Interpolating Sparse GPS Measurements Via Relaxation Labeling and Belief Propagation for the Redeployment of Ambulances

Andrew Phan and Frank P. Ferrie, *Member, IEEE*

***Abstract*—One major challenge for traffic management systems is the inference of traffic flow in regions of the network for which there are little data. In this paper, Global-Positioning-System (GPS)-based vehicle locator data from a fleet of 40–60 roving ambulances are used to predict the most likely ambulance speeds in a network of 20 000 streets in the city of Ottawa, ON, Canada. First, the road network is represented as a directed graph data structure. Then, we compare two algorithms, i.e., relaxation labeling and belief propagation, that interpolate the sparse and noisy measurements from the fleet to obtain dense locally consistent ambulance speeds. Unlike several other systems in the literature, we model all of the city's freeways and surface streets, and both road types are treated with equal importance. Furthermore, the data structure and algorithms described in this paper are not only extended to real-world needs such as road closures and the incorporation of live data with historical data but are also computationally efficient, providing updates in intervals of less than 5 min on commodity hardware. Presented experimental results address the key issue of validating the performance and reliability of the system.**

***Index Terms*—Belief propagation, interpolation, Global Positioning System, graph theory.**

I. Introduction

MAINTAINING the state of a complex road network, given limited sensor input, is a major challenge to the design of modern traffic control systems. In this paper, we address the specific problem of inferring most likely ambulance speeds along each of 20 000 streets in the greater city of Ottawa, ON, Canada, from Global Positioning System (GPS)-based vehicle locator data supplied by a fleet of 40–60 roving ambulances. The context for this paper is a system, jointly developed under the project heading RISER,[1] for managing the redeployment of a fleet of ambulances operated by the Ottawa Paramedic Service (OPS).

Manuscript received July 20, 2010; revised June 13, 2011; accepted August 8, 2011. Date of publication September 15, 2011; date of current version December 5, 2011. The Associate Editor for this paper was J. A. Miller.

The authors are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 2A7, Canada (e-mail: aphan2@cim.mcgill.ca; ferrie@cim.mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2011.2165281

[1]RISER stands for Rapid Intelligent Scheduling for Emergency Responders. It is a joint project led by CAE Inc. under the Precarn Inc. CORE program, with the participation of Actenum Inc., McGill University, Simon Fraser University, and the OPS.

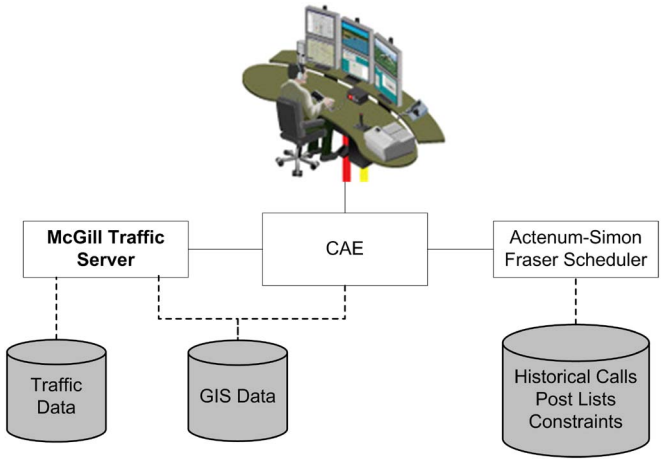

Fig. 1. RISER system overview.

The focus of this paper is on the McGill Traffic Server (TS), as depicted in Fig. 1, whose task is to provide an estimate of the most likely ambulance speed along each street in the road network using a combination of sparse GPS data and a Geographic Information System (GIS) database. The key idea is that, given prior information in the form of date/time indexed historical data, we can reliably interpolate a set of sparse real-time measurements and obtain reasonable estimates over the extent of the network. This approach is achieved through an interpolation algorithm that operates on a representation of the traffic network as a directed graph and serves to integrate information from different sources. Here, we consider only vehicle locator data and user-input speeds, but the algorithm is sufficiently general to incorporate any available source of information, provided that the following two conditions are satisfied: 1) It is georeferenced to a particular location in the network, and 2) it can be expressed in terms of relative congestion (described shortly). The particular interpolation algorithms that we investigate are relaxation labeling (RL) [1] and belief propagation (BP) [2], which have widely been used in computer vision [3]–[8].

This paper is organized as follows. Section I-A reviews the state of the art. Section I-B outlines the contributions of this paper. Section II describes the solution that we propose. Section III details the experimental methodology and the validation results. Section IV concludes this paper with a discussion and suggestions for future work.

1524-9050/$26.00 © 2011 IEEE

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

A. Related Work

In the literature, there exist a number of systems that model traffic flow. FreeSim [9]–[11] models freeways as a directed graph data structure and implements a multitude of algorithms for the fastest path calculation. Corsim [12], [13], which is sponsored by the U. S. Federal Highway Administration, consists of the following two traffic models that have undergone extensive validation: 1) Netsim for modeling surface streets and 2) Fresim for modeling freeways. Vatsim [14], [15] uses a graph that is composed of nodes and segments to implement a vehicle and traffic simulator for the evaluation of vehicle and traffic control methods. Similarly, Mitsim [16] models the road network using nodes, links, segments and lanes to simulate individual vehicle movements for the evaluation of traffic management systems. Transims [17] is a large-area long-time-horizon transportation planning tool that can be parallelized [18] for better performance and is designed to predict population trends and displacements of individual households and travelers. Paramics [19], [20] is another parallel traffic simulator that provides road network planners with a range of tools for the detailed modeling of complex traffic flows. Vissim [21] models detailed traffic flow for the design of traffic-actuated control systems. Renaissance [22] uses a traffic state estimator based on an extended Kalman filter to obtain real-time traffic estimates of freeways. CellSIM [23] discretizes roads into small intervals or cells and uses a measure of occupancy to model high volume of traffic at the regional level.

We observe that, despite the prevalent use of the graph data structure [9], [14], [16], [24], [25] to model the road network topology, the RL and BP graph algorithms have seen little or no attention in traffic modeling. To the best of our knowledge, only BP has received some recent recognition [26]; however, the authors did not have access to real-world data to validate their model. With this paper, we hope to remediate the situation and demonstrate the usefulness of RL and BP, if nothing else than as a baseline, based on extensive real-world experimental results.

Another surprising observation that we note are the very few attempts that deal with the sparse spatiotemporal sensor data issue. However, this issue, because the number of road segments is currently orders of magnitude greater than the number of sensors, is commonplace and thus unavoidable. On small scales, microscopic traffic flow models that consider the behavior of individual vehicles such as [27] can extrapolate estimates into areas with few or no sensors. However, for large scales, we are only familiar with the translation of smoothing indices in [28] and of the workaround in [29] to generate synthetic speed data to prevent the average speed from monotonically decreasing when no new data are available. RL and BP provide another way of interpolating the sparse data on a city-wide scale such that the final result is locally consistent at all points in the network.

B. Contribution

The main contribution of this paper includes two algorithms (and their evaluation) for interpolating a sparse set of sensor measurements across an entire road network in a computationally efficient manner. In particular, we show how traffic congestion can be predicted on a city-wide scale from sparse automatic vehicle locator data to a reasonable degree of accuracy. Both algorithms convert sparse and locally inconsistent data using constraints as defined by the road network topology (e.g., linkages, number of lanes, and speed limits), historical data, and user inputs (e.g., road closures and accidents) into dense and locally consistent estimates. Because the framework is sufficiently general, it can accommodate arbitrary sensor information, provided that sensor data can be georeferenced to a particular road segment and expressed in terms of relative congestion (described shortly). On commodity hardware, our system can predict road congestion for the entire city of Ottawa, including both freeways and surface streets, in less than 5 min.

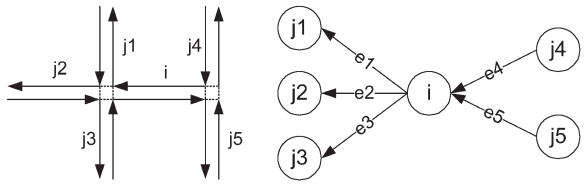

Fig. 2. Simplified road network (left) and the corresponding graph representation from the perspective of node i (right). Nodes j^1–j^5 are neighbors of i and nodes e^1–e^5 are the corresponding edges that connect i with its neighbors.

