This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

Non-Myopic Adaptive Route Planning in Uncertain Congestion Environments

Siyuan Liu, Yisong Yue, Ramayya Krishnan

Abstract—We consider the problem of adaptively routing a fleet of cooperative vehicles within a road network in the presence of uncertain and dynamic congestion conditions. To tackle this problem, we first propose a Gaussian Process Dynamic Congestion Model that can effectively characterize both the dynamics and the uncertainty of congestion conditions. Our model is efficient and thus facilitates real-time adaptive routing in the face of uncertainty. Using this congestion model, we develop efficient algorithms for non-myopic adaptive routing to minimize the *collective travel time* of all vehicles in the system. A key property of our approach is the ability to efficiently reason about the long-term value of exploration, which enables collectively balancing the exploration/exploitation trade-off for entire fleets of vehicles. Our approach is validated by traffic data from two large Asian cities. Our congestion model is shown to be effective in modeling dynamic congestion conditions. Our routing algorithms also generate significantly faster routes compared to standard baselines, and achieve *near-optimal performance* compared to an omniscient routing algorithm. We also present the results from a preliminary field study, which showcases the efficacy of our approach.

Index Terms—Gaussian process dynamics, adaptive routing, planning under uncertainty

1 Introduction

We consider the problem of collectively routing a fleet of cooperative vehicles in the presence of uncertain travel conditions, with the goal of minimizing the total travel time. Such problem settings naturally arise in contexts such as delivery services, cooperative fleets of automated self-driving vehicles and community-based traffic and navigation applications (e.g., Waze [1]).

Intelligently routing vehicles in urban environments is a challenging problem due to uncertainty in the traffic conditions of the road network. For clarity of exposition, consider the simple example depicted in Figure 1, where seven cooperative vehicles wish to travel from point A (left side) to point B (right side) using either the top road or the bottom road. Suppose from historical measurements that we know the bottom road is faster in expectation. However, there is a reasonable chance that the top road is currently faster. A good *collective* strategy in this setting would be to send the first two vehicles down separate roads in order to observe current traffic conditions on both roads (i.e., vehicles also act as sensors). Afterwards, the remaining vehicles can be routed using much more reliable traffic information.[2]

Two technical challenges arise from this example. First, we require a (probabilistic) model that reliably captures the distribution, or uncertainty, of traffic conditions (e.g., reliably predict that the top road in Figure 1 is faster 40% of the time). Furthermore, the model must be able to (efficiently) predict reliable *posterior*, or updated, distributions given real-time observations.

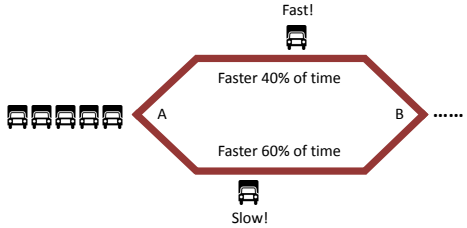


Fig. 1. Simple example collective routing problem with uncertainty. Seven vehicles wish to travel from A (left) to B (right, one destination of several ones). The bottom road is faster in expectation, but the top road may be faster currently. A good collective strategy is to send the first two vehicles down separate roads to observe which road is currently the faster one.

Second, we require routing algorithms that balance the exploration/exploitation trade-off underlying all problems pertaining decision-making under uncertainty – this issue is typically not explicitly considered in the routing literature [1], [2]. In Figure 1, the first two vehicles are engaged in "exploratory" routing so that later vehicles can be routed more optimally. Exploration is inherently non-myopic since it often reduces the utility (i.e., increases the travel time) of the exploring vehicle. As such, effective routing algorithms must be careful to not over-explore, which requires reasoning about the long-term impact of exploration. Our setting is further complicated due to collectively routing a fleet of cooperative vehicles, rather than a single vehicle in isolation – this leads to a complex optimization problem.

- *Siyuan Liu and Ramayya Krishnan are with the Heinz College at Carnegie Mellon University.*
- *Yisong Yue is with the California Institute of Technology.*

1. https://www.waze.com

2. In practice, each vehicle will have their own unique starting and target locations. Note also that our approach does not require vehicles to wait while the first two vehicles are exploring, but rather automatically balances the instantaneous collective exploration/exploitation trade-off for the entire fleet (see Section 4).

In this paper, we make the following contributions:

- We propose a Gaussian Process Dynamic Congestion Model (GPDCM), which can model both the uncertainty and dynamics of traffic conditions while making minimal assumptions about the road network. We show how to fit a GPDCM using historical GPS tracking data.
- We develop an adaptive routing algorithm, called "Planning using Canonical Routes" (PCR), that can effectively reason about the long-term value of exploration for an entire fleet of cooperative vehicles. Our algorithm development yields an intermediate algorithm design that is a variant of the well-studied "upper confidence bound" (UCB) algorithm [3], which serves as a natural "bridge" between conventional adaptive approaches for planning under uncertainty and our more advanced PCR algorithm.
- We fit our GPDCM using real traffic data collected from two large Asian cities, and show that it can effectively capture real-world traffic conditions.
- We show that our adaptive routing approach dramatically out-performs conventional baselines, and achieves near-optimal performance compared to omniscient routing with perfect knowledge of traffic conditions. We also present a preliminary field study showcasing the efficacy of our approach.

