

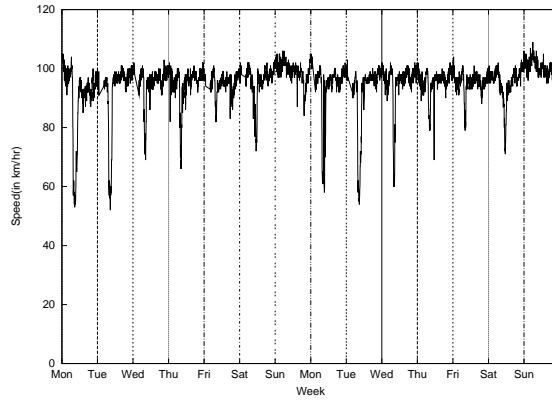


Figure 1.1: Real data example on the Chubei-to-Hsinchu highway segment, Taiwan, recorded in 2011, May 16-29

methods, not to mention numerous classic clustering methods in data mining theories. Hence, how to find the most effective selection of time series distance function with the clustering method is an important issue. Second, to achieve our second objective, i.e. exploring the sub-patterns of the valley waves among one general all-day speed pattern, we should develop an extended phase following the first clustering phase. It is another issue that in the second phase, we should find out the time series distance function that works well under the first phase clustering conditions. There is still one more issue. With limited background knowledge of realistic factors that influence the periodic traffic behaviors, how to predict the future traffic based on our clustering results is the issue. For example, we find that pattern A and pattern B are the most representative patterns of road R . If we want to predict the traffic state at time t on a future day d , which pattern is the most likely to happen on the queried date d ? Besides, if pattern A is the predicted pattern on d , but the queried time t is in the *peak time*, i.e. the valley wave in a traffic pattern, on d while we have explored the peak-time pattern $A - p_1$ and $A - p_2$ of pattern A , which peak-time pattern should we choose? With consideration of these three issues, we have developed a two-phase clustering method along with three traffic prediction functions to accomplish our two above mentioned objectives.

In this paper, we first propose a two-phase clustering method with time series speed data as input for each target road segment. In the first clustering phase, the data series will be clustered according to *macroscopic scale* time series similarities. In the second phase, for each clustered time series group, the pieces of *peak* series data will be cut out and clustered for the second time. This step is also performed as *microscopic scale* clustering. For

traffic prediction problems, the accuracy in peak-time traffic is the most important task; hence the *microscopic scale* clustering in the second phase is our revision of the peak-time traffic estimation. We then propose three traffic prediction functions that choose the output patterns from the clustering results given a set of time-period features. By the above proposed methods, for a target road, given the query time with its time-period features, our prediction model will output a speed pattern during the queried time period or the speed value at the queried time point. In conclusion, the main contribution of our work can be summarized as the follows:

1. We propose a traffic prediction model generalized for a traffic service system. This model provides driving speed predictions at any time at a given date. By considering the defined features of that date, our model can output the speed pattern for the whole day at once.
2. As our first-phase clustering method estimates the all-day speed pattern, it can detect the peak-time interval and revise the peak-time pattern by an extended calculation. Namely, our prediction model can tell the peak-time behavior and interval given a target road and date, along with more precise peak-time traffic conditions.
3. We evaluate the performances of the most frequently used time series distance functions combined with classic clustering methods to find the most effective selection for exploring speed patterns with intensive experiments.

In the following sections, we first discuss some related topics and works in section 2, then present the problem definition and data description in section 3. Section 4 introduces our two-phase traffic prediction model using the framework description, and explains the clustering method flow. Section 5 is the description of our clustering method along with the specially formulated peak time finding method. Section 6 defines the three different prediction functions. The last section is the evaluation of our method with a real data set.

Chapter 2

Related Work

2.1 Traffic Estimation and Prediction

The widespread use of GPS sensor embedded portable devices (smart phones, GPS loggers, etc.) has provided more sources of traffic data besides static sensor data like loop detectors. Due to this fact, new techniques and issues related to the problem of measuring or forecasting traffic conditions have also arisen. For measuring traffic, numerous existing works have proposed approaches to deal with either static sensor data [5] or GPS data [25, 20], such as to estimate traffic conditions accurately when the data are not wide-spread in the road network or not dense for some roads [3, 15, 24]. In this work, we assume that we have already obtained sufficient data for every queried road and we focus on the prediction problem. Traffic data from GPS sensors are usually sampled by individual moving objects, and hence form trajectories: sequential points contain time and speed or location information. Some works predict traffic in the road network by inferring the next movement of individual trajectories [16, 18], such as Gaussian process regression [16] or statistical approaches [18]. The input of this methods is limited to drivers' trajectories, but we hope to deal with general traffic data sources. Besides, they focus on inferring the next several steps of traffic conditions given previous steps of conditions (drivers' moving) before the current one, but we aim to mine the traffic patterns of roads and directly infer several days in advance. The goal is thus different from these works. Other works are not limited to trajectory sources, but the proposed meth-

ods such as autoregressive regression (AR) [21, 27], linear regression [5] and support vector regression (SVR) [7] are also more suitable for predicting next-step traffic conditions with on-line incoming data rather than the off-line pre-computations that predict future days' traffic behavior like our work in this paper.

2.2 Time Series Forecasting

We can see that regression models are frequently used in traffic prediction. In fact, these all belong to state space models and have been popular in time series modeling and forecasting, especially in econometrics such as stock prediction [2, 12]. Among the existing methods, the most popular is ARIMA(Autoregressive Integrated Moving Average)[6]. ARIMA is a representative statistical approach in econometrics, and a generalization of an autoregressive moving average (ARMA) model. The latter consists of an autoregressive (AR) part and a moving average (MA) part. Again, ARIMA model calculates the t step given the p previous steps' (in fact, the AR and MA parts need *two* previous referencing step parameters) real data. Back to our goal in traffic prediction, we want to predict a whole day (24 hours), new day traffic condition, or look ahead to three days ahead, just like the scenarios illustrated in the introduction such as planning a trip several days in advance, or routing system pre-computation. The steps we want to predict may involve hundreds of steps, and usually we lack the previous hundreds of steps in our scenarios. The regression models such as ARIMA, AR and SVR mentioned above are inefficient and difficult to apply to our situations. As a matter of fact, our predictions are not made by extending the possible trend from the historical time series. We have explained in the introduction that the traffic naturally forms a daily pattern. Hence, instead, we view the daily time series as objects generated from a set of classes, and each of the classes defines a kind of daily traffic pattern profile. Furthermore, the classes even have extended sub-classes which define the "peak-time patterns" embedded in the parent classes. Hence, our approach differs significantly from the existing time-series prediction works.

2.3 Time Series Distance Function

There are numerous existing time series similarity measuring methods, such as Euclidean distance, DTW [4], LCSS [23], EDR [9], ERP [8] and many other proposed novel methods. According to the experimental survey in *Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures*[11], the conclusion is that elastic measuring methods such as DTW, LCSS, EDR and ERP are significantly more accurate than lock-step methods like $L_p - norms$ [13] and DISSIM when the data set is not huge. Moreover, the elastic measuring methods also generally outperforms some novel methods (TQuEST, SpADe) in accuracy. For comparison between the elastic measuring methods, the experimental results also show that the edit distance based methods like LCSS, EDR and ERP in fact have very close accuracy compared to DTW. Based on this conclusion in [11], we implement DTW, EDR and LCSS for our time series data similarity measuring.

2.4 Fastest Path Searching

Thinking that the traffic states on roads may change all the time instead of being static, recent works on fastest path searching have often defined the road network as a *dynamic road network*. Some of the works view the path searching problem as a time-dependent shortest path problem [10], that is, the routing algorithm should use the traffic condition at the time when the road was actually driven. Hence, each edge in the graph has a function for telling the traveling time length given a time point. This traveling-time function can be transferred from a pattern of speed variation according to time axis, hence some of the works say their methods use *speed patterns*[14, 17]. However, although the above works have detailed methods for routing algorithms, they do not include much discussion of how the *speed pattern* or *traveling time functions* are created. On the other hand, some of the works have proposed a framework for a path searching system with real time traffic information from the road network continually updating to the system [22, 19, 26]. In the previous works like [22, 19], the fastest path re-evaluation is triggered by every traffic state alteration regarded as impacting the present computed route. Without prediction of the upcoming traffic state alternation, the above mentioned system is encumbered by the heavy cost of online computation, because it

has to deal with all updated traffic data. If the traffic pattern can be estimated off-line, the path searching system will not always be tied up by real-time traffic data, but will only be required when the updating traffic data are contrary to prediction.



Chapter 3

Preliminary

Our problem is that, given a road segment r , the historical speed-time data set of r , and a query time t_q , the proposed model should output a predicted speed value V . In this work, we view the predictable road segments as those possessing sensors to record the driving speed at each specific time point of a sampling frequency, no matter what kind of sensor it really used. Due to this fact, the raw data point can be presented as a speed-time data tuple, $\langle \textit{speed value}, \textit{time value} \rangle$. Note that our problem is to predict the speed of one given road segment by its historical traffic data and not to consider data sources from other road segments, so we do not add the road segment attributes, such as road ID, to the data tuple format. Since the speed data are sampled continuously along the time axis, the data source becomes a kind of ***time series data*** which is also a typical data type used in the data mining field. We define the $\langle \textit{speed value}, \textit{time value} \rangle$ data sequence as ***speed-time series data*** to emphasize that the ***time series data*** used in this work features only speed values.

Definition 1. (*Speed-time Series data*): The ***speed-time series data*** S is a sequence of speed values, with each value s_i sampled at a specific time point t_i , i.e.; $S = [(s_1, t_1), \dots, (s_n, t_n)]$, where n is the length of S .

According to the *speed-time series data* definition, a sequence S with its time value spread over 24 hours of one specific date, is a ***daily speed-time series***. The ***daily speed-time series*** files of the past dates are the **standard input format** of our prediction model learning

process.

Definition 2. (*Daily Speed-time Series*): The **daily speed-time series** S_D is a **speed-time series data** $[(s_1, t_1), \dots, (s_n, t_n)]$, where $\forall t_i \in [t_1, \dots, t_n]$, t_i is located on the time points during the 24 hours of the specified date D . For each pair of $[t_i, t_{i+1}]$ in S_D , the **time gap** $\Delta_t = t_{i+1} - t_i$, and $\Delta_t < \Gamma_g$, where Γ_g is a system defined parameter.

In the experiments of the prediction model learning process, we set Γ_g as 10 minutes. With the **daily speed-time series** files of the past dates, our model can learn to predict the speed variation given a query date. This is the problem we aim to solve in this work. Formally speaking, the output will be a sequence of **daily speed-time series**, but for a future date on query D_q , and since the predicted sequence represents the average traffic state at each time scale of query date, the time distance between each element is also equalized. We defined the predicted time-speed sequence, which is the output of our prediction model, **as the all-day speed pattern**.