II. Proposed Solution

In the literature, microscopic models are mostly used, because they can better handle complicated road geometries and features such as traffic lights or on-ramp metering [12]. However, a macroscopic traffic model has the advantage when it comes to large-scale designs due to the reduced number of variables [30]. Because we hope to estimate potential ambulance speeds for every road segment in a city, our approach takes on a macroscopic flavor to keep our first attempt at a large-scale implementation as simple as possible.

The TS takes any available road congestion data and outputs a potential ambulance speed for every road segment in a road network. To accomplish this task, we first represent the road network as a directed graph where nodes correspond to road segments and edges correspond to linkages. Then, we obtain initial estimates for each node using the available historical, live, and user data. Finally, we interpolate the graph using RL or BP.

A. Represent the Road Network as a Graph

The first step is to convert the road network into a graph representation that will serve as a basis for subsequent computations, with nodes being road segments and edges being linkages. The TS accepts any road network database, as long as it is given in the widely used geospatial vector data Environmental Systems Research Institute (ERSI) shapefile format [31]. We use the Shapefile C Library available online [32] to extract the street geometry and attributes and use the Boost Graph C++ library available online [33] to conveniently represent the road network as a directed graph composed of nodes and edges, as shown in Fig. 2.

1) Nodes: Each node in the graph corresponds to a single road segment with a unique direction. As such, a two-way street gives rise to two nodes or road segments whose directions are

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

180° apart, whereas a one-way street results in a single node. In Fig. 2, nodes j^1–j^5 are neighbors of node i. To indicate the direction of vehicle flow, we add a prefix "in" or "out" as a qualifier and refer to neighbors as in-neighbors or out-neighbors. In Fig. 2, nodes j^1–j^3 are out-neighbors of node i, whereas nodes j^4 and j^5 are its in-neighbors.

2) Edges: Nodes are linked to each other through edges. Because we have a directed graph, each edge has a particular direction. To indicate the origin and destination of an edge, we refer to the originating node as a source and the destination node as a target. To indicate the position of an edge relative to a node, we add a prefix "in" or "out" as a qualifier and refer to edges as in-edges or out-edges. In Fig. 2, node i has three out-edges e^1–e^3 and two in-edges e^4 and e^5.

Each edge has a weight, bound between 0 and 1, that defines the likelihood of vehicle trajectories. In principle, edge weights should vary according to the time and date and be estimated using historical and live turn ratios. Because we do not have access to such information, we compute static edge weights using a generalized assumption about the behavior of vehicles at intersections. Explicitly, we assume that vehicles tend to travel in the path with the least resistance or, conversely, the path with the most conductance. In general, the out-edge with the greatest weight has the smallest deviation angle and the target road segment with the greatest number of lanes and highest speed limit. Note, however, that this assumption does not always hold, because at times, on-ramps or off-ramps are favored over main arteries. Fortunately, these cases, although predictable, are isolated, and this assumption allows us to form a simpler baseline estimate that will serve to evaluate the model's performance as more information is eventually included. In Fig. 2, we may refer to the weight of edge e^1 using one of two interchangeable notations, i.e., ω_{e^1} or $\omega_{i\to j^1}$, where i is the source of e^1, and j^1 is its target.

Numerically, we calculate edge weights as follows. If a node i has n out-edges $\{e^1,\ldots,e^n\}$ whose targets are out-neighbors $\{j^1,\ldots,j^n\}$, respectively, then the weight of edge e^k for $k\in\{1,\ldots,n\}$ is defined as

$$\omega_{e^k}=\omega_{i\to j^k}=\frac{G_{e^k}}{\sum_{m=1}^{n}G_{e^m}} \tag{1}$$

where the conductance of edge e^k is

$$G_{e^k}=\sigma_{j^k}\cdot e^{-\beta\cdot\theta_{e^k}} \tag{2}$$

where the road segment capacity or conductivity of j^k is

$$\sigma_{j^k}=L_{j^k}\cdot(S_{j^k}+\epsilon) \tag{3}$$

where L_{j^k} and S_{j^k} are the number of lanes and the speed limit of j^k, respectively, and

$$\beta=\frac{-\ln 0.1}{90}. \tag{4}$$

Note that (3) is a virtual road segment capacity and not the actual road segment capacity, because the actual road segment capacity is not known. Note also that θ_{e^k} is the angle between vectors formed by road segments i and j^k, ϵ is an empirical value that compensates for the difference between the average ambulance speed and the speed limit, and β is a constant that is empirically obtained such that a greater bend in the road results in a smaller conductance value. In this case, we make it so that a deviation angle of 90° results in a conductance of 10% of the road capacity. Note that we assume symmetric β such that left and right turns have equal weight. Finally, out-edges whose targets are dead ends have a weight of 0, because we assume that the number of vehicles that enter a dead-end road segment is too small to impact the congestion of neighboring roads.

B. Incorporate Available Road Congestion Data

The next step, after representing the road network as a directed graph, is to obtain an initial estimate for each node in the graph using whatever road congestion data are available. To accomplish this task, we first register the data to a particular node. Then, for each node, we construct a speed profile that is essentially a trend in ambulance speeds over time. Finally, we combine historical trends with live and user data to obtain the desired initial estimates, which we express as relative congestion.

1) Register the Road Congestion Data to Specific Nodes: The only road congestion data that we have incorporated at this time, aside from user-input speeds, are the Automatic Vehicle Location (AVL) GPS data provided by the OPS. In particular, we have access to almost a year's worth (February 1–December 20, 2007) of historical ambulance AVL data, resulting in approximately 18 million AVL samples. Because the AVL data only give the positions of ambulances in intervals of 10–12 s on the average, we use three consecutive positions in time to reliably estimate and register the ambulance's direction and speed to a specific node. The ambulance speed is estimated using a straight-line approximation by computing the distance traveled and dividing this value by the time interval. Then, we register the speed sample to the correct node by choosing the node that shares the closest position, orientation, and (known) out-neighbor. For time intervals on the order of 10–12 s, this straight-line approximation provides a satisfactory baseline, although there is room for improvement by using a more sophisticated road-following algorithm.

Unfortunately, the AVL data do not currently distinguish deployment speeds with lights or sirens on from redeployment speeds with lights and sirens off. Hopefully, the information will eventually be made available so that we can separately consider the two classes of data. Although it is possible to employ outlier detection or pattern recognition techniques to classify the data, for now, we maintain our goal of obtaining a baseline estimate and assume that all AVL speeds are redeployment speeds.

2) Construct Historical Speed Profiles: Next, we construct a historical speed profile for each node in the graph, which is essentially a trend in ambulance speeds over time. We start by separating the registered AVL data into the following two classes: 1) weekdays and 2) weekends. Then, we evaluate the following two approaches for speed profile construction: 1) linear least squares regression (LS) [34] and 2) bin averaging (BA).

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

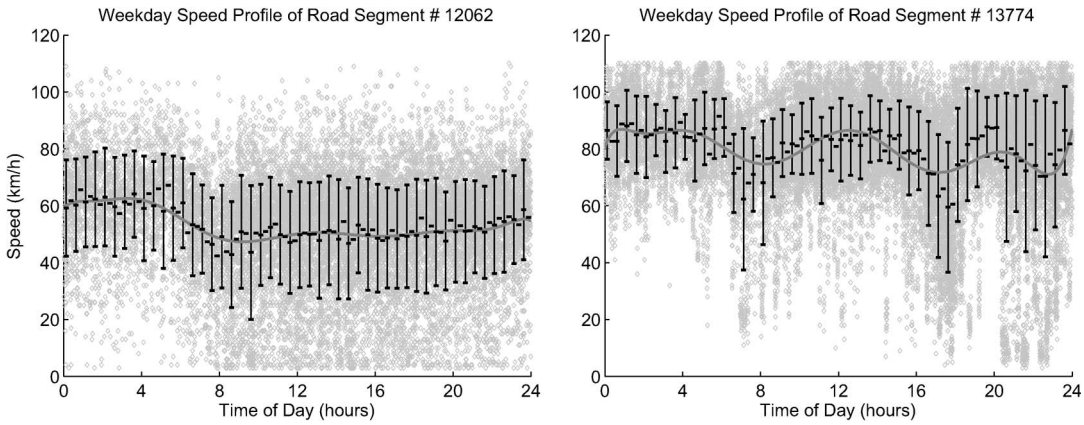

Fig. 3. Weekday speed profiles for two different road segments with abundant samples. Light gray points show the registered AVL speed samples. The dark gray smooth continuous line is the LS fit, whereas the black dashes near the regression fit are the means of each 15-min bin. The STD bars show the extent of the uncertainty in the data.

