State Space Reduction in Modeling Traffic Network Dynamics for Dynamic Routing under ITS

Mahyar Movahed Nejad, *Student Member, IEEE*, Lena Mashayekhy, *Student Member, IEEE*, Ali Taghavi, and Ratna Babu Chinnam

***Abstract*—Effective en route guidance for vehicles can play an important role in alleviating the negative impacts of ever-growing congestion. As network traffic conditions change due to recurrent and non-recurrent congestion, the optimal route can change, and updated directions should be given to the driver in real-time. However, the task of exploiting real-time traffic information for optimal routing is computationally challenging. On the other hand, simplistic schemes (e.g., assuming constant speeds for different network arcs across all hours of the day) lead only to poor travel time performance and driver dissatisfaction. Hence, there is need for compact yet effective representations of traffic network dynamics for supporting routing algorithms. In this paper, we propose two state space reduction approaches employing knowledge discovery and data mining (KDD) methods and mathematical programming (MP) to strike an effective balance between accuracy and state space reduction (i.e., compactness). In doing so, they exploit historical data from ITS systems. We demonstrate the performance of the proposed approaches using actual road network data from Southeast Michigan.**

I. Introduction

THE primary concern in intelligent transportation systems (ITS) is using real-time traffic information while a vehicle is en route. Effective en route guidance for vehicles can play an important role in alleviating the negative impacts of ever-growing congestion. Real-time traffic information can be used to develop effective re-routing policies in order to avoid or reduce the impact of congestion, promising reduced travel times as well as cost. The cost of congestion has been growing rapidly in the US, increasing from $63.1 billion in 2000 to $87.2 billion in 2007 [1]. While road transportation network capacity is not growing fast enough to cope with increasing demand [1], the quickly expanding ITS coverage in the US can be a key enabler for reducing or controlling traffic congestion.

Not only can congestion be recurrent, which develops due to high volume of traffic seen during peak commuting hours, but it also can be non-recurrent due to such factors as accidents, vehicle breakdowns, bad weather, work zones, lane closures, special events, etc. The location and severity of non-recurrent congestion is unpredictable. Therefore, getting informed about this type of congestion requires real-time traffic information.

Real-time and historical traffic information can be gathered from embedded sensors. These sensors in transportation networks are hardware devices that can continuously track speed, density, and other traffic information of vehicles passing through each lane. Embedded sensors provide a source of massive real-time and historical traffic data that can support the development of effective data mining algorithms to predict recurrent congestion in the network by time of day.

In addition to recurrent congestion, two major factors change the optimal route while a vehicle is en route:

- Non-recurrent congestion, which changes the state of the network
- Changes in the driver's route choice, e.g. avoiding passing through high crime areas at night

With these potential changes in the optimal route, updated directions should be given to the driver in real-time. To obtain this updated direction, it is critical to have compact yet reliable information about the "state of the network", while a vehicle is in-route [2, 3].

Current state-of-the-art dynamic routing algorithms are incapable of computing these updated directions in an acceptable time as the network size increases. This is particularly true for algorithms that attempt to account for the non-stationary and stochastic aspects of traffic network dynamics (fluctuations in traffic speeds/densities over time and/or explicit treatment of congestion states). Dynamic programming methods (both deterministic and stochastic) are prevalent and suffer from curse of dimensionality in dealing with the scale and complexity of transportation networks in urban areas and require unacceptable run times for computing routing policies and offering rerouting options once the vehicle is en route. Naïve policies that arbitrarily limit the degree of "look ahead" to few links ahead of the vehicle can on the other hand lead to inferior performance (of reducing travel times and/or cost).

The state space can be quite large when the size of the transportation network increases, and this makes all of the above-mentioned challenges more complicated. Using simplistic schemes such as adopting constant speeds for different arcs across all hours of the day lead only to poor

Manuscript received April 10, 2011.

M. M. Nejad, A. Taghavi, and R. B. Chinnam are with the Industrial and Systems Engineering Department, Wayne State University, Detroit, MI 48202 USA (email: mahyar@wayne.edu (corresponding author); taghavi@wayne.edu; r_chinnam@wayne.edu).