Definition 3. (*All-day Speed Pattern*): The **all-day speed pattern** S_{D_q} is a **speed-time series data** $[(s_1, t_1), \dots, (s_n, t_n)]$, where $\forall t_i \in [t_1, \dots, t_n]$, t_i is located on the time scales during $[00 : 00 \ 24 : 00]$ on the queried date D_q . For each pair of $[t_i, t_{i+1}]$ in S_{D_q} , $\tau_t = t_{i+1} - t_i$ is a system defined parameter.

In the experiments of the prediction model learning process, we set τ_t as 5 minutes, and all the speed values use with km/hour units. With the standard input and output format defined, we then present the problem definition below.

Definition 4. (*All-day Speed Pattern Query*): The **all-day speed pattern query** is that, given a road segment r , a query date D_q , and its day features $df(D_q) = \langle f_1, f_2, \dots \rangle$, the **Speed Prediction Model** returns an **all-day speed pattern** S_{D_q} .

Another form of speed prediction problem is the query of a single time on the given query date. Considering that the system applying this prediction model may send a query of this kind, we give the alternative form of our problem definition below.

Definition 5. (*Single-time Speed Query*): The **single-time speed query** is that, given a road segment r , a query date D_q with its day features $df(D_q) = \langle f_1, f_2, \dots \rangle$, and the query time t_q , the **Speed Prediction Model** returns a speed value s_q by referencing the **all-day speed pattern** S_{D_q} which is simultaneously output by the **Speed Prediction Model**, where $s_q = s_i$ and $t_i \leq t_q < t_{i+1}, (t_i, s_i) \in S_{D_q}$.



Chapter 4

Overview of Prediction Model:

Framework and Method

In this section, we first present our speed prediction framework with Figure 4.1, and briefly introduce the 2-phase clustering method with Figure 4.2.

4.1 Speed Prediction Based on Clusters

The main goal of our model is to predict the *all-day speed pattern* given a query date of a target road segment based on historical speed records. Since the *all-day speed pattern* is figured out at once, the single speed value at any queried time point during the queried date can be obtained directly by looking up the *all-day speed pattern*. Moreover, according to the structure of our **2-phase prediction model**, the computation process can be separated into two parts, the 1st phase operation and the 2nd phase operation. When receiving the queried date and road segment, the **1st prediction model** outputs the *all-day speed pattern* along with the peak-time interval information. After the 1st phase operation, the queried time point will be labeled as either *normal-time query* or *peak-time query* if the query is a *single-time speed query* instead of an *all-day speed pattern query*. The *normal-time query* in fact only needs the 1st prediction process. Note that the peak-time speed pattern is estimated in the **1st prediction**

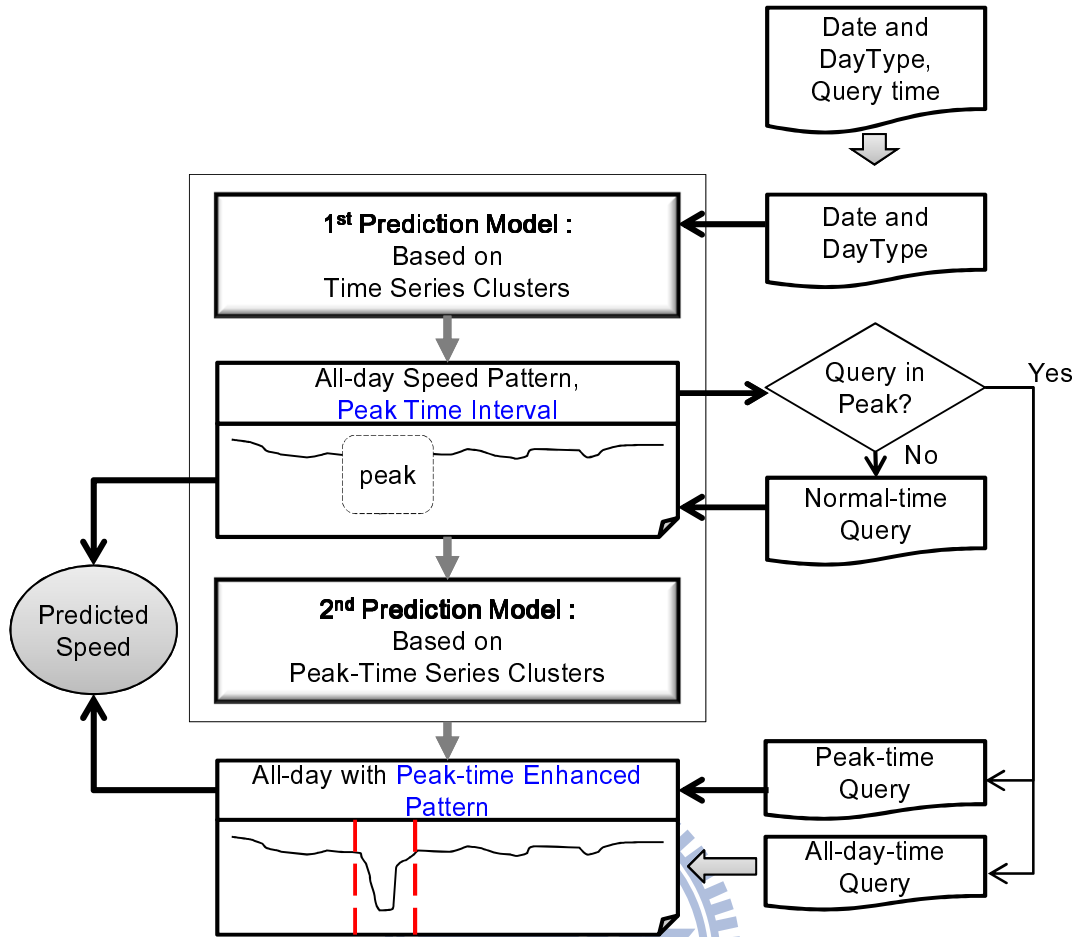


Figure 4.1: Speed prediction framework

model, too, but the speed values will be more precise after the 2nd phase operation. Hence, if the system requires a more precise prediction for a *peak-time query*, the 2nd phase operation should be executed then. By the **2nd prediction model**, the revised peak-time pattern is computed and embedded back into the *all-day speed pattern*. Consequently, after the 2nd phase operation, the model can answer either a *peak-time query* or an *all-day speed pattern query* precisely.

4.2 Two-Phase Clustering Flow

As shown in the Figure 4.2, we have developed a 2-phase model to solve the speed prediction problem. The 2-phase prediction model learns to predict using the 2-phase clustering method.

For each target road segment, the driving speed has been continuously recorded day by day during a past period such as two months. These raw traffic data are initialized as a *daily speed-time series* as defined in the third section. These daily files are the input of the 2-phase clustering method. In the first phase, the time series similarity measuring method such as DTW is adopted to evaluate the distance between arbitrary pairs of *daily speed-time series*. Then, for clustering the time series data, the classic clustering methods like K-means, hierarchical clustering, and DBSCAN are all implemented. After the first time clustering, the input *daily speed-time series* turns into several clusters, and some are noise. For those series in the same cluster, the general speed variation during a whole day is viewed as more similar than it is for other series which are not in their cluster. In the next phase, the method digs more deeply into the shape of the peak-time patterns, where speed usually drops intensely to a valley and climb back to normal driving speed. Here we develop a *peak finding approach* to detect and cut out the peak-time interval by scanning the *all-day speed pattern* at once. The *daily speed-time series* in the same cluster is averaged into one *all-day speed pattern* firstly, then the *peak finding approach* is applied to find every representative peak-time interval from each cluster's *all-day speed pattern*.

In the second phase clustering, the *daily speed-time series* in one cluster will be separated into sub-clusters according to their peak-time pieces. That is, for each *daily speed-time series* in one cluster, the sequence segment fitting the peak-time interval of its own cluster will be extracted. These peak-time pieces of *speed-time series data* are the input files of the second phase clustering. The steps of the second phase clustering are the same as those of the first phase, but they are performed for each of the cluster outputs from the first phase clustering in order to find their own sub-clusters focusing on peak-time patterns.

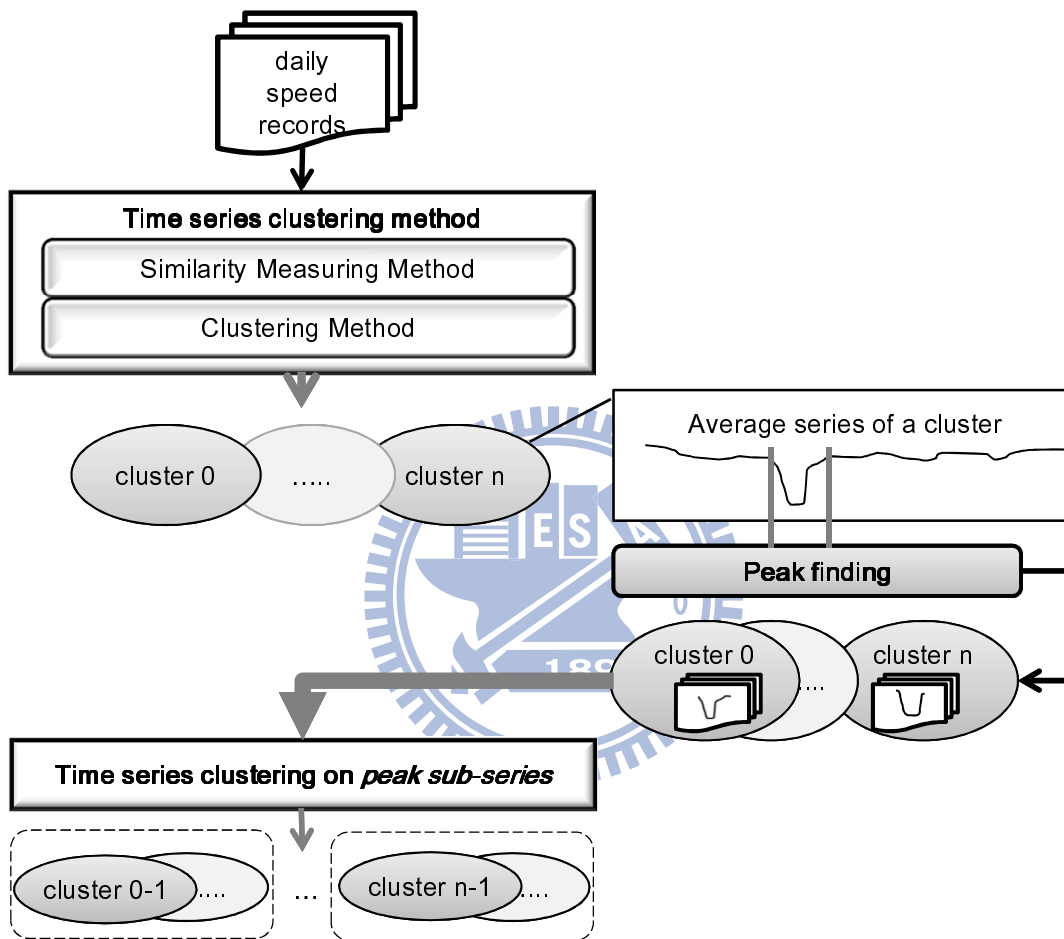


Figure 4.2: Two phase clustering flow

Chapter 5

Two-phase Clustering Method

In the last section we introduced the framework of our 2-phase clustering method. The details of each steps of the clustering method are described in the following sections.