In LS, the goal is to estimate the coefficients $\{c_0, \ldots, c_p\}$ of a polynomial fit to the data, i.e.,

$$s(t) = c_0 t^0 + c_1 t^1 + c_2 t^2 + \cdots + c_p t^p \tag{5}$$

where t and s are the normalized time of day and the registered ambulance speed, respectively, of the sample data. To reduce the chance of overfitting, we perform K-fold cross validation for $K = 10$ and choose the degree p, bound between 1 and 10, that results in the smallest prediction error. The second method, BA, does not attempt to fit a curve to the data. Instead, we divide a 24-h day into 15-min bins and take the mean for each bin to obtain the speed profile. Fig. 3 shows the weekday speed profiles of two different road segments with ample AVL samples using both methods.

Finally, we complete the historical speed profile by obtaining a confidence value that is a function of the number and variance of samples available. For example, a large number of samples and small variance results in a very high confidence value.

3) Incorporate Live Data: The initial road congestion estimate for each node depends on the available data and their confidence values. The confidence values of the live data are a weighted average between historical and live confidence values, with more weight on live confidence values. As such, if there are only historical but no live data, then the initial estimate previously obtained remains unchanged. If there are both historical and live sensor data, then the initial estimate is a weighted average, with less weight given to the more uncertain (lower confidence) of the two. If there are only live data, then the initial estimate is directly obtained. Finally, if a node does not have access to any data, then its initial estimate remains unknown. We store all live data, because they are added to the historical data and are used to periodically update the historical speed profiles.

4) Incorporate User Data: The TS is designed to require as little user data or intervention as possible. Only when current events start to greatly differ from the historical data is it necessary for the user to input any changes. Fortunately, the TS is locally adaptable, and the following two types of user overrides are currently supported: 1) road closures and 2) speed overrides. As such, a user may accomplish one of the following two actions: 1) Close off entire road segments such that affected vehicles are forced to find alternate routes, or 2) manually input a speed and confidence value as if it were a live sensor.

5) Express Initial Estimates in Terms of Relative Congestions: We address the problem of merging different sensors by converting arbitrary sensor measurements into a relative congestion term R that is bound between 0 and 1 using an equation that may be unique for each sensor modality. For the processed AVL and the user-input speed data, we use

$$R(t) = 1 - \frac{s(t)}{S} \tag{6}$$

where $s(t)$ is the initial speed estimate, and S is the speed limit. However, for other sensors such as inductive loops, the function $R(t)$ might directly be obtained from counts rather than from vehicle speeds.

Unfortunately, the relative congestions that serve as the initial estimates are spatiotemporally sparse and noisy for a number of reasons. The initial estimates are sparse (i.e., not all nodes have an initial estimate) because of the following two cases: 1) There are fewer than 60 moving ambulances that provide data at any time for a road network that consists of 20 000 streets, and 2) ambulances drive over some roads more often than other vehicles such that some road segments may have tens of thousands of AVL samples, whereas other road segments have none. Furthermore, the initial estimates are noisy, because as shown in Fig. 3, there is a great deal of variance in the AVL data. Both the sparsity and the variance of the data contribute to the likelihood that relative congestions of neighboring road segments are contradictory. The next section addresses these two concerns by interpolating the initial estimates such that we obtain dense estimates that are locally consistent.

C. Interpolate the Graph

We now explain how we can interpolate the sparse initial estimates previously obtained and resolve any local inconsistencies. In our previous work [35], we only looked at the RL algorithm. Now, we reexamine RL in more detail, consider the additional interpolation algorithm BP, and compare the two algorithms. The underlying goal is to use the road network

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

topology to determine how likely each road segment is congested based on how likely its neighboring road segments are congested. Both algorithms, as described in the following sections, allow for the splitting of a complex computation into many simple and parallel computations, use context to compensate for noise and ambiguity, are deterministic, are relatively simple to implement, and offer a baseline to which we can compare eventual more complex strategies.

1) RL: The RL algorithm [1] involves incrementally and iteratively adjusting the likelihoods of a set of labels for each node according to local constraints and local evidence. Formally, each node in the graph has a set of m labels λ, where each label corresponds to a possible relative congestion value. In turn, each label has a weight, which is bound between 0 and 1, that defines its likelihood. The notation that we use is $p_i(\lambda)$ to mean the weight of label λ for node i. The initial distribution of label weights depends on the initial relative congestion estimate previously obtained and our confidence in its value. During the RL iterative process, we incrementally change the label weights until convergence is reached. The most likely relative congestion corresponds to the label with the greatest weight.

A compatibility function defines the relationship between labels of neighboring nodes and can be of the first order or of a higher order type. The notation that we use for first-order compatibilities is $r_{ij}(\lambda, \lambda')$ to mean the compatibility between label λ of node i and label λ' of node j. If the two labels greatly support each other, then $r_{ij}(\lambda, \lambda')$ should be large and positive. If the labels greatly inhibit each other, then $r_{ij}(\lambda, \lambda')$ should be large and negative. Otherwise, if there is no relation between the two labels, then $r_{ij}(\lambda, \lambda')$ should be 0. In practice, node i has multiple neighbors, and the notation for higher order compatibilities becomes $r_{ij^1j^2\cdots j^n}(\lambda, \lambda^1, \lambda^2, \ldots, \lambda^n)$. Note that the compatibility function can be any monotonically decreasing function that decreases as the difference between neighboring labels increases.

Numerically, there are a number of ways of computing the compatibilities. In this paper, we experiment with a compatibility function that is applicable to both first-order and higher order compatibilities, which we refer to as the relative difference model (RDM). The RDM compatibility function between label λ of node i and labels $\lambda^1, \lambda^2, \ldots, \lambda^n$ of in- or out-neighbors $j^1, j^2, \ldots, j^n$ is defined as

$$r_{ij^1j^2\cdots j^n}(\lambda, \lambda^1, \lambda^2, \ldots, \lambda^n) = \prod_{k=1}^{n} \omega_k \cdot \exp\left(-\alpha \cdot \frac{|\lambda^k - \lambda|}{m}\right) \tag{7}$$

where

$$\omega_k = \begin{cases} 1, & \text{for higher order compatibilities} \\ \omega_{i \to j^k}, & \text{for first-order compatibilities due to out-neighbors} \\ \frac{\omega_{j^k \to i} \cdot \sigma_{j^k}}{\sum_{h=1}^{n} (\omega_{j^h \to i} \cdot \sigma_{j^h})}, & \text{for first-order compatibilities due to in-neighbors.} \end{cases} \tag{8}$$

In addition, σ is the road conductivity, m is the number of labels, and α is an exponential decay constant.

The main steps of the RL algorithm are detailed as follows.

RL 1: Initialize the label weights of each node according to the simulation or current date and time. If the initial estimate of node i is unknown, then the label weights $p_i(\lambda)$ will have uniform distribution such that all labels are equally likely. Otherwise, we initialize the distribution of $p_i(\lambda)$ using a Gaussian distribution whose mean is the initial estimate previously obtained and whose variance is determined by the number of samples and the variance in speed profile. For example, a large number of samples and small variance should result in a more peaked distribution.