A preliminary version of this work appeared in [4]. In this paper, we extend that work with a significantly more thorough algorithm development and empirical validation. In particular, we present a UCB-style routing algorithm that serves as a natural "bridge" between conventional adaptive approaches for planning under uncertainty and our more advanced PCR routing algorithm. Furthermore, we demonstrate that our approach can be applied to more complex tasks, such as the multi-destination routing problem (i.e., Traveling Salesperson).

2 Gaussian Process Preliminaries

Let $\mathcal{R}$ denote our road network, and $\mathcal{Z}$ denote the space of temporal contexts (e.g., containing information about time of day or day of week). We wish to model the travel speed of road segments $r \in \mathcal{R}$ under varying temporal contexts $z \in \mathcal{Z}$. We do so via a function $f : \mathcal{R} \times \mathcal{Z} \rightarrow \Re_+$, that outputs the travel speed for a given (r, z) pair.[3]

We assume that f is sampled probabilistically from a Gaussian process prior $f \sim P(f)$ [5]. A Gaussian process prior is fully specified by its mean function:

$$\mu(r, z) = \mathbf{E}[f(r, z)]$$

its covariance, or kernel, function:

$$\begin{aligned} k((r, z), (r', z')) &= \mathbf{E}[(f(r, z) - \mu(r, z))(f(r', z') - \mu(r', z'))] \\ &= \texttt{cov}((r, z), (r', z')), \end{aligned}$$

and observation noise with variance σ^2.

3. For simplicity, we assume that all vehicles in our fleet are the same type. We can extend our approach by introducing vehicle contexts $v \in \mathcal{V}$ to model the travel speed of different types of vehicles $f(r, z, v)$.

A major computational benefit of Gaussian processes is that posterior inference can be computed in closed form. Suppose we have collected recent observations $Y = [y_1, \ldots, y_T]^\top$ at $X = [(r_1, z_1), \ldots, (r_T, z_T)]$. Then we can write the posterior distribution of f given X and Y also as a Gaussian process distribution with mean:

$$\mu_{Y,X}(r, z) = \mu(r, z) + \hat{k}_X(r, z)^\top (\hat{K}_{Y,X} + \sigma^2 I)^{-1} (\delta Y)^\top \quad (1)$$

and covariance $k_{Y,X}((r, z), (r', z')) =$

$$k((r, z), (r', z')) - \hat{k}_X(r, z)^\top (\hat{K}_X + \sigma^2 I)^{-1} \hat{k}_X(r', z'), \quad (2)$$

where δY is the deviation of Y from its prior mean:

$$\delta Y = [y_1 - \mu(r_1, z_1), \ldots, y_T - \mu(r_T, z_T)]^\top,$$

$\hat{k}_X(r, z)$ is a column vector of the kernel values between (r,z) and every observed location in X:

$$\hat{k}_X(r, z) = [k((r_1, z_1), (r, z)), \ldots, k((r_T, z_T), (r, z))]^\top \in \Re^T$$

and $\hat{K}_X$ is the Gram matrix of all locations in X:

$$\hat{K}_X = [k((r_i, z_i), (r_j, z_j))]_{i,j \in [1,\ldots,T]} \in \Re^{T \times T}.$$

The posterior variance of $f(r, z)$ is $k_{Y,X}((r, z), (r, z))$.

The posterior mean $\mu(r, z|Y, X)$ of $f(r, z)$ deviates from its prior mean $\mu(r, z)$ according to how much (and in what direction) past observations deviated from their prior means in Eq. (1). If some $(r', z') \in X$ had high positive covariance with (r, z), then $\mu(r, z|Y, X)$ would shift along the deviation of the associated observation y' from its mean $y' - \mu(r', z')$. Conversely, if some $(r', z') \in X$ had little covariance with (r, z), the associated y' would have little impact on $\mu(r, z|Y, X)$. Similarly, the posterior covariance between (r, z) and (r', z') decreases as we gather more observations that are related to (i.e., has high prior covariance with) (r, z) and (r', z') in Eq. (2). Thus, the variance of our posterior estimate, $k_{Y,X}((r, z), (r, z))$, of $f(r, z)$ decreases faster when past observations have higher prior covariance with (r, z).

3 Gaussian Process Dynamic Congestion Models

Gaussian Process Dynamic Congestion Model (GPDCM) captures spatial and temporal relationships of road segment travel times within the Gaussian Process framework. We will show how to estimate the mean μ and covariance k to build a Gaussian Process probabilistic model of travel speeds over road segments and temporal contexts. After estimating μ and k from historical data S, posterior inference is fully specified via Eq. (1) and Eq. (2). We assume that the observation noise variance σ^2 is known and/or pre-specified.