5.1 The Distance Measuring of Speed-Time Series Data

To cluster the *speed-time series data*, we need to measure the closeness between every input sequence of *speed-time series data*. In the clustering process, each piece of *speed-time series data* can be viewed as a point located in an abstract space, and the distances between these points are usually figured by *time series distance functions*. It can be said that a time series distance function quantifies the distance between the sequences of *time series data* as points in the clustering space. Based on the observation in [11], which was also described in the related works section, we chose to implement the elastic measuring methods DTW, LCSS, EDR and ERP for our *speed-time series data* distance function. One important point about *speed-time series data* is that *time shifting constraint* needs to be added when applying elastic measuring methods. That is, the range of local time shift should be limited. We can not say that the speed falling during morning hours is the same behavior as speed falling during night hours, although the speed falling slope may look alike in the two cases. This is for an intuitive reason. We define the *time shifting constraint* as ω . For example, if we set $\omega = 30(\text{minutes})$, the similarity of any two elements from the two sequences can be counted only when their

Table I: Notations

Symbols	Meaning
S	a speed sequence with timestamps $[(s_{1,v}, s_{1,t}), \dots, (s_{n,v}, s_{n,t})]$
s_1	the 1 st element vector of S
$s_{1,v}$	the speed value of 1 st element vector of S
$s_{1,t}$	the timestamp of 1 st element vector of S
$\text{Rest}(S)$	the sub-sequence of S without the first element: $[(s_{2,v}, s_{2,t}), \dots, (s_{n,v}, s_{n,t})]$
ω	time shifting constraint of elastic measuring
ϵ	matching threshold on speed for edit distance based measuring methods: EDR, LCSS
g	a constant value for ERP computing the distance for gaps

time distance is still below 30 minutes. The similarity measuring methods for *speed-time series data* are formally described below, along with the notation table used in the formulas.

$$DTW(R, S) = \begin{cases} 0 & \text{if } m = n = 0 \\ \infty & \text{if } m = 0 \text{ or } n = 0 \\ \infty & \text{if } |r_{1,t} - s_{1,t}| > \omega \\ |r_{1,v} - s_{1,v}| + \min\{DTW(\text{Rest}(R), \text{Rest}(S)), & \text{otherwise} \\ DTW(\text{Rest}(R), S), DTW(R, \text{Rest}(S))\} \end{cases}$$

$$ERP(R, S) = \begin{cases} \sum_1^n |s_i - g| & \text{if } m = 0 \\ \sum_1^m |r_i - g| & \text{if } n = 0 \\ \infty & \text{if } |r_{1,t} - s_{1,t}| > \omega \\ \min\{ERP(\text{Rest}(R), \text{Rest}(S)) + |r_{1,v} - s_{1,v}|, & \text{otherwise} \\ ERP(\text{Rest}(R), S) + |r_{1,v} - g|, \\ ERP(R, \text{Rest}(S)) + |s_{1,v} - g|\} \end{cases}$$

$$EDR(R, S) = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ \min\{EDR(\text{Rest}(R), \text{Rest}(S)) + \text{subcost}(r_1, s_1), & \text{otherwise} \\ EDR(\text{Rest}(R), S) + 1, EDR(R, \text{Rest}(S)) + 1\} \end{cases}$$

, where $\text{subcost}(r_1, s_1) = 0$ if $|r_{1,t} - s_{1,t}| \leq \omega \wedge |r_{1,v} - s_{1,v}| \leq \epsilon$

and $\text{subcost}(r_1, s_1) = 1$ otherwise

Table II: Notations

	Time shifting	Time scaling	Noise	Amplitude distance	Metric
Euclidean			very sensitive	delicate	✓
DTW	✓	✓	relatively sensitive	delicate	
ERP	✓	✓	relatively sensitive	delicate	✓
LCSS	✓	✓	available	coarse	
EDR	✓	✓	available	coarse	

$$LCSS(R, S) = \begin{cases} 0 & \text{if } m = 0 \text{ or } n = 0 \\ LCSS(Rest(R), Rest(S)) + 1 & \text{if } |r_{1,t} - s_{1,t}| \leq \omega \wedge |r_{1,v} - s_{1,v}| \leq \epsilon \\ \max\{LCSS(Rest(R), S), \\ LCSS(R, Rest(S))\} & \text{otherwise} \end{cases}$$

Among the four methods, DTW was first proposed as an *elastic distance function* that aims to find the optimal alignment between two time series sequences because the earliest method, Euclidean distance, has been found to be very weak at handling noise and local time shifting. DTW can handle local time shifting, but is still sensitive to noise. Later, LCSS and EDR were proposed to measure distance allowing the skipping of some points to match similar common subsequences. ERP is another method that handles local time shifting and remains metric. In this work, we want to find the distance function that can deal with local time shifting under a time shifting constraint, and can which deal adequately with noise, but which does not allow too much amplitude shifting. We may even need two kinds of distance functions of which one is optimal to find the general pattern in the first phase while the other works better when finding the more delicate peak-time patterns. We first list the feature table, Table. II, of the four distance functions, and the availabilities of these functions are evaluated in the experiment section.

5.2 Clustering By Time Series Data Similarities

In this work, our main goal is to predict the one-day speed pattern given a query date and the target road segment. With this pattern, the prediction model can report an estimated

driving speed given any queried time point during the day.

By observing the historical speed-time records, it can be discovered that the traffic behavior of one road reflects similar patterns at the same time on different days. We further find that there exist a few kinds of one-day patterns on different day types. For example, the peak and off-peak time intervals are located very closely on some of the weekdays. However, when it comes to the weekends, the traffic behavior is very likely to exhibit other patterns. In fact, weekdays do not always exhibit only as one pattern, and moreover, Saturdays and Sundays usually exhibit different patterns for some of the roads. It is obvious that categorizing traffic patterns according to weekday and weekend is not specific enough. Due to this observation, we believe that some basic day-types should have their own traffic behavior patterns. For example with Figure 5.1, in our observing experiment of a chosen highway segment in Taiwan (Chubei-to-Hsinchu), the Sunday time-speed records in May 2011 are very similar, appearing to be peaceful and close to the speed limit for the whole 24 hours. Hence, if the path searching system requires referencing the traffic condition of the Chubei-Hsinchu segment on a day that is a Sunday, the prediction model can report that the speed value should be close to the speed limit at all the query times throughout the whole day.

With the above observations, we have come up with the concept of finding the day-types with their own speed patterns of a target road segment. For example, we know that it is not specific enough to only define two day-types as weekdays and weekends, and the fact is that if we gather all the Monday records of one road, they are not all the same kind of pattern; hence we need to do clustering on the daily speed-time series data, and identify rules to conduct different day-types to different clusters. However, the detailed shape of the peak-time pattern may further turn into two more variations. Take the Saturday time-speed records of the Chubei-to-Hsinchu highway segment in May for example(Figure 5.1). Most of the Saturday traffic conditions tend to be congested (driving at a low speed) around noon; however, the length of the peak time and the amplitude of the speed drop varies. This is our motivation for proposing the second phase clustering, which aims to find more detailed types of peak-time traffic patterns.

Since we decided to take clustering as our mining method, it is an issue to choose a clustering algorithm that can lead our model to the best performance and accuracy. In this work, we implemented K-means, hierarchical clustering and DBSCAN, which are three typical

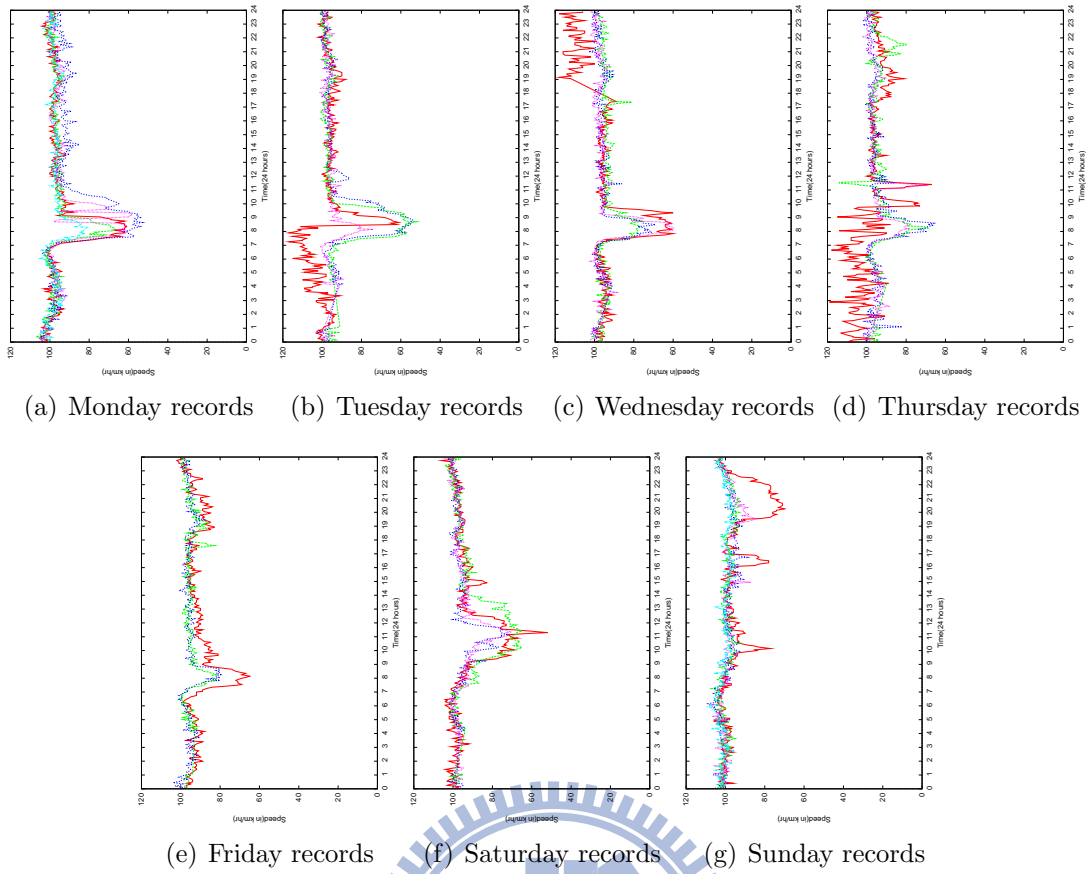


Figure 5.1: Real data example on the Chubei-to-Hsinchu highway segment recorded in May, 2011