RL 2: Compute the label support of each node. The support for a given label of a given node depends on the label weights of neighboring nodes and the compatibility function. For example, the support for λ, assuming first-order compatibility $r_{ij}(\lambda, \lambda')$ if node i with labels λ has neighbor j with labels λ', can be expressed as the following sum over all labels λ':

$$q_i(\lambda) = \sum_{\lambda'} r_{ij}(\lambda, \lambda') p_j(\lambda'). \tag{9}$$

Because node i can have multiple in- and out-neighbors $j^1, j^2, \ldots, j^n$ with labels $\lambda^1, \lambda^2, \ldots, \lambda^n$, respectively, where n is often 2–6, the support using higher order compatibilities becomes

$$q_i(\lambda) = \sum_{\lambda^1} \sum_{\lambda^2} \cdots \sum_{\lambda^n} r_{ij^1j^2\cdots j^n}(\lambda, \lambda^1, \lambda^2, \ldots, \lambda^n) \cdot p_{j^1}(\lambda^1) \cdot p_{j^2}(\lambda^2) \cdot \cdots \cdot p_{j^n}(\lambda^n). \tag{10}$$

However, for performance reasons, we separately consider the higher order compatibilities of in- and out-neighbors using the following approximation:

$$q_i(\lambda) = q_i^{\text{in}}(\lambda) \cdot q_i^{\text{out}}(\lambda) \tag{11}$$

where

$$q_i^{\text{in}}(\lambda) = \sum_{\lambda_{\text{in}}^1} \sum_{\lambda_{\text{in}}^2} \cdots \sum_{\lambda_{\text{in}}^y} r_{ij_{\text{in}}^1 j_{\text{in}}^2 \cdots j_{\text{in}}^y}\left(\lambda, \lambda_{\text{in}}^1, \lambda_{\text{in}}^2, \ldots, \lambda_{\text{in}}^y\right) \cdot p_{j_{\text{in}}^1}\left(\lambda_{\text{in}}^1\right) \cdot p_{j_{\text{in}}^2}\left(\lambda_{\text{in}}^2\right) \cdots p_{j_{\text{in}}^y}\left(\lambda_{\text{in}}^y\right) \tag{12}$$

$$q_i^{\text{out}}(\lambda) = \sum_{\lambda_{\text{out}}^1} \sum_{\lambda_{\text{out}}^2} \cdots \sum_{\lambda_{\text{out}}^z} r_{ij_{\text{out}}^1 j_{\text{out}}^2 \cdots j_{\text{out}}^z}\left(\lambda, \lambda_{\text{out}}^1, \lambda_{\text{out}}^2, \ldots, \lambda_{\text{out}}^z\right) \cdot p_{j_{\text{out}}^1}\left(\lambda_{\text{out}}^1\right) \cdot p_{j_{\text{out}}^2}\left(\lambda_{\text{out}}^2\right) \cdots p_{j_{\text{out}}^z}\left(\lambda_{\text{out}}^z\right) \tag{13}$$

and y and z are the number of in- and out-neighbors, respectively. Because, in general, nodes have three in-neighbors and three out-neighbors, the aforementioned approximation effectively reduces the complexity and the number of computations required from $O(n^6)$ to $O(2n^3)$.

RL 3: Compute the update direction for the label weights $\bar{u}^k$, as described in [1, App. A].

RL 4: Check the stop criteria (i.e., the maximum number of iterations reached or update direction is negligible); otherwise, update label weights $\bar{p}^{k+1} = \bar{p}^k + h\bar{u}^k$, where

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

$0 < h \leq \alpha_k$, and α_k is a small-valued maximum step size that may decrease as k increases to speed convergence.

RL 5: Obtain the final labeling and go back to RL 1 to resume the entire process for the next moment in time. When the RL process ends, each node has a distribution of label weights that is locally consistent with its neighboring nodes according to the local constraints that we defined. The label with the highest weight corresponds to the most likely relative congestion, which we convert back to ambulance speed using

$$s = S \cdot (1 - R). \tag{14}$$

The entire RL algorithm starts over again with the new date and time to keep the road network up to date with the latest speed estimates.

2) BP: The BP algorithm [2] involves incrementally and iteratively adjusting the messages that enter each node in the graph according to local constraints and local evidence until convergence is reached. In BP, each node in the graph has a set of m states, where each state x_i at node i corresponds to a possible relative congestion value. In turn, each state has a weight, bound between 0 and 1, that defines its likelihood. The state with the greatest weight indicates the most likely relative congestion. The notation that we use is $b_i(x_i)$ to mean the weight or belief of state x_i for a node i.

Beliefs for each node sum to 1 and are a function of a local evidence vector $\phi_i(x_i)$, messages $m_{ij}(x_j)$, and compatibilities $\psi_{ij}(x_i, x_j)$. $\phi_i(x_i)$ denotes the likelihood of each state according to available road congestion data. The distribution of $\phi_i(x_i)$ is determined at the start and does not change throughout the BP update cycle. $m_{ij}(x_j)$ denotes how likely node i thinks node j is in state x_j. The messages that enter each node are initialized using a uniform distribution and, during the BP iterative process, are incrementally updated according to update rules until convergence is reached. $\psi_{ij}(x_i, x_j)$ denotes the relationship between states x_i at node i and states x_j at j. Compared with RL, BP only supports first-order compatibilities and is therefore simpler and faster to compute.

The main steps of the BP algorithm are detailed as follows.

BP 1: Initialize the local evidence of each node according to the simulation or current date and time and the BP messages to a neutral state using a uniform distribution. The initialization of the local evidence $\phi_i(x_i)$ is similar to the initialization of the label weights in RL 1.

BP 2: Update all BP messages until convergence or until the maximum number of iterations is reached. Messages are updated using the following update rule:

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \left(\phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i) \backslash j} m_{ki}(x_i) \right) \tag{15}$$

where first-order compatibilities $\psi_{ij}(x_i, x_j)$ are obtained using the weighted RDM (i.e., $\omega_k \neq 1$), as described in (7). Note that $N(i) \backslash j$ refers to all the neighbors of node i other than j. Note also that each edge in the graph has two BP messages, i.e., one message from the source to the target and another message from the target to the source, because nodes are influenced as much by their in-neighbors as by their out-neighbors.

BP 3: Obtain the final beliefs and resume the entire process for the next moment in time. After updating the BP messages in the previous step, we can compute the beliefs for each node. The belief $b_i(x_i)$ over the possible states x_i at a node i is given by

$$b_i(x_i) = k\phi_i(x_i) \prod_{j \in N(i)} m_{ji}(x_i) \tag{16}$$

where k is a normalization constant such that beliefs sum up to 1, $\phi_i(x_i)$ is the local evidence at i, $N(i)$ refers to the neighbors of i, and $m_{ji}(x_i)$ corresponds to all the messages that come into i from neighbors j. The state with the highest weight corresponds to the most likely relative congestion, which we convert to a potential ambulance speed using (14). Similar to RL, the entire BP algorithm starts over again with the new date and time to keep the road network up to date with the latest speeds.

For both RL and BP, the TS outputs the most likely ambulance speed for every road segment in the road network at any given date and time.

III. Experimental Methodology and Results

A. Background

Before we present our results, we reiterate that all experimental results were obtained using real GPS data samples and a real GIS database. As such, for this first attempt at obtaining a baseline estimate, we briefly describe several simplifying assumptions that we made. Because the GIS road network database is incomplete, we assumed that all streets in the road network are bidirectional (two way) and each road segment is two lanes wide. Furthermore, due to errors in the AVL data, we ignore ambulance speeds above 130 km/h, scale the speeds down by a factor of 11/13, and set a minimum ambulance speed of 10 km/h such that ambulance speeds are bounded between 10 and 110 km/h.