L. Mashayekhy is with the Computer Science Department, Wayne State University, Detroit, MI 48202 USA (e-mail: mlena@wayne.edu).

travel performance and driver dissatisfaction. On the other hand, wanting to capture traffic dynamics of every arc at a one minute resolution (being required/promoted by some recent dynamic routing methods that rely on ITS data), become extremely unwieldy when dealing with differences in traffic dynamics across days of the week, weekends, months, holidays, significant events, and uncontrollable factors such as weather. What is necessary is a compact yet effective representation of path and network state dynamics.

Vast majority of the current literature still revolves around deterministic routing models and are yet to demonstrate any resemblance of a practical real-world algorithm that can support the realities of current day transportation network dynamics. Overcoming these challenges mostly depends on effectiveness of algorithms to both extract valuable information from large scale transportation databases in a timely fashion and reducing the run time performance of effective routing algorithms [4].

A few studies have focused on reducing the "state space" of routing algorithms by finding unnecessary nodes/links and eliminating them. Kim *et al.* [4] proposed a two-step procedure for state space reduction leading to improvements in run time performance. In the first step, the procedure eliminates redundant links that would not be traversed by any optimal route. The second step uses a priori reduction on the state space by deleting unnecessary links as the vehicle passes through the network. In [5], a hierarchical routing algorithm is proposed to reduce the state space using a heuristic "node promotion" technique. This technique reduces the number of route computations in hierarchical routing algorithms and improves computational performance. Song and Wang [6] applied graph-based hierarchical community detection algorithms to retrieve a road network structure. Moreover, they proposed a hierarchical routing algorithm based on the graph model which could compute optimal routes for between-community node pairs on large-scale road networks.

Chen *et al.* [7] employed three data reduction algorithms to cut down the computing time, decrease the memory space, and speed up the train positioning. These algorithms provide a simpler representation of the train tracks by extracting a few data points from the large amount of GPS data points. Chabini and Yadappanavar [8] proposed a bit-stream representation for discrete-time dynamic data. They showed its positive impacts on storage. However, their proposed representation does not capture the behavior of the historical traffic data. In [9], the sets of traffic data were organized into four basic classes, and a classification algorithm was proposed to assign these sets into their classes, automatically. Jula *et al.* [10] used real-time and historical data to predict travel times on a link. They developed methodologies to estimate the arrival times at the nodes of a stochastic and dynamic network. However, knowledge discovery from traffic data and state space reduction by extracting network dynamics and compacting the time windows is still missing.

In this study, the authors focus on state space reduction in modeling traffic network dynamics. Since real-time data does not have information about look ahead dynamics of the network, using historical traffic data along with analyzing real-time traffic data can provide a practical prediction of the behavior of the look ahead network dynamics. To this end, we propose a Knowledge-Discovery and Data Mining (KDD) approach and a Mathematical Programing (MP) approach. We use the raw empirical real-time traffic data from a road network in Southeast Michigan to demonstrate the performance of the proposed methods. Throughout this paper, we use the terms "link" and "arc" interchangeably.

The rest of the study is organized as follows. Section II presents the proposed KDD approach. In section III, a mathematical model is developed for the MP approach. Experimental results and evaluation of both proposed approaches are presented in section IV through a case study on a Southeast Michigan road network. In section V, conclusions and future research directions are discussed.

II. Knowledge Discovery & Data Mining Approach

ITS data (such as the traffic speed and number of vehicles passing through the different links of the network) recorded on an ongoing basis yields large databases suitable for Knowledge Discovery & Data Mining (KDD). Given that most ITS systems collect traffic data at one minute resolution from their sensors, they yield for the entire network $24 \times 60 = 1{,}440$ network traffic state observation vectors each day. The size of the vector (for each minute) is the number of sensors monitoring the different links of the network. While there could be multiple sensors for each link and they often record multiple pieces of information (speed, density etc.), we aggregate the information from all sensors monitoring a link and rely only on traffic speed information throughout the rest of the paper (future work will try to exploit other information being collected by the ITS).