kinds of clustering algorithms. K-means is the most basic Partitional clustering approach. At the initial step of K-means, the number of cluster centroids should be specified, that is, the value of K of K-means should be assigned. For the initial centroids of the clusters, we choose to randomly select them out. Afterwards, the K-means algorithm will recursively assign each point to the cluster with the closest centroid and recompute the centroid of each cluster. K-means terminates when the centroid of each cluster remains same as the one in previous iteration. Since we take every sequence of time series data as one point in our clustering process, the centroid of each cluster is the data point with the minimal aggregated distance to all the other points in the same cluster. For the distance measuring between each point to another, we have mentioned in the last subsection that we utilize the time series distance functions: DTW, ERP, LCSS, and EDR. The hierarchical clustering algorithm we choose to implement is the agglomerative clustering method. The basic agglomerative hierarchical clustering method starts with the points as individual clusters, and merges the closest pair of

clusters at each step until only one cluster is left. There are two parameters that we need to determine when applying the agglomerative clustering method to our model.

- *plev*: The *partition level* of the agglomerative hierarchical tree structure that partitions the tree structure into its sub-trees at the cutting level *plev* as output clusters. The *plev* of the root in the tree is set as 0.
- *linkType*: The definition of proximity between clusters, i.e., how to measure the distance between clusters. Here we implement three basic proximity types for our agglomerative hierarchical clustering: *MIN*, *MAX* and *Group average*.

The above mentioned three types of proximity definitions are the simplest ones among existing methods. *MIN* defines cluster proximity as the proximity between the closest two points in different clusters. On the contrary, *MAX* defines that with the farthest. *Group average* defines proximity as the average pairwise proximities of all pairs of points from different clusters.

Still another clustering method for our model is DBSCAN, the most common density-based clustering algorithm. For DBSCAN, two parameters should be given:

- *Eps*: The maximum radius of the neighborhood
- *MinPts*: The minimum number of points in an *Eps*-neighborhood of that point

In our experiments, the *MinPts* is initialized to be 2. After the distances between all the input elements, that is, all the sequences of *daily speed-time series*, are measured, we then set *Eps* as the value between minimum $dist(R, S)$ returned value and the maximum $dist(R, S)$ returned value. Again, *dist* is one of the four time series distance functions: DTW, ERP, LCSS, and EDR, used to measure the distance between different sequences of time series data.

5.3 Speed Pattern Peak Finding

We referenced the peak finding Matlab tool and modified it into our own speed pattern peak finding method. As mentioned in the third section, before the second phase clustering, the peak-time interval should be figured out by inputting the **all-day speed pattern** $S_{D_q} =$

$[(s_1, t_1), \dots, (s_n, t_n)]$ to the peak-finding method. The peak finding steps can be summed up as the following list.

1. Smooth the sequence S_{D_q} by *sliding-average smoothing*
2. Calculate the *first derivative* of smoothed S_{D_q}
3. Scan the *first derivative* sequence to find the *zero-crossing* point.
4. Start the two-way sliding window from the *zero-crossing* point, as the two-way sliding window respectively moves toward left and right, determine whether it conforms to the *peak derivative requirements* and stop the two-way sliding until all the requirements are totally satisfied, or stop when any conflict with the requirements occurs.

In fact, after step 3, the existing method uses *least-squares curve-fitting* to estimate the position, height, and width at each detected peak location, and ignores the noise single which does not fit the threshold of the peak height and width. However, the existing method is used when the peak-time interval is required to be very accurately estimated, while the *least-squares curve-fitting* needs extensive computation because it is used to infer the polynomial equation fitting the given sequence data, and then uses the equation coefficients to infer the peak height and width. This method is not efficient when applying it to the peak-finding case in this work. Since the motivation for finding the peak-time interval of a speed pattern is that we can cut out the peak-time pieces and do more detailed clustering on the peak-time pattern, the peak-time interval estimation do not need to be very accurate, because even if we make the peak-time interval slightly longer, the clustering quality will not be influenced, and we can still save the real peak-time data points. On the other hand, our own solution to find the peak-time interval in the fourth step can still estimate the interval correctly, and it is only necessary to scan the sequence **once**, with time complexity $O(n)$. In the following subsections, we describe each step in detail.

5.3.1 Sliding-average smoothing

This smoothing simply replaces each point in the sequence with the average of m adjacent points, where m is a positive integer called the *smooth width*. Given speed-time series data

$S = [(s_1, t_1), \dots, (s_n, t_n)]$ and a *smooth width* m , the sliding-average smoothing function can be expressed as

$$\tilde{S}_j = \left\{ \frac{\sum_{k=j-lh}^{k=j+rh} s_k}{m}, t_j \right\},$$

where $lh = \lfloor \frac{(m-1)}{2} \rfloor$ and $rh = \lceil \frac{(m-1)}{2} \rceil$,

for $j = lh$ to $n - rh$, n is the length of input sequence S

, in which \tilde{S}_j is the j^{th} element in the smoothed sequence \tilde{S} for original sequence S . In the experiments, we set $m = 3$. Figure 5.2(a) shows an example, where the cross-dotted sequence is the original sequence, a real sample of *speed-time series data*, and the green-dotted sequence is the smoothed sequence where $m = 3$. Smoothing eliminates the sawteeth shape of the raw sequence. It is useful because the very small or narrow peaks appearing like sawteeth are noise when we are finding a "traffic jam time" from the real world data.

5.3.2 First derivative sequence

The first derivative of a sequence is the rate of change of y coordinate to x coordinate, in mathematics, dy/dx , which is interpreted as the **slope** of the *tangent* to the sequence at each point. Here we use the most simple differentiation concept to calculate the first derivative of the given *speed-time series data*.

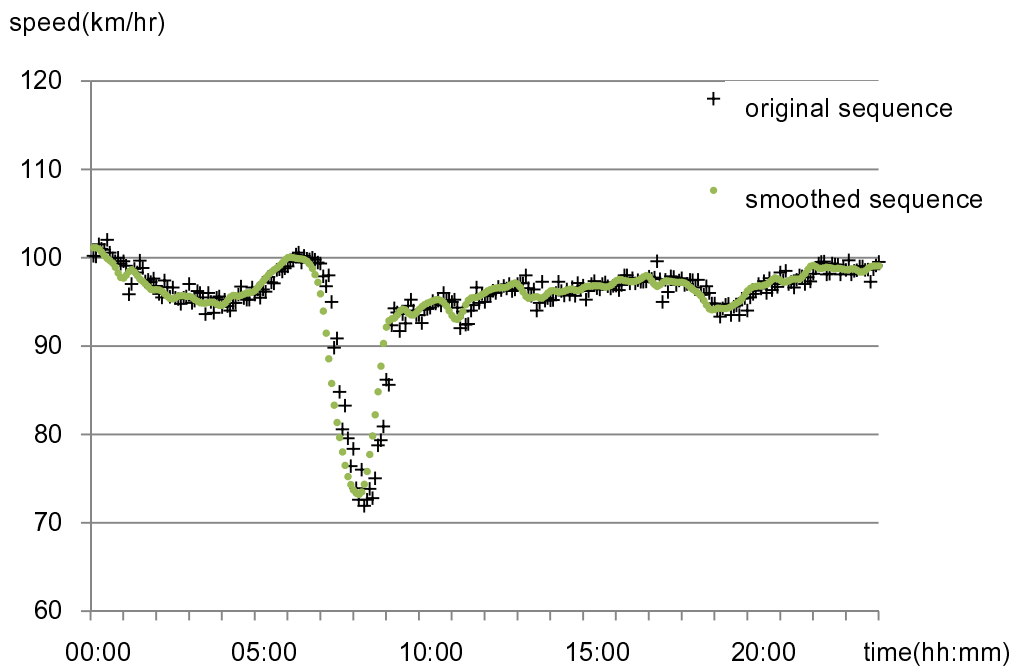
$$S'_j = \left\{ \frac{s_{j+1} - s_{j-1}}{t_{j+1} - t_{j-1}}, t_j \right\},$$

for $j = 2$ to $n - 1$, n is the length of input sequence S

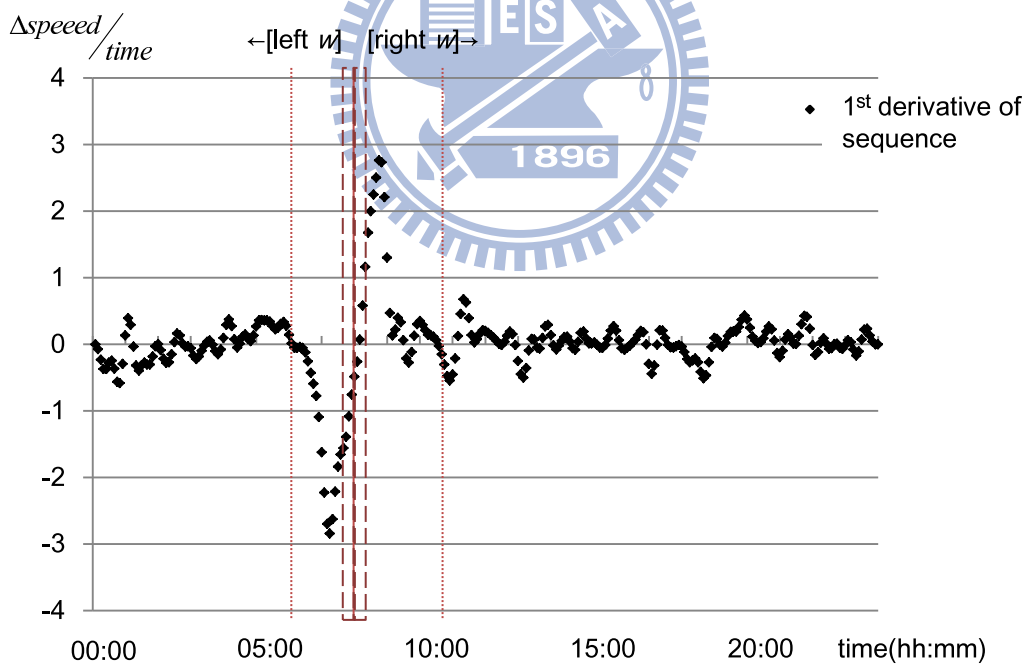
, in which S'_j is the j^{th} element in the first derivative sequence S' for original sequence S . Since the input sequence is an *all-day speed pattern* defined in section.3, each time scale $t_{i+1} - t_i$ is a constant τ_t . Hence, $t_{j+1} - t_{j-1}$ in the above equation is in fact equal to 2τ , which is a constant, so we not then need to normalize the derivative.