B. Stratified K-Fold Cross Validation

Evaluating the TS in a timely manner was particularly challenging due to the large size of the road network and the millions of data samples involved. To this end, we developed a thorough and automated process using a variant of the classical K-fold cross validation [36] called the stratified K-fold cross validation. This approach first involves randomly splitting the historical data into K evenly sized parts called folds. Then, we choose one of the folds to act as the test set and train the model on the remaining $K - 1$ folds. We repeat this process K times using each fold only once as the test set and always training on the remaining folds. As such, each iteration randomly excludes a percentage of the data from the data set for initialization of the graph and then uses these excluded data to test the resulting interpolation estimates. Then, the estimates are merged by computing the mean of all K runs. The advantage of this technique is that all data samples contribute to the training and testing of the system. Furthermore, doing so allows us to gauge

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

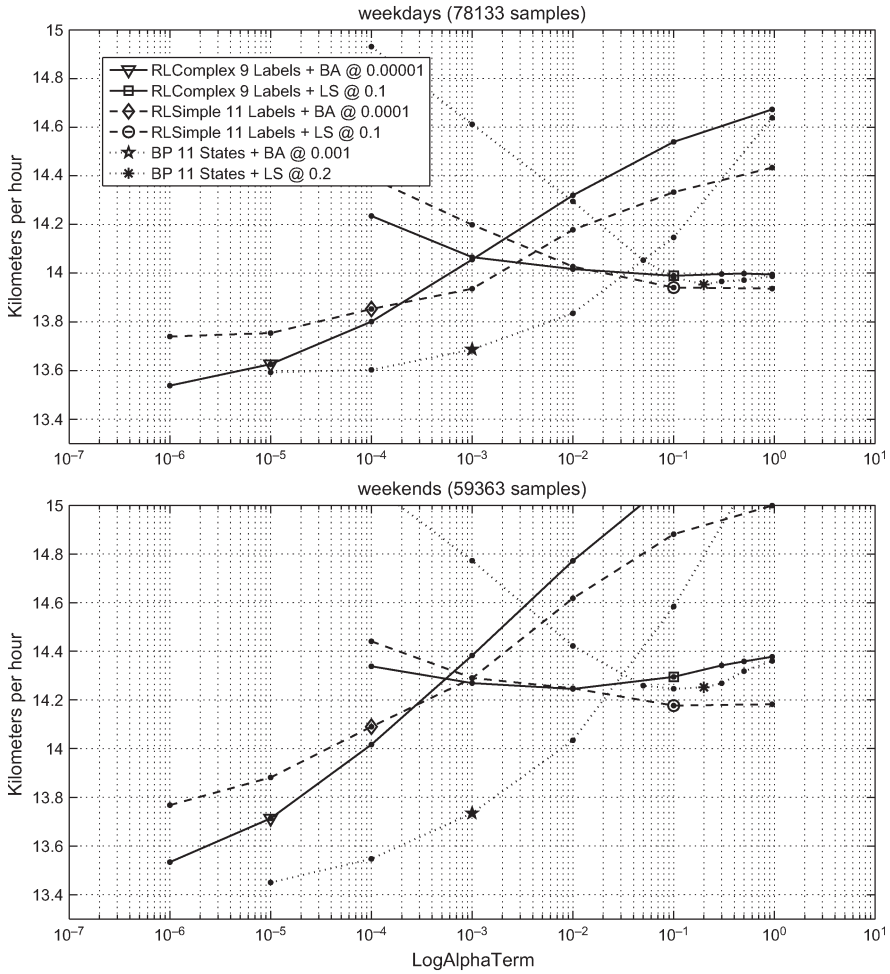

Fig. 4. Sensitivity analysis for α showing MAEs for road segments with known speed limits (40–100 km/h) and without historical speed profiles, where $\alpha = -\ln(\text{LogAlphaTerm})$. Markers identify curves and indicate the best LogAlphaTerm values for which MAEs and STDs are minimal. STDs are omitted from the plot to reduce clutter.

the stability of the algorithm with respect to variations in the input data. Typically, five folds are enough to conclude whether the model's performance is stable. The only difference between the classical and the stratified approach is that, in the stratified approach, each fold maintains the same ratio between classes as the original data set. For example, if 20% of the samples in the original data set occurred on the weekend, then the same percentage of weekend samples would be maintained in each fold after the random splitting of the data into K folds.

Due to the large number of historical data samples (approximately 2.5 million samples per fold, assuming five folds) and the processing time required to update the model (up to 5 min per update for RL with nine labels and higher order compatibilities), it is not feasible to interpolate the graph for every sample in the test set, because it would take 5×2.5 million $\times$ 5 min of processing time. Instead, we let the TS run for a continuous span of 24 h per fold per data class, interpolating the road network in intervals of 15 min and storing, for each road segment, the most likely interpolated ambulance speed estimate for the given time and day. Then, each sample in the test set is matched to the nearest sample in time to the stored interpolated estimates, and the difference between the two ambulance speeds is used as the estimate error. This approach has the significant advantage that the validation time is the same, regardless of the size of the test set. In particular, using $K = 5$ folds, two data classes (weekdays and weekends), 5 min per update, and a 24-h-day discretization interval of 15 min, the processing time required is reduced to $5 \times 2 \times 5 \div 15 = 3.33$ days to validate the system for a given set of parameters.

C. Experimental Results

We experimented with a number of configurations. For the initialization of the graph, we compared LS and BA. For the interpolation of the graph, we tried different numbers of states for BP and labels for RL, ranging from four to 21. In addition, for RL, we investigated first-order (RL Simple) and higher order (RL Complex) compatibilities. Finally, to gauge the effectiveness of the interpolation algorithm, we separately considered nodes with historical data from nodes without historical data.

We start by presenting in Fig. 4 the results of a sensitivity analysis for the exponential decay constant α defined in (7), where $\alpha = -\ln(\text{LogAlphaTerm})$. The purpose of this analysis is to obtain the optimal value for α, which results in the best

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

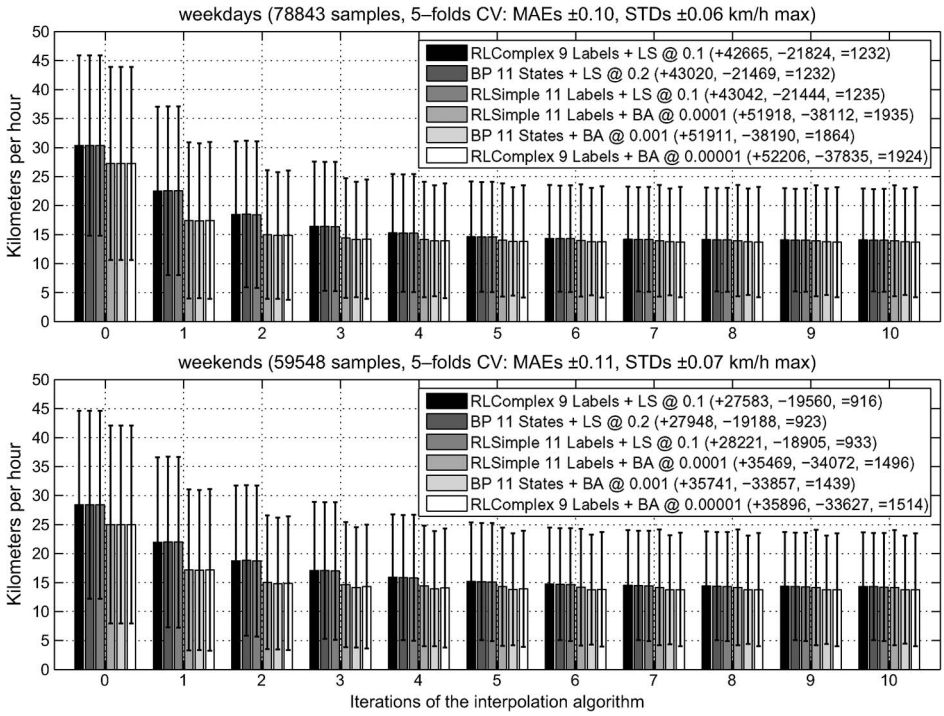

Fig. 5. MAEs and STDs for road segments without historical speed profiles and with known speed limits (40–100 km/h). In the legend, numbers in parentheses indicate the number of times that we overestimated $(+)$, underestimated $(-)$, and correctly estimated $(=)$ the most likely ambulance speed.

ambulance speed prediction. Note that we obtain six curves, because we wish to compare the two types of historical speed profiles (BA versus LS) and the three types of interpolation strategies (BP versus RL Simple versus RL Complex). To avoid cluttering the figure, we only show 11 states/labels for BP and RL Simple, because higher values did not measurably improve the estimates. Furthermore, for RL Complex, we limited the number of labels to nine, because higher values took longer than 5 min of processing time to update the model. Finally, note that the marker locations in the figure not only help identify the particular curve but also show the best values for α, for which the mean absolute errors (MAEs) and standard deviations (STDs) are the smallest. For example, for RL Complex 9 Labels + BA, α that is smaller than $-\ln(10^{-5})$ resulted in lower MAEs (good) but larger STDs (bad). To reduce clutter, STDs are not plotted.

Given the ideal values for α, we computed the MAEs and STDs for each of the six interpolation strategies and obtain Fig. 5 for road segments without historical data and Fig. 6 for road segments with historical data. At iteration 0, the MAEs and STDs observed are large, because the errors are due to using the posted speed limit $+\,\epsilon$ defined in (3) as the predicted ambulance speed.