This section proposes a KDD approach to cluster the network traffic state vectors based on similarity. The KDD involves data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation, and knowledge presentation [11]. Throughout this paper, we use link and arc interchangeably. In addition, unless explicitly stated otherwise, the analysis is carried out by day of week (holidays and special event days can be handled separately). Meaning, we allow network traffic dynamics to change by day of week (seasonal fluctuations can be handled by regularly updating and limiting the learning datasets to reasonable time spans surrounding the month(s) of interest). The final output of the KDD approach is prediction of the traffic speed for every link in the network, by day of week, based on time windows derived from the clusters. There are two phases to this approach. The goal in the first phase is to identify the network links that experience congestion during any part of the day. Given the broader objective of developing a state space reduction method for routing algorithms, we exclude these stable links from further

consideration during the second phase. The goal in the second phase is to partition network traffic state vectors (excluding the stable links) into distinct but contiguous time partitions/windows, where the network traffic dynamics are similar within a partition but are different across adjacent partitions (two non-adjacent partitions can be part of the same cluster). We explain these two phases in the following subsections. The case study section presents the preprocessing step and the results from experimental study to evaluate the performance of the proposed KDD approach.

A. Phase-1: Identification of links experiencing congestion

It is typical for some links in the network not to experience any congestion during the day, yielding very "stable" traffic speeds all through the day; while in other links, congestion could occur during peak travel times and such. We propose a clustering algorithm to identify the stable links of the network.

The procedure involves applying a *k*-means clustering algorithm to traffic data for each link over the course of the day (e.g., Mondays). To find the appropriate number of clusters (*k*), we rely on the finite mixture model proposed in [12] for each link. Fig. 1 shows an illustrative example of results from applying this procedure for traffic data from a Monday for all 116 ITS links of the Southeast Michigan network. The traffic data used for clustering comes from several consecutive Mondays. While Fig. 1 (left) shows the preprocessed links speeds (mph) for all links across the full 24 hours of this Monday, Fig. 1 (right) reports the discretized clustered states for the same links after *k*-means clustering. It is apparent from Fig. 1 (right) that a significant number of the links do not experience more than a single state. In fact, 60 out of 116 links have one state. For the links with one cluster, their centroids show the steady speed for a whole day.

The state space reduction method need not consider links that are stable across the day. Hence, only links yielding more than one cluster are considered for state space reduction, the second phase of the KDD approach.

B. Phase 2: Segmentation of network traffic over the course of the day into partitions with similar traffic states

As stated earlier, the goal here is to partition network traffic state vectors (i.e., the one-minute network traffic state columns of Fig. 1 (left) excluding the stable links) into distinct but contiguous time partitions/windows, where the network traffic dynamics are similar within a partition but different across adjacent partitions (two non-adjacent partitions can be part of the same cluster). To achieve this, we once again employ the *k*-means clustering algorithm. The data points for clustering are made up of columns from Fig. 1 (right) after excluding the stable links. To find the appropriate number of clusters (*k*), we once again rely on the finite mixture model proposed in [12]. Each cluster includes similar network states with its representative centroid.

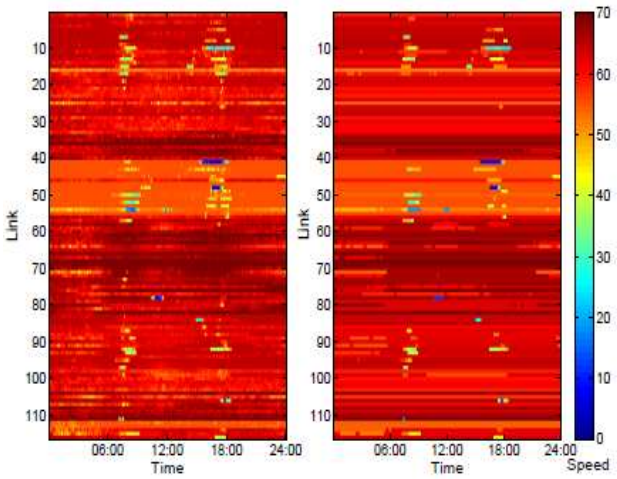


Fig. 1. Network traffic speeds (mph) for links from Southeast Michigan spanning all 116 arcs and all 24 hours of a Monday (10/18/2010). Left: Original preprocessed data. Right: Speeds after clustering.

Similar states consist of multiple time partitions/windows in which dynamics of the network are relatively the same. A centroid gives a deterministic prediction of speeds for all links in the network in its time partition. As a result, instead of having a database containing all 24×60 time windows for the day, phase two reduces the state space in modeling traffic network dynamics to *k* clustered states.

The output of phase two analysis for Monday (10/18/2010) is shown in Fig. 3 where we compare the results of two proposed approaches. While stable arcs are excluded from the input of phase two, we did not eliminate the stable arcs in Fig. 3 to maintain format consistency with Fig. 1. The arcs are also presented in the same order for ease of comparison.