5.3.3 Two-way sliding window

According to the meaning of *tangent* value presented in Section 5.3.2, the critical point in the original peak curve is the *zero-cross point* in the first derivative curve. Before the critical



(a) smoothing step



(b) first derivative and two-way sliding window

Figure 5.2: Peak finding sample

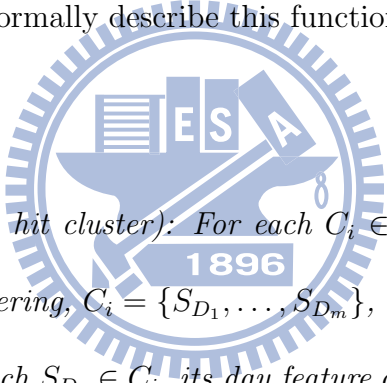
point is a speed falling series from a normal peace curve, hence the *tangent* value should be negative. During the speed falling interval, in the normal traffic jam case, the driving speed starts to decrease with a small degree and then decreases faster and faster. When the traffic flow nearly reaches saturation point, the decrease in speed becomes slow again. Mapping to the first derivative, before the zero-crossing point, the negative point breaks up the peace line composed of a series of zero-around points and then the following are all negative points, and the absolute value of the negative point first keeps increasing until achieving the top(largest absolute y-axis value) and then decreasing until it arrives at the *zero-cross point*. Vice versa, after the zero-cross point, the shape of the positive sequence in the first derivative is symmetrical to that in the negative sequence, except that the y-axis value is located in the positive field. To sum up, we want to use the above mentioned behavior of the first derivative sequence while mapping to the peak interval of the original sequence.

Here we propose a two-way sliding window to help us detect the start time and end time of the "during-peak pattern" of the first derivative sequence. Intuitively, the rightward sliding window aims to detect the end point of the "during-peak pattern", which moves towards 24:00, while the leftward sliding window moves in the opposite direction, towards 00:00, until it detects the start point of the "during-peak pattern". The y-axis absolute value of points in the current sliding window is averaged, so according to the "during-peak pattern", this value should first climb, and then fall, until the value in the next sliding window does not fall anymore. In our algorithm, we add a *buffer mechanism* to the two-way sliding window. This is to avoid being too sensitive to some slight vibration of the derivative sequence, while still conforming to the during-peak features in general.

Chapter 6

Speed Pattern Prediction Function

Prediction Function Based on WeekID: This prediction chooses the speed pattern of the cluster containing the maximum number of the same weekID days identical to the queried date weekID. We formally describe this function by the following definitions.



Definition 6. (*first phase hit cluster*): For each $C_i \in \langle C_1, \dots, C_n \rangle$, the set of output clusters of first-phase clustering, $C_i = \{S_{D_1}, \dots, S_{D_m}\}$, where m is the element numbers contained in C_i , and for each $S_{D_j} \in C_i$, its day feature $df(D_j) = \langle \text{WeekID on } D_j \rangle$, and $\text{WeekID} = \{\text{Mon}, \text{Tue}, \text{Wed}, \text{Thu}, \text{Fri}, \text{Sat}, \text{Sun}\}$. Let $\text{Week_num}(\text{WeekID}, C_i) = |\{S_D | S_D \in C_i, \text{ and } df(S_D) = \text{WeekID}\}|$. Given a query date D_q , the **first phase hit cluster** is the cluster having $\max(\text{Week_num}(df(D_q), C_i))$ among the clusters set $C_i \in \langle C_1, \dots, C_n \rangle$.

Definition 7. (*first phase pattern*): Given a query date D_q , the **first phase pattern** is the all-day speed pattern averaged from all the **daily speed-time series**

$\{S_{D_1}, \dots, S_{D_m}\}$ in the **first phase hit cluster** C_{first} .

Definition 8. (*second phase hit cluster*): Given the **first phase hit cluster** C_{first} , with its sub clusters set output from the second-phase clustering $c_{first,i} \in \langle c_{first,1}, \dots, c_{first,n} \rangle$, and a query date D_q , the **second phase hit cluster** is the cluster with $\max(\text{Week_num}(df(D_q), c_{first,i}))$ among the cluster set $c_{first,i} \in \langle c_{first,1}, \dots, c_{first,n} \rangle$

Definition 9. (*second phase revised peak-time pattern*): Given a query date D_q and the **second phase hit cluster** $c_{first,second}$, the **revised peak-time pattern** is the **speed-time series data** averaged from all the **speed-time series data** $\{S_{peak,D_1}, \dots, S_{peak,D_m}\}$ in the **second phase hit cluster** $c_{first,second}$, where each $S_{peak,D_i} \in c_{first,second}$ is the cropped part of S_{D_i} , i.e.; $[(s_1, t_1), \dots, (s_n, t_n)]$, where $\forall t_i \in \langle t_1, \dots, t_n \rangle$, $peak_{start} \leq t_i < peak_{end}$.

Prediction Function Based on Max Probability: This prediction directly choose the biggest cluster and take its average speed pattern as the predicted pattern, no matter whether in the first phase or the second phase clustering. The **first phase pattern** and **revised peak-time pattern** definitions are identical to those in the **Prediction Function Based on WeekID**, so we do not repeat them in this part.

Definition 10. (*first phase hit cluster*): For each $C_i \in \langle C_1, \dots, C_n \rangle$, the set of output clusters of first-phase clustering, $C_i = \{S_{D_1}, \dots, S_{D_m}\}$, where m is the element numbers contained in C_i . Let $cluster_size(C_i) = |\{S_D | S_D \in C_i\}|$. Given a query date D_q , the

first phase hit cluster is the cluster with $\max(\text{cluster_size}(C_i))$ among the cluster set $C_i \in \langle C_1, \dots, C_n \rangle$.

Definition 11. (*second phase hit cluster*): Given the **first phase hit cluster** C_{first} , with its sub clusters set output from the second-phase clustering $c_{first,i} \in \langle c_{first,1}, \dots, c_{first,n} \rangle$, and a query date D_q , the **second phase hit cluster** is the cluster with $\max(\text{cluster_size}(c_{first,i}))$ among the cluster set $c_{first,i} \in \langle c_{first,1}, \dots, c_{first,n} \rangle$

WeekID and Max Probability Feature Combined Prediction This prediction uses the weekID based prediction function as the first phase prediction, and utilizes the max probability based prediction function as the second phase prediction.

first phase hit cluster:

The **first phase hit cluster** is the same as **definition.6**, that is, given a query date D_q , the **first phase hit cluster** is the cluster with $\max(\text{Week_num}(df(D_q), C_i))$ among the cluster set $C_i \in \langle C_1, \dots, C_n \rangle$

second phase hit cluster:

The **second phase hit cluster** definition is identical to **definition.11**, that is, given a query date D_q , the **second phase hit cluster** is the cluster with $\max(\text{cluster_size}(c_{first,i}))$ among the cluster set $c_{first,i} \in \langle c_{first,1}, \dots, c_{first,n} \rangle$.

Chapter 7

Experiments

In this section, we first introduce our data sets and related setting in Section 7.1.1, then describe how we measure the performance of our method in Section 7.1.2. In Section 7.2, we show the performance of our approach adopting the different clustering methods, and compare all of the clustering results to another time series forecasting method, ARIMA. In Section 7.3, We analyze the performance of different models we have proposed in this work. Furthermore, we study the effects of using different time series distance functions in our two-phase clustering approach in Section 7.3. Table I lists the notations used in this section.

Table I: Notations in experiment

Γ_g	Boundary of time gap in the Definition 2, <i>daily speed-time series</i>
ω	Time shifting constraint of DTW, ERP, LCSS and EDR
K	The number of clusters to be determined in the initial step of K-means
$plev$	The <i>partition level</i> of the agglomerative hierarchical tree structure
$linkType$	The definition of proximity between clusters
$MinPts$	DBSCAN parameter, default value is 2 in experiment
Eps	DBSCAN parameter to be adjusted in experiment

7.1 Datasets and Setting

7.1.1 Data Preparation

Traffic data source: The real data set we used is acquired from the freeway traffic database of the Taiwan National Freeway Bureau[1], with access to the database authorized in advance. The official freeway traffic database receives real-time traffic data from all the inductive loop sensors spread over Taiwan National Freeway Road Network every 3 to 5 minutes, and the data received at every update time point were recorded as an XML file. Simultaneously, the server in our laboratory copies the real-time updated XML file every 5 minutes. In this way, our server stored the historical traffic data of the freeway road network for more than one year.

Data set preparation: We extracted the speed data that was recorded by our target sensor from a series of historical XML files, and transformed the speed data with time information into a ***daily speed-time series***, which is defined in Section 3. First, each data point we transformed from one XML file contains fields of speed (km/hr), year, month, date, week-day (Mon, Tue,..., Sun.), and time in the form of HH:MM. Second, all the data points are separated into individual files according to their *date* values. That is, each file contains the time series data with speed value during one day, 24hours, and each file is registered with the year, month, date, and week number. Hence, the time series data in one date file obeys Definition 2 in Section 3, the ***daily speed-time series*** definition, which is a data sequence $[(s_1, t_1), \dots, (s_n, t_n)]$, with **time gap** $\Delta_t = t_{i+1} - t_i < \Gamma_g$, and here Γ_g is 10 minutes for our data set while the average Δ_t is 5 minutes. Hence, the average time series data size in one file is 288 elements. The visualization sample can be referenced to Figure 5.1. In Figure 5.1, although the files are grouped according to week, we can see that each colored line is one time series of a file for one date.