In Fig. 5, for road segments without historical speed profiles, we observe that all six strategies perform well, in which RL Complex 9 Labels + BA has a slight edge in terms of lower MAE, but BP 11 States + BA performs better in terms of lower STD. Although the difference is marginal, we also observe that BA appears to be the better choice for obtaining the historical speed profiles, because it consistently outperforms LS. Finally, we note that all six interpolation strategies tend to overestimate ambulance speeds, particularly on weekdays, because we sometimes observe more than twice the number of overestimated speeds than underestimated speeds. As a result, there is a higher possibility that the errors could add up and that travel times could significantly be underestimated. On weekends, the three strategies with the lowest MAEs fared much better in terms of the errors being more even. As a result, the errors have a higher possibility of canceling each other out when estimating the travel times. Surprisingly, the difference between MAEs for weekdays and weekends is negligible, and we observe only a slight increase in STDs for weekends, although there are almost three times fewer samples available for weekends than for weekdays (600 000 versus 1.7 million samples obtained by adding the total samples in Figs. 5 and 6).

In Fig. 6, for road segments with historical speed profiles, we observe no measurable improvement in terms of MAE and a very slight reduction in STD for the two strategies with the lowest MAEs. For the remaining four strategies, we observe that the MAEs tend to worsen (increase) with each iteration. This case is a direct result of the interpolation process that essentially oversmooths the data to achieve local consistency across all the nodes in the graph. We conclude that we are currently unable to improve upon the AVL data, except to slightly reduce their variance.

For road segments without historical speed profiles, we show how our MAEs and STDs vary according to the posted speed limits in Fig. 7 and to five periods of day in Fig. 8. In Fig. 7, we observe that BA significantly outperforms LS, except for the 40 km/h limit, where LS has a marginal edge. In addition, as expected, the errors gradually increase with the increase in the posted speed limit, because the range over possible ambulance speeds is greater. Finally, we observe that RL Complex has an edge over BP and RL Simple, because it is often first or second

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

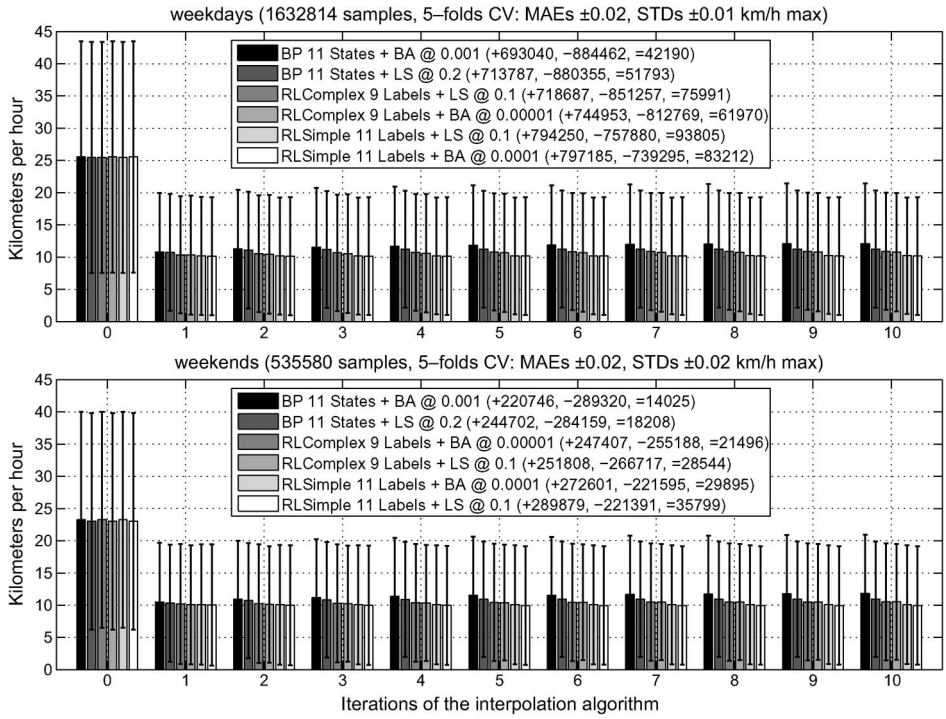

Fig. 6. MAEs and STDs for road segments with historical speed profiles and with known speed limits (40–100 km/h). In the legend, numbers in parentheses indicate the number of times that we overestimated $(+)$, underestimated $(-)$, and correctly estimated $(=)$ the most likely ambulance speed.

in obtaining the lowest MAEs and STDs. In Fig. 8, we observe that RL Complex consistently gives the lowest MAE, regardless of the period of day. In addition, we note that the TS does not favor a period of day, because it performs the same throughout the day.

Finally, we note that our estimates appear to be very stable, despite variations in the input data. Across the five folds, we observe very small variations in the MAEs and STDs. The largest variations were observed in Fig. 7 for road segments without historical speed profiles on weekdays for posted speed limits of 70 km/h, where the MAEs and STDs varied by a maximum of 0.92 and 0.55km/h, respectively, across the five cross-validation folds.

IV. Discussion

A. Validation Results

The MAEs and STDs that we obtained may seem large at first, but we believe that they are remarkably good and consistent. One reason that the errors may seem large is due to the great deal of variance in the AVL data, although we compensated by using the variance and the number of samples in the AVL data to initialize the graph. Another reason is the road network database of Ottawa, which lacks critical information and for which we had to make several assumptions. The results that we obtained can, hopefully, only get better using the current implementation as a baseline. Overall, given the sparse nature of the measurements, the results obtained were much better than expected and more than sufficient to estimate travel times between any two nodes in the network.

B. Scale and Performance

The TS was designed with the issue of scale in mind. Indeed, there is no limit on the number of AVL samples in the historical data, because the initial one-time heavy number crunching is done offline, and the runtime calculations are negligible. On commodity hardware, the TS can update the state of the entire road network, including both freeways and surface streets in less than 5 min. For much larger cities, we can take advantage of the parallel nature of interpolation algorithms and split the processing over multiple cores using readily available multicore processors to further minimize the processing time.

C. Portability and Flexibility

The TS is sufficiently flexible such that it can be ported to any arbitrary city with very little user modifications. The only requirement is that the road network database of the new city follows the same widely used ESRI Shapefile format [31] and contains the required attributes such as street name and speed limit. If necessary, a user can modify the road network (i.e., road closures) or update its status (i.e., override historical data with new speed data) without rendering the model useless. This approach allows the system to be more dynamic and to more closely model actual road conditions.

D. Future Work

There remains much work to do in terms of improving the accuracy of the TS, and we now list possible areas of improvement. In terms of performance, we need to take advantage of the parallel nature of interpolation algorithms to minimize the