III. MATHEMATICAL PROGRAMMING APPROACH

In the mathematical programming (MP) approach, we use the preprocessed ITS traffic data, explained in the following section, as an input. Similar to the proposed KDD approach of Section II, the desired output of MP approach is a reduced state space of the network. The following notation is used in the mathematical programming formulation:

TW	Set of time windows {1,…,T}, indexed by t, r
L	Set of links, indexed by l
M	Large number
$V_{l,t}$	Average observed velocity on link l at time t
$Q_{l,t}$	Binary indicator variable; $Q_{l,t} = \begin{cases} 1, & \text{if } \lvert V_{l,t+1} - V_{l,t} \rvert \geq \alpha \\ 0, & \text{otherwise} \end{cases}$
P_t	Integer variable, partition number for time window t

We once again seek to consolidate as many consecutive time windows as possible into larger partitions for the entire network in order to reduce the state space. However, we need to ensure the following: 1) The range of speeds within a partition for each link should be bounded (denoted δ and is user defined) and 2) There should not be any significant

abrupt changes in speed between two consecutive time windows within the same partition (threshold is denoted β and is user defined). The full mathematical formulation is as follows:

$$Min\ Z = P_T \tag{1}$$

s.t.

$$P_t \le P_{t+1} \qquad \forall t; t = 1\ to\ T-1 \tag{2}$$

$$|V_{l,t+1} - V_{l,t}| \le \alpha + M * Q_{l,t} \qquad \forall l \in L,\ \forall t; t = 1\ to\ T-1 \tag{3}$$

$$|V_{l,t+1} - V_{l,t}| \ge \alpha - M * (1 - Q_{l,t}) \qquad \forall l \in L,\ \forall t; t = 1\ to\ T-1 \tag{4}$$

$$\sum_{l \in L} |V_{l,t+1} - V_{l,t}|.Q_{l,t} - \beta \sum_{l \in L} Q_{l,t} \le M * (P_{t+1} - P_t) \qquad \forall t; t = 1\ to\ T-1 \tag{5}$$

$$|V_{l,r} - V_{l,t}| \le \delta + M * (P_r - P_t) \qquad \forall l \in L,\ \forall t, r; t, r = 1\ to\ T-1;\ r \ge t \tag{6}$$

$$P_t \ge 0, P_1 = 1 \qquad \forall t \in 1\ to\ T \tag{7}$$

Constraint (2) requires the partition numbers to be assigned to all time windows in a non-descending order. Constraints (3)-(4) determine the value of the binary variable $Q_{l,t}$. In other words, they determine the links that differ in speed between two consecutive time windows by more than a threshold α. Constraint (5) requires that the average of differences in speed for those links not to exceed β, from time window t and time window t+1. Constraint (5) is the linearized form of the following statement:

$$P_t = P_{t+1} \Rightarrow \frac{\sum_{l \in L} |V_{l,t+1} - V_{l,t}|.Q_{l,t}}{\sum_{l \in L} Q_{l,t}} \le \beta \tag{8}$$

We introduce constraints (3)-(5) to ensure that the average differences between speeds of all consecutive time windows with difference more than α do not exceed β in the same partition. And finally, equation (6) ensures that the range of speeds within a partition for each link is less than δ. Since the output of MP approach relatively depends on the threshold parameters (α, β, δ), we present the sensitivity analysis on these parameters in section IV.

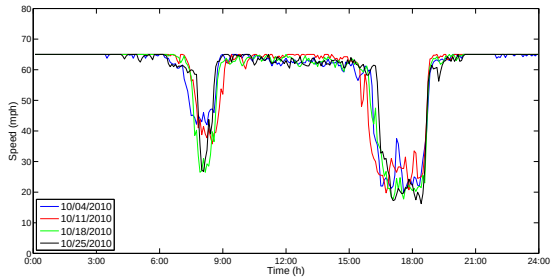
The output for the mathematical programming approach is the estimated speed for every link in all optimally partitioned time windows.

IV. SOUTHEAST MICHIGAN CASE STUDY

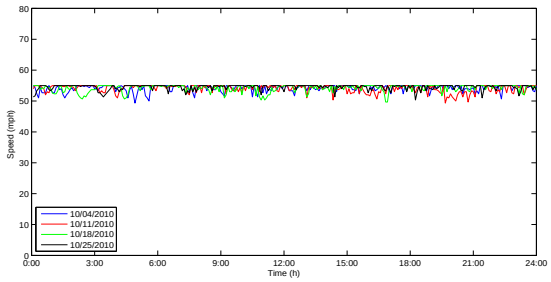
We use real traffic data from a road network in Southeast Michigan. We explain preprocessing step first. Then, we present the experimental results for both the proposed KDD approach and the MP approach.