Selected sensor: The target sensor for our evaluation is on the freeway segment from Chubei to Hsinchu. Simultaneously, its opposite direction segment, that is, Hsinchu to Chubei, is also selected as another target sensor. Chubei to Hsinchu is the most frequently passed segment by the commuters in northern Taiwan, because Hsinchu Science Park in Hsinchu city is the capital science industrial estate in Taiwan, and employs a huge number of employees in northern

Taiwan. Hence, the car flow from north cities to Hsinchu on working day mornings must pass along the Chubei-to-Hsinchu freeway segment, and usually causes obvious jams. We adopted this segment as the testing segment because we expected that it would have some interesting regular traffic behaviors. On the other hand, the Hsinchu-to-Chubei segment is known for traffic congestion in the evening when the commuters finish work. However, the Hsinchu-to-Chubei traffic behavior is more complicated than that of the Chubei-to-Hsinchu segment, as more kinds of drivers and not only the Science Park commuters frequently use this segment. Hence, in our experiment section, we consider two classic traffic cases, **(1)** the segment frequently passed by drivers to their workplace in the morning, and **(2)** the segment frequently passed by drivers leaving their workplace in the evening; at the same time, traffic in (2) is with less regular than (1) so we can evaluate the performance of our method under different conditions of traffic complexity.

7.1.2 Evaluation Measure

Training and testing steps: In this experiment, we picked data for the days in March and April, 2011, giving two months data as training data for our traffic prediction model. As for the testing data, we picked the days from two weeks in May, 2011. In the training process, as mentioned in last subsection, the input time series files were transformed from the training data set in the form of Definition 2 ***daily speed-time series***. Then, the files were analyzed by the two-phase clustering method, and so the prediction functions were set. In the testing process, the ground truth files were transformed from the testing data, and were also in the form of a ***daily speed-time series***. The queries were given the dates and weekday (Mon, Tue, Wed, Thu, Fri, Sat, Sun) and for each queried date, we picked 15% of the *normal-time points* randomly and 15% of the *peak-time points* randomly to test according to that date's ground truth.

Performance metrics: The performance of this work is evaluated by the *average speed error(km/hr)*, that is, the average speed difference between the predicted speed and the ground truth speed. In our experiment, we measured the model performance in ***normal-time query*** and ***peak-time query*** separately. Consequently, we have two kinds of averaged speed error, ***normal-time error*** and ***peak-time error***. *Parameters of clustering methods:* Since we

implemented three clustering algorithms: K-means, hierarchical clustering and DBSCAN, in our model, when evaluating the performance of our approach with each clustering method, we also needed to adjust the parameters for each clustering method and find the best value of their parameters. These parameters are listed in Table I above and have been clarified in Section 5.2. For K-means, the K should be adjusted. For agglomerative hierarchical clustering, the best $plev$ and $linkType$ should be found. For DBSCAN, we should find the best value of $MinPts$ and Eps .

Models to be evaluated: In this section, we discuss the performance of our proposed models adopting three different prediction functions: (1) **Prediction Function Based on WeekID** (2) **Prediction Function Based on Max Probability**, and (3) **WeekID and Max Probability Feature Combined Prediction**. Besides comparing the three different models, we also tested the performance of the single-phase version of each model, that is, each model only consisted of the 1st prediction model and did not use the 2nd prediction model to deal with *peak-time queries* (Figure 4.1). However, the model(3) **WeekID and Max Probability Feature Combined Prediction** means that it used the WeekID based prediction function for the first phase and the Max Probability based prediction function for the second phase, so it did not have a single-phase version since it is originally a two function mixed two-phase model. To sum up, we have three main models to evaluate, and for model(1) and model(2), there are two versions, the two-phase version and the single-phase version. In addition, for each model, we have two kinds of speed error, *normal-time error* and *peak-time error*. The two-phase version of models(1)(2)(3) have *normal-time error* and *peak-time error*, with *peak-time error* output by their second phase prediction. The single-phase version of models(1)(2) have *normal-time error* as in the two-phase version, so we only put on their *peak-time error*. On the other hand, model(3) has the same *normal-time error* as that of model(1) because they all use the WeekID based prediction function, so we also omit the *normal-time error*. Consequently, there are seven error values in one chart here, listed as the following:

1. weekID based prediction model

- normal-time error
- single-phase peak-time error
- two-phase peak-time error

2. max probability based prediction model
 - normal-time error
 - single-phase peak-time error
 - two-phase peak-time error
3. weekID and max probability mixed prediction model
 - two-phase peak-time error

7.2 Clustering Method Performance

First of all, the accuracy is not only influenced by clustering algorithm but also by the time series similarity measurement and the type of prediction function which we have defined in Section 6. In this section we want to find the most suitable clustering algorithm for our two-phase clustering approach. Afterwards, in the following sections, we discuss the effect of adopting different time series similarity measurements and prediction functions. Hence, the best performances achieved by the different clustering methods by adjusting other factors (time series similarity measurement, prediction function) are selected for comparison. We evaluated the performance of three clustering methods: K-means, hierarchical and DBSCAN in our two-phase clustering framework. Furthermore, we implemented ARIMA (Autoregressive Integrated Moving Average) [6], a popular statistical approach for time series forecasting, to compare with our two-phase clustering approach.

Figure 7.1 and Figure 7.2 show the best performance achieved by the three clustering methods of the two-phase clustering model compared with ARIMA. First, for the speed error in normal time, Figure 7.1 shows that either with the Chubei-to-Hsinchu data set (Figure 7.1(a)) or the Hsinchu-to-Chubei data set (Figure 7.1(b)), our prediction models using clustering methods are better than ARIMA. K-means shows the lowest error in the Chubei-to-Hsinchu data set; however it has the highest error rate in the Hsinchu-to-Chubei data set among the three clustering methods. On the other hand, DBSCAN has a slightly higher error rate than K-means in the Chubei-to-Hsinchu data set, but has the lowest error rate in the Hsinchu-to-Chubei data set. The hierarchical method has the highest error rate among the three

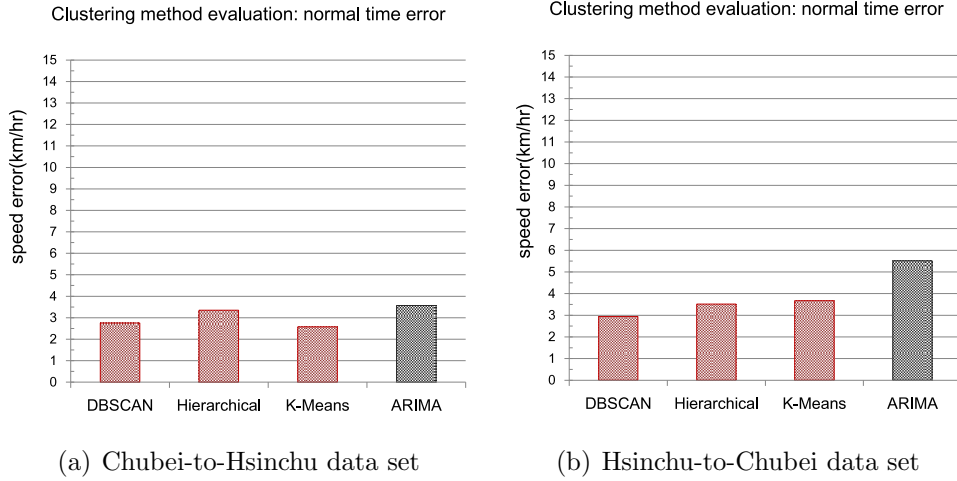


Figure 7.1: Clustering method and regression method comparison(1): speed error in normal time

methods in both data sets. Second, Figure 7.2 shows the speed error in peak time when each clustering method adopts the single-phase clustering approach and the two-phase clustering approach. We can see that either with the Chubei-to-Hsinchu data set (Figure 7.2(a)) or the Hsinchu-to-Chubei data set (Figure 7.2(b)), our *two-phase* prediction models using clustering methods are better than ARIMA, although the hierarchical method in the single-phase prediction model has speed error greater than ARIMA. This shows that the two-phase clustering model improves the prediction accuracy in reality, no matter what clustering method it adopts. Looking into the results of the three clustering methods, DBSCAN in the two-phase prediction model succeeds in both the Chubei-to-Hsinchu data set and the Hsinchu-to-Chubei data set, while K-means has error very close to DBSCAN in the Hsinchu-to-Chubei data set. Considering the performances in both normal time speed prediction and peak time speed prediction, DBSCAN in the two-phase prediction model has the stability of performing with higher accuracy among all the methods; hence we will focus on the two-phase clustering approach adopting the DBSCAN method and discuss the effects of the other two factors: time series similarity measurement and prediction function with the DBSCAN clustering model in the following sections.

Table II: The setting of clustering parameters of the best performances

Clustering method	1st-phase setting	2nd-phase setting
K-means	$K = 4$	$K = 2$
hierarchical	$plev = 2$	$plev = 3$
hierarchical	$linkType = \text{group average}$	$linkType = \text{group average}$
DBSCAN	$MinPts = 2$	$MinPts = 2$
DBSCAN	$Eps = 0.73 * data.length$ for LCSS	$Eps = 96$ (accumulated speed difference) for DTW

7.3 Evaluation of Prediction Models

We have the results of the clustering algorithm performance evaluation in Section 7.2: DBSCAN can lead our two-phase clustering approach method with to the best performance. In this section, with the DBSCAN clustering method, the best performance that can be achieved by different prediction models was found. For example, Figure 9(a) and Figure 10(a) show the *normal-time error*. According to the model list shown in Section 7.1.2, although model(1) and model(2) have single-phase and two-phase versions, the normal-time errors are directly output in the first phase; hence model(1) and model(2) have just one kind of normal-time error. Additionally, according to Section 7.1.2, model(3) uses the same prediction function in the first phase as model(1) does, so we also ignore the *normal-time error* of model(3) in all the following charts in this section. Figure 9(a) shows that model(1) (as well as model(3)) generally performs better than model(2) when dealing with *normal-time queries*. When the data set is changed to Hsinchu-to-Chubei, as shown in Figure 10(a), model(1) is better than model(2) for most of the time series distance functions.

Next, Figure 9(b)(c), Figure 10(b)(c) represent the *peak-time error*. The color bar labels the model type and the phase version. For example, $M(1) 2ph$ means that the error bar shows the best performance of Model(1) using the two-phase version to deal with *peak-time queries*, while $M(1) 1ph$ uses only the first phase of model(1) to deal with the *peak-time queries*. Clearly, in Figure 9 and Figure 10, it can be found that models in two-phase version, that is, $M(1) 2ph$, $M(2) 2ph$, and $M(3)$, perform better than their single version. This result proves that the second phase a the prediction model really improves the accuracy of the *peak-time queries*.

7.4 Evaluation of Time Series Distance Functions

We have proved in Section 7.3 that the two-phase clustering structure outperforms the single-phase clustering method. The usage of time series distance function in the first-phase influences both the accuracy of *normal-time error* returned in the first-phase and *peak-time error* returned in the second-phase. The usage of the time series distance function in the second-phase directly influences the *peak-time error* returned in the second-phase. We explain and conclude the results in the following subsections.