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

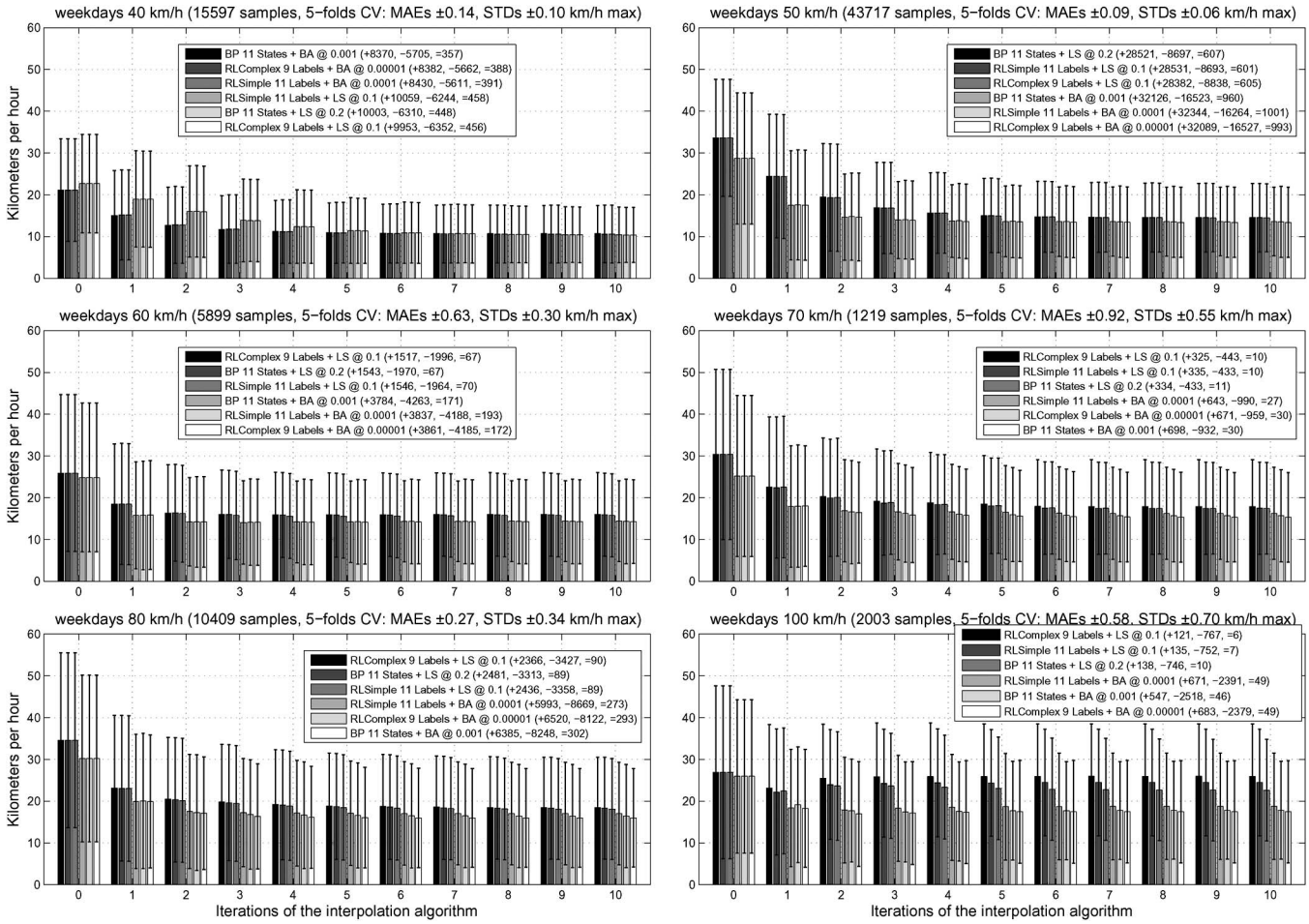

Fig. 7. MAEs and STDs for road segments without historical speed profiles by posted speed limits. In the legend, numbers in parentheses indicate the number of times we overestimated $(+)$, underestimated $(-)$, and correctly estimated $(=)$ the most likely ambulance speed.

processing time. The speedup would be significant, and cities that are significantly larger than Ottawa could be supported. To find optimal values for parameters such as determining β in the calculation of edge weights for left and right turns, we need to investigate statistical approaches such as simulated annealing to find the global minima and avoid falling into a local minima. With regard to edge weights, we need to incorporate statistics at intersections such as turn ratios to obtain dynamic rather than static edge weights. In addition, we need to learn how we can aggregate individual nodes into supernodes, which we believe will result in estimates that are more robust to noise, because each supernode would have access to more historical data. With regard to the AVL data, more sophisticated data-clustering techniques can be employed to help segment the data into additional classes aside from weekdays and weekends. Furthermore, a road-following algorithm rather than our straight-line approximation would improve speed estimation from GPS positions. We also need to distinguish ambulance deployment data with sirens/lights on from redeployment data with sirens/lights off. We need to incorporate data from additional GPS devices on other fleets of vehicles such as taxis, delivery trucks, or police cruisers. In addition, we need to incorporate data from traffic cameras or inductive loops, because this approach can provide useful information about the state of the road network. With respect to the GIS data, we need one/two-way street information and the number of lanes to improve our compatibility function. Finally, as a practical system, more experiments are needed to verify that the TS can successfully be ported to other cities. For example, what are the road network size limits and data sparseness limits? How sparse is too sparse?

V. Conclusion

We have represented a road network as a directed graph and used two graphical interpolation algorithms to obtain dense locally consistent road speed estimates from sparse and noisy sensor data. We have found that the algorithms are surprisingly good at compensating for the sparse and noisy sensor data by using context to resolve ambiguities. Furthermore, the validation results were remarkably consistent and are very stable, despite variations in the input sensor data.

Our main contribution has been the extensive validation of RL and BP for predicting ambulance redeployment speeds. We used real GPS data and a real GIS database to interpolate the sparse and noisy sensor data. Our approach can accommodate arbitrary data sources, as long as they can be georeferenced to a particular location in the road network and be represented in terms of relative congestion. It incorporates confidence

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

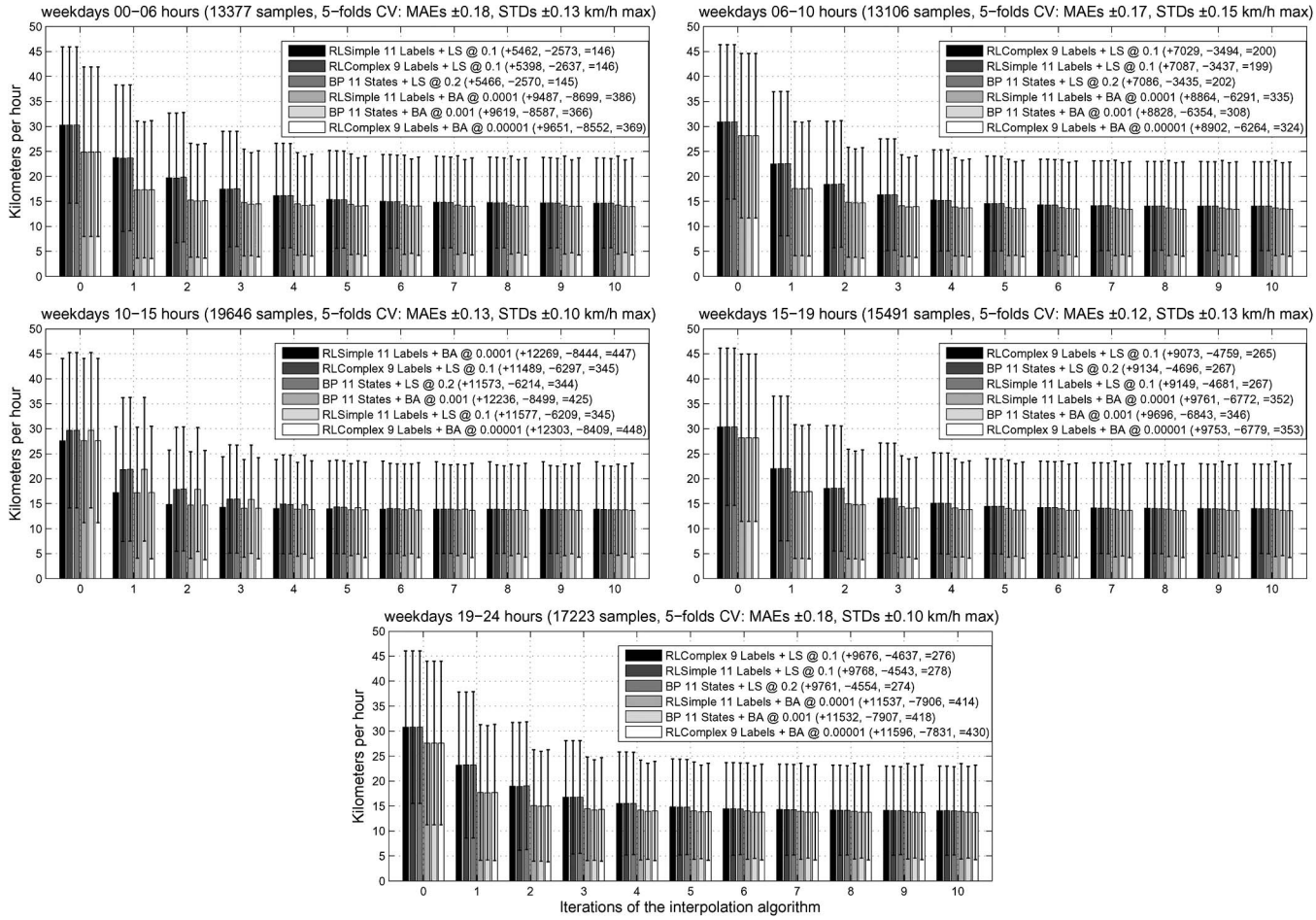

Fig. 8. MAEs and STDs for road segments without historical speed profiles by time of day. In the legend, numbers in parentheses indicate the number of times that we overestimated $(+)$, underestimated $(-)$, and correctly estimated $(=)$ the most likely ambulance speed.

measures to represent uncertainties in the input data and in the output speed estimates. In terms of scale, it can represent the entire city of Ottawa, including both freeways and surface streets. Currently, updating the entire road network takes less than 5 min when running on a single processing core, but this approach can run in much less time if the processing is distributed across multiple processing cores. It has been validated and tested using stratified 5-fold cross validation. Finally, it can easily be ported to other cities and is locally adaptable, making it quite robust to user interventions.