A. Traffic data preprocessing

1) Data integration: In the data integration step, multiple data sources may be combined. In our study, real-time traffic data from Michigan Intelligent Transportation System (MITS) and Traffic.com for a road network in Southeast Michigan are integrated. The case study road network covers major freeways and highways in and around the Detroit metropolitan area for the month of October 2010.



(a)



(b)

Fig. 2. Traffic speed data from four consecutive Mondays for two particular arcs in Southeast-Michigan. (a) Arc with congestion (b) Arc without congestion.

The network has 116 observed links. The raw speed data was aggregated at a resolution of 5-minute intervals, yielding 60/5×24×31 (minutes/5 × hours × days) data points for each link of the network.

2) Data selection: In the traffic data selection step, it is vitally important to note the date and time of the study. For example, there is a significant difference in traffic flow from rush hour in the morning to midnight. One should also consider seasonal effects depending on the area of selection. We select the data from a whole month (October 2010) to encompass the changes between different time windows of days and the differences between days (weekdays and weekends) while making sure the data is not considerably distorted by seasonal effect.

3) Data cleaning: In the data cleaning step, noise and inconsistent data are removed from the traffic database. For example, some technical problems might occur with a sensor, and it reports some impractical data (e.g., an unrealistic number of vehicles passing through a lane in a minute). In addition, in case of a defective sensor for a specific lane in a link, we eliminate that sensor's data and consider the remaining sensors of that link.

4) Data transformation: In the data transformation step, data is transformed or consolidated into forms appropriate for traffic data mining. Since the traffic flow of any link depends on all of its lanes, we set the average speed of the different lanes as the speed of the link. This leads to reducing the reported speeds in one day from 123,408 to 35,217.

B. Preliminary analysis of network traffic data

Review of ITS network data does confirm that traffic speed patterns over the course of the day tend to be quite

similar for individual arcs given a day of the week (barring long-term shifts due to seasonality and incidents). For example, Fig. 2 shows the similarity traffic speed signatures for two particular arcs over four consecutive Mondays in October 2010. Fig. 2(a) shows an arc experiencing both morning and afternoon rush hour congestion while Fig. 2(b) shows an arc with stable traffic speeds all through day for the selected Mondays.

C. Results from formal experiments

To test the efficiency of the proposed methods, we setup a series of experiments.

First set of experiments apply the proposed state space reduction methods on data from a particular day, e.g. Monday (Sunday), and test the effectiveness of the resulting partitions on other Mondays (Sundays) from adjacent weeks. The performance measure here is root-mean-square-error (RMSE) in mph, calculated by estimating the differences between actual speeds recorded and the speeds estimated for the partitions by the proposed methods and aggregated for the entire network over the full day.

Table I reports the results from applying the MP, with parameter setting of $\alpha = 3,\ \beta = 6, \text{and}\ \delta = 15,$ to data from a particular Monday (10/04/2010) and then testing the resulting partitions on three future Mondays. The process is also replicated for Sundays and the results are also reported in Table I. The low RMSEs (< 2mph) and their consistency across baseline days and corresponding future days do confirm that the proposed methodology holds good promise for traffic data modeling and forecasting. It is good to see that the RMSE for the fourth Monday (Sunday) is less than the RMSE of the baseline itself, suggesting that the baseline partitions are quite robust. Given stochastic variability in traffic conditions, we naturally expect some RMSE fluctuations from day to day, and hence, testing RMSE being lower than training data RMSE is nothing unusual.

Fig. 3 reports the output of applying KDD approach (left) and MP approach (right) to network data from Monday (10/18/2010). The vertical lines on the plots identify the resulting partitions. The plots also reveal the estimated link speeds for different partitions at different times of the day. The plots clearly reveal that MP approach has a tendency to produce more partitions in comparison to KDD approach.

Although the performance of the MP approach is dependent on the parameter settings and different settings will lead to different results, all parameter settings tried for the MP approach in Table III produce more partitions than the KDD approach.