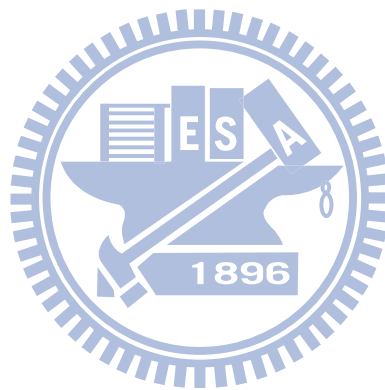
7.4.1 The Accuracy in Normal Time Prediction

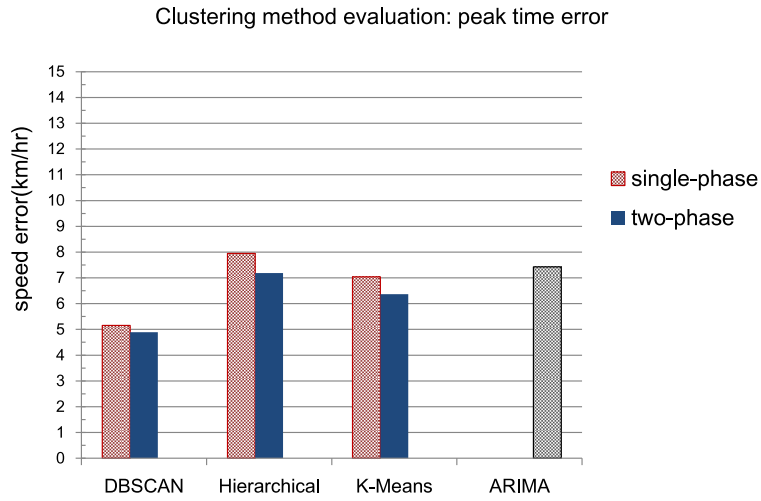
With the Chubei-to-Hsinchu data set, Figure 9(a) shows that LCSS outperforms the other three time series distance functions for *normal-time error*. With the Hsinchu-to-Chubei data set, the result is similar in that LCSS adopted to model(1), the weekID based prediction model, outputs the smallest speed error. The second most accurate results are provided by DTW. Now we know that if we adopt LCSS for the first-phase clustering we will have the smallest *normal-time error*; however, it is more important that adopting LCSS for the first-phase will lead to the smallest *peak-time error*.

7.4.2 Accuracy in Peak Time Prediction

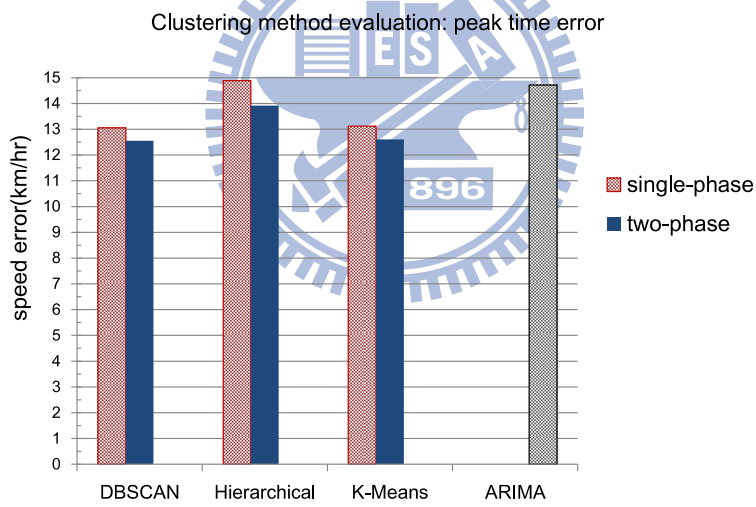
With the trying of all the time series distance functions for the first-phase clustering, we found that no matter what kind of distance function we used, adopting DTW for the second-phase clustering results in the smallest *peak-time error* when compared with using other distance functions in the second-phase clustering. Hence, we just need to compare four kinds of two-phase time series distance function combinations: DTW-DTW, ERP-DTW, EDR-DTW, and LCSS-DTW. Figure 9(c) shows the result for the Chubei-to-Hsinchu data set. In Figure 9(c), DTW-DTW and LCSS-DTW have very close results with the smallest error for model(3) (the weekID and max probability mixed prediction model). On the other hand, Figure 10(c) shows the result for the Hsinchu-to-Chubei data set. Still, DTW-DTW and LCSS-DTW outperform ERP-DTW and LCSS-DTW according to the average performance of the three models. However, we can see that the smallest *peak-time error* achieved by the LCSS-DTW

set with model(3), the weekID and max probability mixed prediction model (noted as Wk-max in figure). Moreover, the result of *normal-time error* in Figure 9(a) and Figure 10(a) also shows that using LCSS in the first-phase clustering produces the smallest error, which we previously have mentioned. Conclusively, with the comprehensive results from the two different data sets, we can say that using LCSS as the first-phase time series distance function and DTW for the second-phase suggests the best performance for our two-phase clustering approaches, while adopting DBSCAN as our clustering algorithm.





(a) Chubei-to-Hsinchu data set



(b) Hsinchu-to-Chubei data set

Figure 7.2: Clustering method and regression method comparison(2): speed error in peak time

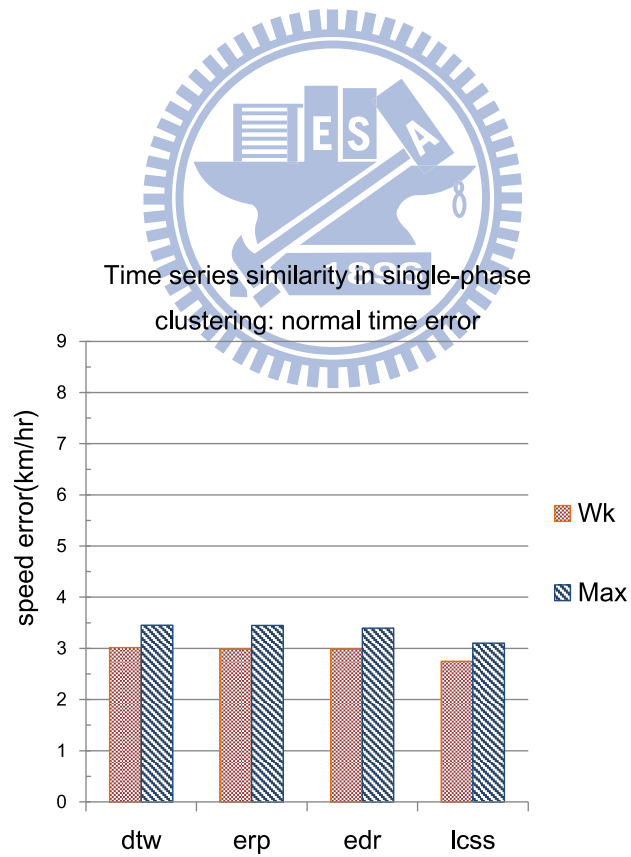
Clustering Method	Best case of normal error	Best case of peak error [single-phase clustering]	Best case of peak error [2-phase clustering]
DBSCAN	LCSS using f(Wk)	LCSS using f(Wk)	LCSS_DTW using f(Wk-max), DTW_DTW using f(Wk-max)
Hierarchical	LCSS using f(Wk) or f(Max)	LCSS using f(Wk) or f(Max)	LCSS_EDR using f(Wk)
K-Means	LCSS using f(Wk)	LCSS using f(Wk)	LCSS_DTW using f(Wk)

(a) Chubei-to-Hsinchu

Clustering Method	Best case of normal error	Best case of peak error [single-phase clustering]	Best case of peak error [2-phase clustering]
DBSCAN	LCSS using f(Wk)	EDR using f(Wk)	LCSS_DTW using f(Wk)
Hierarchical	LCSS using f(Max)	LCSS using f(Wk)	LCSS_EDR using f(Wk)
K-Means	LCSS using f(Wk)	LCSS using f(Wk)	LCSS_DTW using f(Wk)

(b) Hsinchu-to-Chubei

Figure 7.3: Conditions of best performance



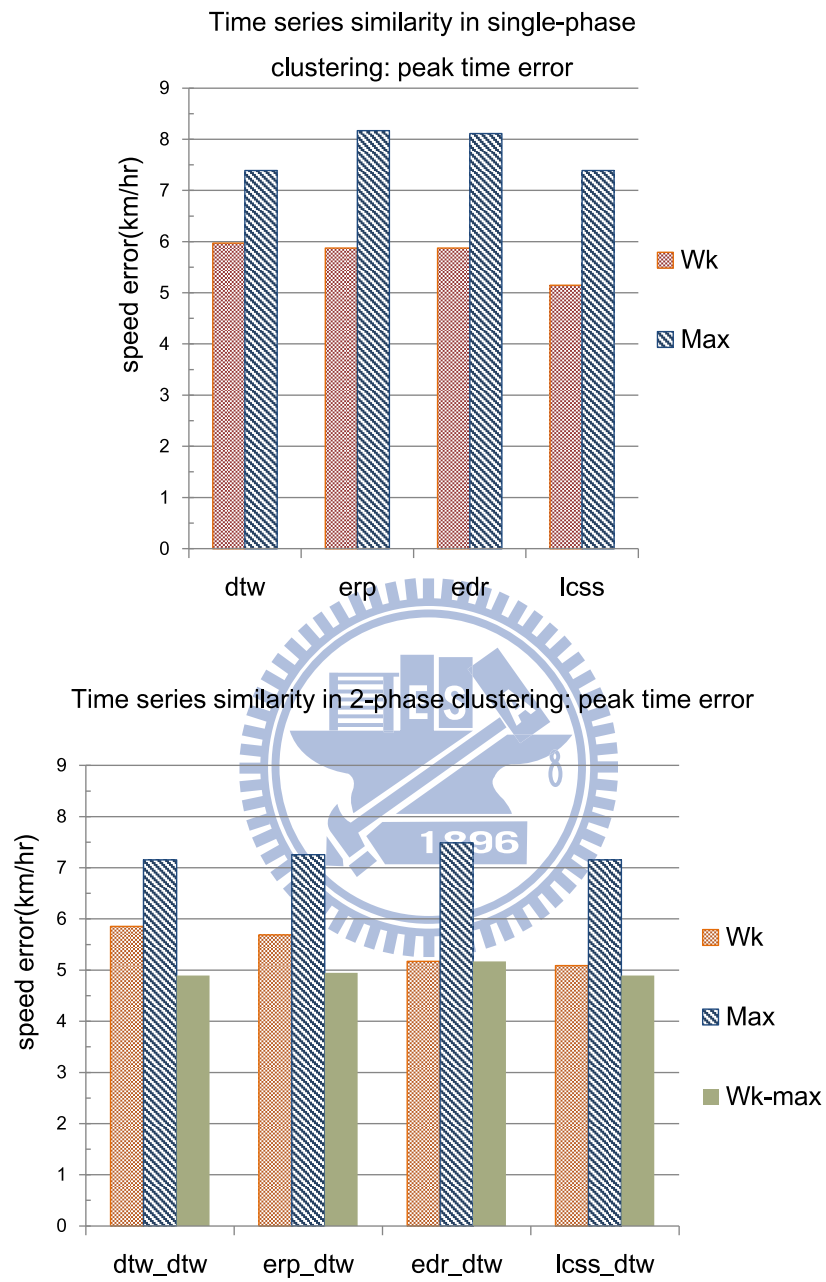
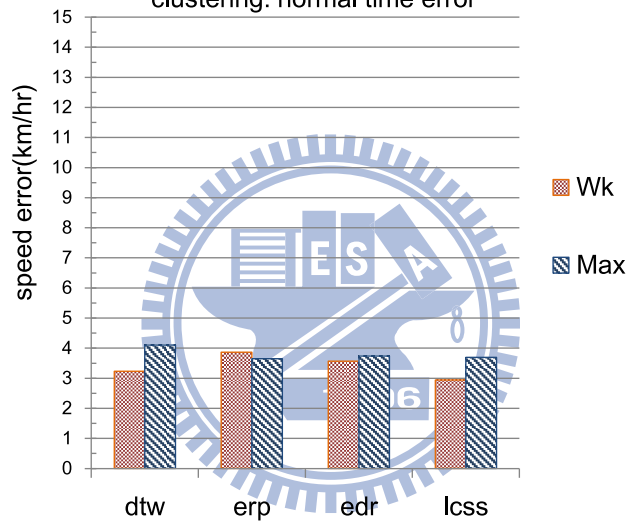