We believe that the work presented in this paper is valuable, because it addresses the sparse and noisy sensor data problem and offers a baseline estimate to which we can compare more complex strategies.

REFERENCES

[1] R. Hummel and S. Zucker, "On the foundations of relaxation labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 3, pp. 267–287, May 1983.

[2] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Exploring Artificial Intelligence in the New Millennium*. San Mateo, CA: Morgan Kaufmann, 2003, pp. 239–269.

[3] P. W. H. Kwan, K. Kameyama, and K. Toraichi, "On a relaxation-labeling algorithm for real-time contour-based image similarity retrieval," *Image Vis. Comput.*, vol. 21, no. 3, pp. 285–294, Mar. 2003.

[4] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 749–764, Aug. 1995.

[5] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 6, pp. 420–433, Jun. 1976.

[6] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vis.*, vol. 70, no. 1, pp. 41–54, Oct. 2006.

[7] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proc. ICPR*, 2006, vol. 3, pp. 15–18.

[8] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation and facial appearance estimation," in *Proc. Comput. Vis. Pattern Recognit.*, 2003, vol. 11, p. 1.

[9] J. Miller, "Dynamically computing fastest paths for intelligent transportation systems," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 1, pp. 20–26, Spring 2009.

[10] J. Miller and E. Horowitz, "Freesim—A free real-time freeway traffic simulator," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2007, pp. 18–23.

[11] Freesim. [Online]. Available: http://www.freewaysimulator.com/

[12] L. E. Owen, Y. Zhang, L. Rao, and G. McHale, "Traffic flow simulation using Corsim," in *Simul. Conf. Proc.*, 2000, vol. 2, pp. 1143–1147.

[13] Corsim. [Online]. Available: http://ops.fhwa.dot.gov/trafficanalysistools/corsim.htm

[14] J. Lei, K. Redmill, and U. Ozguner, "VATSIM: A simulator for vehicles and traffic," in *Proc. IEEE Intell. Transp. Syst.*, 2001, pp. 686–691.

[15] K. A. Redmill and U. Ozguner, "VATSIM: A vehicle and traffic simulator," in *Proc. IEEE Intell. Transp. Syst.*, 1999, pp. 656–661.

[16] Q. I. Yang and H. N. Koutsopoulos, "A microscopic traffic simulator for evaluation of dynamic traffic management systems," *Transp. Res. C*, vol. 4, no. 3, pp. 113–129, Jun. 1996.

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.

[17] L. Smith, R. Beckman, and K. Baggerly, "Transims: Transportation analysis and simulation system," Los Alamos Nat. Lab., Los Alamos, NM, Tech. Rep. LA-UR–95-1641, 1995.
[18] K. Nagel and M. Rickert, "Parallel implementation of the Transims microsimulation," *Parallel Comput.*, vol. 27, no. 12, pp. 1611–1639, Nov. 2001.
[19] G. D. B. Cameron and G. I. D. Duncan, "PARAMICS—Parallel microscopic simulation of road traffic," *J. Supercomput.*, vol. 10, no. 1, pp. 25–53, Mar. 1996.
[20] G. Cameron, B. J. N. Wylie, and D. McArthur, "PARAMICS: Moving vehicles on the connection machine," in *Proc. ACM/IEEE Conf. Supercomput.*, 1994, pp. 291–300.
[21] M. Fellendorf, "VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *Proc. 64th ITE Annu. Meeting*, 1994, vol. 32, pp. 1–9.
[22] Y. Wang, M. Papageorgiou, and A. Messmer, "RENAISSANCE—A unified macroscopic model-based approach to real-time freeway network traffic surveillance," *Transp. Res. C: Emerg. Technol.*, vol. 14, no. 3, pp. 190–212, Jun. 2006.
[23] G. H. Bham and R. F. Benekohal, "A high-fidelity traffic simulation model based on cellular automata and car-following concepts," *Transp. Res. C: Emerg. Technol.*, vol. 12, no. 1, pp. 1–32, Feb. 2004.
[24] H. B. Celikoglu, E. Gedizlioglu, and M. Dell'Orco, "A node-based modeling approach for the continuous dynamic network loading problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 165–174, Mar. 2009.
[25] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 124–132, Mar. 2006.
[26] C. Furtlehner, J. M. Lasgouttes, and A. de La Fortelle, "A belief propagation approach to traffic prediction using probe vehicles," in *Proc. IEEE ITSC*, 2007, pp. 1022–1027.
[27] R. Chrobok, J. Wahle, and M. Schreckenberg, "Traffic forecast using simulations of large-scale networks," in *Proc. IEEE Intell. Transp. Syst.*, 2001, pp. 434–439.
[28] W. Shi and Y. Liu, "Real-time urban traffic monitoring with global positioning system-equipped vehicles," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 113–120, Jun. 2010.
[29] J. Fawcett and P. Robinson, "Adaptive routing for road traffic," *IEEE Comput. Graph. Appl.*, vol. 20, no. 3, pp. 46–53, May/Jun. 2000.
[30] A. Barisone, D. Giglio, R. Minciardi, and R. Poggi, "A macroscopic traffic model for real-time optimization of signalized urban areas," in *Proc. 41st IEEE Conf. Decision Control*, 2002, pp. 900–903.
[31] ESRI Shapefile Technical Description—An ESRI White Paper, Jul. 1998. [Online]. Available: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
[32] Shapefile C library. [Online]. Available: http://shapelib.maptools.org/
[33] Boost C++ libraries. [Online]. Available: http://www.boost.org/
[34] T. Emanuele and V. Alessandro, *Introductory Techniques for 3D Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
[35] A. Phan and F. P. Ferrie, "Obtaining dense road speed estimates from sparse GPS measurements," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2008, pp. 157–162.
[36] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, vol. 2. New York: Wiley, 2001.

Andrew Phan received the B.Eng. and M.Eng. degrees in electrical engineering from McGill University, Montreal, QC, Canada, in 2006 and 2009, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, McGill University, where he works on tracking humans and estimating their pose using a multimodal sensor network.

His research interests include computer vision, particularly optical flow and structure from motion.

Frank P. Ferrie (M'84) received the B.Eng., M.Eng., and Ph.D. degrees in electrical engineering from McGill University, Montreal, QC, Canada, in 1978, 1980, and 1986, respectively.

From 1998 to 2001, he was the Director of the Centre for Intelligent Machines, McGill University. From 2002 to 2004, he was the Director of the Québec Research Network in Distributed Reality. From 2005 to 2006, he was the Associate Dean of Research and Graduate Studies with the Faculty of Engineering, McGill University. He is currently a Professor with the Department of Electrical and Computer Engineering, McGill University, and a Codirector of the Regroupement Stratégique pour l'Étude des Environnements Partagés Intelligents Répartis research network in distributed environments. He is also a Principal Investigator with the Geomatics for Informed Decisions Network of Centers of Excellence. His research interests include computer vision, particularly 2-D and 3-D shape analysis, active perception, dynamic scene analysis, and machine vision. He has extensively published in these areas and is best known for his work in active vision and environmental modeling.

Dr. Ferrie is a member of the IEEE Computer Society and is a Registered Professional Engineer in the Province of Ontario. He is the recipient of the 1999 Young Investigator Award from the Canadian Image Processing and Pattern Recognition Society.

Authorized licensed use limited to: McGill University. Downloaded on January 18,2023 at 17:08:25 UTC from IEEE Xplore. Restrictions apply.