Second set of experiments not only apply the partitions resulting from the proposed methods but also exploit the resulting partition speed estimates for forecasting speeds from corresponding future days. The following sections report results from applying both the KDD approach and the MP approach and also compare them.

1) KDD approach: In analyzing the baseline Monday 10/4/2010 data, the finite mixture model identified the optimal number of clusters (k) to be 7, which produced 19 partitions for the whole day. Table II reports the results achieved from testing the KDD approach corresponding future Mondays. Once again, the process is replicated for Sundays and the results are also reported in Table II. Two measures are reported. The first measure (RMSE) shows root mean square error between predicted speed (mph) of Mondays (Sundays) and original speed (mph) of the baseline Monday (Sunday). The second measure (Diff) shows the percentage of predicted speeds which do differ more than 7.5 mph from the actual reported speed.

2) Mathematical programming approach: Since the output of MP approach relatively depends on the threshold parameters (α, β, δ), we first start with a sensitivity analysis on the parameters. Fig. 4 reports the impact of changes in α, β, δ on the number of partitions created by the MP

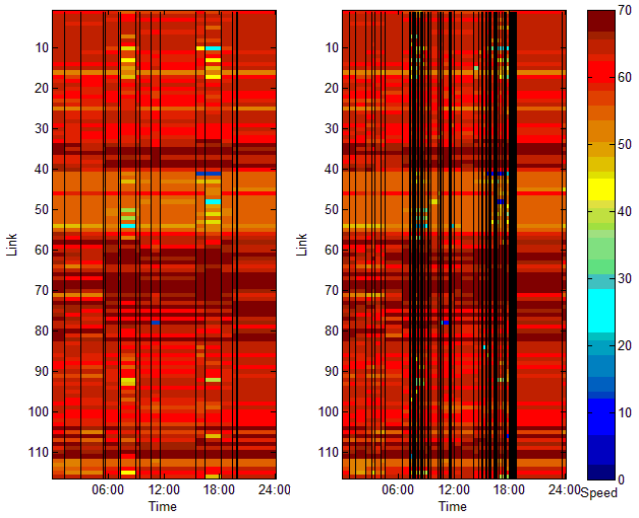


Fig. 3. Output from applying KDD approach (left) and MP approach (right) to data from a Monday (10/18/2010). Vertical lines denote partitions. Colors denote the estimated link speeds (mph) after state space reduction.

TABLE I
PERFORMANCE STABILITY OF MP PARTITIONS FOR FORECASTING

Experiment	Mondays	RMSE (mph)
1 (baseline)	10/04/2010	1.43
2	10/11/2010	1.51
3	10/18/2010	1.73
4	10/25/2010	1.41
Experiment	**Sundays**	**RMSE (mph)**
1 (baseline)	10/03/2010	1.33
2	10/10/2010	1.95
3	10/17/2010	1.31
4	10/24/2010	1.32

TABLE II
PERFORMANCE OF KDD APPROACH

Experiment	Predicted date	Baseline	RMSE	Diff >7.5mph
1 Sunday	10/03/2010	10/03/2010	1.29	0.019
2 Sunday	10/10/2010		4.19	0.03
3 Sunday	10/17/2010		3.52	0.028
4 Sunday	10/24/2010		3.94	0.041
5 Monday	10/04/2010	10/04/2010	1.38	0.024
6 Monday	10/11/2010		4.56	0.046
7 Monday	10/18/2010		4.63	0.049
8 Monday	10/25/2010		4.62	0.053

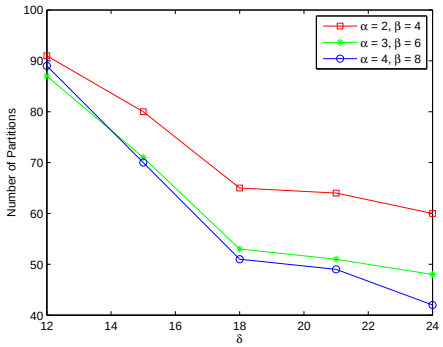


Fig. 4. Effects of changes in δ on number of MP partitions.