Figure 7.4: Best case of four time series similarities, Chubei-to-Hsinchu data set

Time series similarity in single-phase
clustering: normal time error



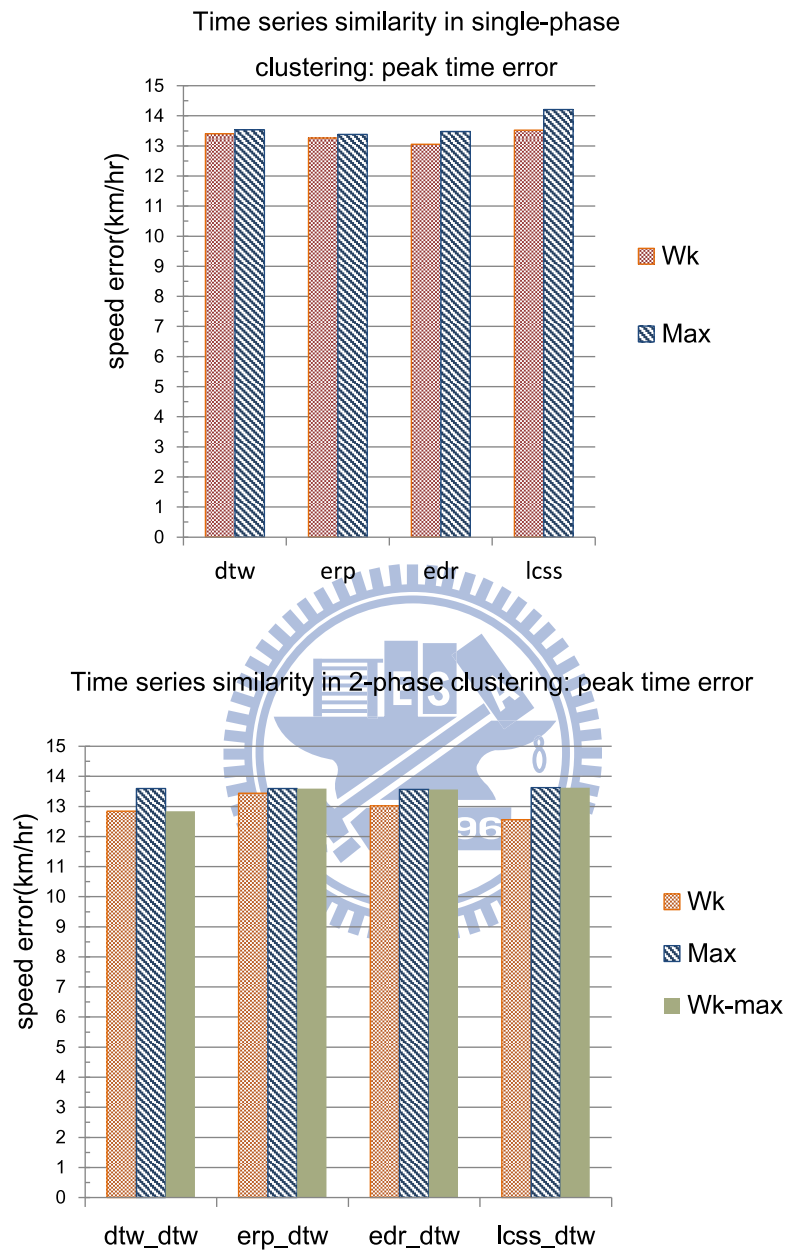


Figure 7.5: Best case of four time series similarities, Hsinchu-to-Chubei data set

Chapter 8

Conclusion

We aim to explore the speed patterns from historical traffic data of one road segment. Once the speed patterns are figured out, given a query time point or query time interval of the road, we can predict the speed values by referencing the speed patterns. In this paper, we have proposed a *two-phase clustering* method to mine the speed patterns given a set of *time series data* with speed value. In each phase of clustering, the *similarity* of each pair of input time series data is measured. Afterwards, the clustering algorithm clusters the time series data according to the measured similarities that are transformed as *distances* between these sequences of time series data. However, we found that the *peak time* of one speed pattern tends to have more complicated sub-patterns within one general all-day speed pattern. Hence, the proposed two-phase clustering method is designed to estimate the speed patterns on a *macroscopic scale* in the first phase, and furthermore determines the peak-time patterns for each general pattern on a *microscopic scale*. Moreover, we evaluate four commonly-used time series similarity measuring methods: DTW, EDR, LCSS, and ERP, with three typical clustering methods: K-means, hierarchical clustering and DBSCAN, to find the most effective selection for exploring speed patterns. In the experiment section, our speed prediction method is evaluated by our three proposed prediction functions and with real data from sensors on freeway segments. The experiment results show that adopting LCSS as the first phase similarity measurement and DTW as the second phase with the DBSCAN clustering method is the best suggested selection. Our prediction method has accuracy of less than 3km/hr speed error during normal

time, and has an error rate of between 5km/hr and 13km/hr according to different complexities of traffic behavior on the target road.



Bibliography

- [1] Taiwan area national freeway bureau. <http://www.freeway.gov.tw/English/Default.aspx>.
- [2] Andrew and Harvey. Chapter 7 forecasting with unobserved components time series models. volume 1 of *Handbook of Economic Forecasting*, pages 327 – 412. Elsevier, 2006.
- [3] A. Bejan, R. Gibbens, D. Evans, A. Beresford, J. Bacon, and A. Friday. Statistical modelling and analysis of sparse bus probe data in urban areas. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1256 –1263, sept. 2010.
- [4] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, Seattle, Washington, July 1994.
- [5] P. J. Bickel, C. Chen, J. Kwon, J. Rice, E. V. Zwet, and P. Varaiya. Measuring traffic. *Statistical Science*, 22(4):581–597, November 2007.
- [6] G. Box, G. Jenkins, and G. Reinsel. *Time series analysis: forecasting and control*. Forecasting and Control Series. Prentice Hall, 1994.
- [7] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Syst. Appl.*, 36:6164–6173, April 2009.
- [8] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30, VLDB '04*, pages 792–803. VLDB Endowment, 2004.

- [9] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.
- [10] B. Ding, J. X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT '08, pages 205–216, New York, NY, USA, 2008. ACM.
- [11] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1:1542–1552, August 2008.
- [12] J. Durbin and S. Koopman. *Time series analysis by state space methods*. Oxford statistical science series. Oxford University Press, 2001.
- [13] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.
- [14] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 794–805. VLDB Endowment, 2007.
- [15] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen. Estimating arterial traffic conditions using sparse probe data. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 929–936, sept. 2010.
- [16] T. Idé and S. Kato. Travel-time prediction using gaussian process regression: A trajectory-based approach. In *SDM*, pages 1183–1194, 2009.
- [17] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 10–, Washington, DC, USA, 2006. IEEE Computer Society.

- [18] H.-P. Kriegel, M. Renz, M. Schubert, and A. Züfle. Statistical density prediction in traffic networks. In *SDM*, pages 692–703, 2008.
- [19] C.-C. Lee, Y.-H. Wu, and A. L. P. Chen. Continuous evaluation of fastest path queries on road networks. In *Proceedings of the 10th international conference on Advances in spatial and temporal databases, SSTD'07*, pages 20–37, Berlin, Heidelberg, 2007. Springer-Verlag.
- [20] C.-H. Lo, W.-C. Peng, C.-W. Chen, T.-Y. Lin, and C.-S. Lin. Carweb: A traffic data collection platform. In *Proceedings of the The Ninth International Conference on Mobile Data Management*, pages 221–222, Washington, DC, USA, 2008. IEEE Computer Society.
- [21] T. Nakata and J.-i. Takeuchi. Mining traffic data from probe-car system for travel time prediction. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04*, pages 817–822, New York, NY, USA, 2004. ACM.
- [22] Y. Tian, K. C. K. Lee, and W.-C. Lee. Monitoring minimum cost paths on road networks. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '09*, pages 217–226, New York, NY, USA, 2009. ACM.
- [23] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering, ICDE '02*, pages 673–, Washington, DC, USA, 2002. IEEE Computer Society.
- [24] L.-Y. Wei, W.-C. Peng, C.-S. Lin, and C.-H. Jung. Exploring spatio-temporal features for traffic estimation on road networks. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases, SSTD '09*, pages 399–404, Berlin, Heidelberg, 2009. Springer-Verlag.
- [25] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *Proceedings of the 5th international conference on Mobile systems, applications and services, MobiSys '07*, pages 220–232, New York, NY, USA, 2007. ACM.

- [26] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 316–324, New York, NY, USA, 2011. ACM.
- [27] C. Zhang, S. Sun, and G. Yu. A bayesian network approach to time series forecasting of short-term traffic flows. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 216 – 221, oct. 2004.