TABLE III
IMPACT OF THRESHOLD PARAMETERS ON MP APPROACH PERFORMANCE

Thresholds			Monday 10/18/2010		Sunday 10/03/2010	
α	β	δ	# of partitions	RMSE (mph)	# of partitions	RMSE (mph)
1	2	12	160	0.604	91	0.849
		15	158	0.748	85	0.947
		18	157	0.767	83	0.993
		21	157	0.767	83	0.993
		24	157	0.767	82	1.011
2	4	12	91	1.146	46	1.286
		15	80	1.511	30	1.697
		18	65	1.951	21	2.350
		21	64	2.021	18	2.601
		24	60	2.110	16	2.637
3	6	12	87	1.196	46	1.281
		15	**71**	**1.661**	**32**	**1.665**
		18	53	2.152	22	2.168
		21	51	2.233	19	2.423
		24	48	2.341	19	2.510
4	8	12	89	1.183	50	1.295
		15	70	1.690	36	1.623
		18	51	2.192	28	2.153
		21	49	2.298	26	2.394
		24	42	2.496	25	2.474

TABLE IV
PERFORMANCE OF MP APPROACH ($\alpha = 3, \beta = 6, \delta$=15)

Experiment	Predicted date	Baseline	RMSE	Diff> $\frac{\delta}{2}$: 7.5 mph
1 Sunday	10/03/2010	10/3/201	1.33	0.025
2 Sunday	10/10/2010		4.26	0.032
3 Sunday	10/17/2010		3.61	0.030
4 Sunday	10/24/2010		4.02	0.046
5 Monday	10/04/2010	10/4/201	1.43	0.024
6 Monday	10/11/2010		4.81	0.055
7 Monday	10/18/2010		4.91	0.058
8 Monday	10/25/2010		4.91	0.064

approach. As we increase α, β,or δ, i.e., increasing the feasible region, the number of partitions decrease in a somewhat non-linear fashion. Table III reports sensitivity analysis results when changing the threshold parameters.

Table IV reports more detailed results regarding the performance of the MP approach when the threshold parameters are set as follows: $\alpha = 3$, $\beta = 6$, and δ=15.

V. CONCLUSION

Transportation networks are becoming more congested, in particular, in urban areas. Fortunately, ITS systems and their coverage are growing in the US and other parts of the world to provide drivers with increasingly accurate real-time data regarding traffic conditions. In support of dynamic routing algorithms, we proposed two approaches for modeling traffic network dynamics. The primarily goals are compact representation and accurate estimation of speeds. The methods are distinct and rely on KDD techniques as well as formal optimization techniques based on mathematical programming. Results from testing the proposed methods on actual road network data from Southeast Michigan are very promising.

Future work will focus on the development of routing algorithms that exploit the results from the proposed methods.

REFERENCES

[1] D. Schrank and T. Lomax, "2009 urban mobility report, " *Texas Transp. Inst.*, Texas A&M Univ. Syst., College Station, TX, 2009.

[2] I. Chabini and S. Lan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, pp. 60-74, 2002.

[3] M. M. Nejad and L. Mashayekhy, "Compact Representation of Traffic Network Dynamics Using an Efficient Knowledge Based Discovery Approach," in *Proc. Industrial Engineering Research Conference*, Reno, NV, May 2011.

[4] S. Kim, M. E. Lewis, and C. C. White III, "State space reduction for nonstationary stochastic shortest path problems with real-time traffic information," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 6, pp. 273-284, 2005.

[5] G. Jagadeesh, T. Srikanthan, and K. Quek, "Heuristic techniques for accelerating hierarchical routing on road networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, pp. 301-309, 2002.

[6] Q. Song and X. Wang, "Efficient Routing on Large Road Networks Using Hierarchical Communities," *Intelligent Transportation Systems, IEEE Transactions on*, pp. 1-9.

[7] D. Chen, Y. S. Fu, B. Cai, and Y. X. Yuan, "Modeling and Algorithms of GPS Data Reduction for the Qinghai–Tibet Railway," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, pp. 753-758, 2010.

[8] I. Chabini and V. Yadappanavar, "Advances in discrete-time dynamic data representation with applications to intelligent transportation systems," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1771, pp. 209-218, 2001.

[9] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg, "Different methods of traffic forecast based on real data," *European Journal of Operational Research*, vol. 155, pp. 558-568, 2004.

[10] H. Jula, M. Dessouky, and P. A. Ioannou, "Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, pp. 97-110, 2008.

[11] J. Han and M. Kamber, *Data mining: concepts and techniques*: Morgan Kaufmann, 2006.

[12] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on pattern analysis and machine intelligence*, pp. 381-396, 2002.