3.1 Estimating μ

To estimate μ, we first specify regularities of how μ depends on temporal contexts z. For example, suppose that μ depends only on time-of-day, denoted as $\tau(z)$. Then we can estimate $\mu(r, z)$ as

$$\mu(r, z) = \texttt{mean}\left\{y_{r,z'} : (y_{r,z'} \in S) \wedge (\tau(z') = \tau(z))\right\}, \quad (3)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

where $y_{r,z'}$ is a historically observed travel speed on road r and time-of-day $\tau(z') = \tau(z)$. This regularity allows us to compute the mean travel speed on road r at any future time, but restricts us to modeling all days as being sampled from the same (prior) distribution (i.e., the distribution of travel speeds on Wednesdays is the same as on Sundays). Other regularity assumptions are possible, such as having μ depend on both time of day and day of week.[4]

3.1.1 Temporal Smoothing

Since it is unlikely that historical observations contain many data points with exactly matching time-of-day contexts, we use standard temporal smoothing techniques to estimate a smoothed version of Eq. (3) [6].

We first partition time into intervals (which correspond to each time step in the routing problem). Let $\tau(t)$ now denote the time-of-day interval corresponding to time step t (e.g., 5:10pm - 5:15pm). For road r and time t, we define $Y_S^{(t)}(r)$ as the set of all observed speeds matching $\tau(t)$ at r in the historical data S, i.e.,

$$Y_S^{(t)}(r) = \{y | ((r,z),y) \in S \wedge \tau(z) = \tau(t)\}.$$

Following [6], we employ temporal smoothing of the empirical *cumulative distribution functions* (CDF). We can write this smoothed CDF at road segment r as

$$\mathrm{G}_S^{(t)}(y|r) = \beta \cdot \mathrm{G}_S^{(t-\gamma)}(y|r) + (1-\beta)(1 - \mathrm{CDF}(y|Y_S^{(t)}),$$

where G is the crowdedness value, $\mathrm{CDF}(\mathrm{y}|\mathrm{Y}_\mathrm{S}^{(\mathrm{t})})$ denotes the empirical CDF w.r.t. $Y_S^{(t)}$, and β and γ control for the degree of smoothing. We refer to [6] for more details. Afterwards, we can compute the empirical mean Eq. (3) using $\mathrm{G}_\mathrm{S}^\mathrm{z}(\mathrm{y}|\mathrm{r})$ via $\mu(r,z) = \mathbf{E}_{y\sim H}[y]$.

3.2 Estimating k

For simplicity, we decompose the covariance kernel into road and context components, i.e.,

$$k((r,z),(r',z')) = k_1(r,r')k_2(z,z'). \tag{4}$$

This decomposition assumes that all roads share the same temporal covariance k_2; while somewhat idealistic, this offers a compact and efficient representation.

3.2.1 Estimating Road Covariance k_1

Estimating $k_1(r,r')$ is relatively straightforward, as it is essentially the covariance of the travel speeds between two road segments:

$$k_1(r,r') = \mathtt{mean}\left\{(y_{r,z} - \mu(r,z))(y_{r',z} - \mu(r',z)) : y_{r,z}, y_{r',z} \in S\right\}. \tag{5}$$

Note that in Eq. (5) the two historical observations $y_{r,z}$ and $y_{r',z}$ share the exact same temporal context (i.e., were observed at the exact same time), which leads to a data sparsity issue. As such, we also employed temporal smoothing (see Section 3.1.1) to estimate k_1 more reliably.

Intuitively, if two roads r and r' have high covariance according to historical data S, then $k(r,r')$ will be high. As such, observations of r' provide more information about the conditions on r than another road segment with low covariance with r'. Note that since the roads in our system form a finite set, we can precompute the entire road-road kernel matrix.

3.2.2 Estimating Temporal Covariance k_2

Estimating $k_2(z,z')$ is slightly more complicated, as we must specify additional regularities about z. For example, we can assume that $k_2(z,z')$ only depends on the difference in time $|z-z'|$, which leads to

$$k_2(z,z') = \mathtt{mean}\left\{(y_{r,z_1} - \mu(r,z_1))(y_{r,z_2} - \mu(r,z_2)) : (y_{r,z_1}, y_{r,z_2} \in S) \wedge (|z_1 - z_2| = |z - z'|)\right\}. \tag{6}$$

Here we compute the empirical covariance of travel speeds at time difference $|z-z'|$ averaged over every road. As when estimating k_1, we also employ temporal smoothing to estimate k_2 more reliably.

Intuitively, we expect that the covariance of travel speeds in the historical data S should decrease as the time difference increases. Thus, from the perspective of each y_{r,z_1}, the distribution of y_{r,z_2} should look more like the background distribution as $|z_1 - z_2|$ increases.

3.3 Efficiency Considerations

3.3.1 Mean and Kernel Functions

Although Eq. (3), Eq. (5) and Eq. (6) correspond to (modulo regularity assumptions) the "correct" mean and covariance estimates,[5] their non-parametric form can be expensive to evaluate. Naively, computing Eq. (3), Eq. (5) and Eq. (6) requires reprocessing S for each evaluation, which may be prohibitively expensive in some cases.

There are two general strategies to address this issue. The first involves using a compact functional form for Eq. (3), Eq. (5) and Eq. (6) that precludes the need to reprocess S for each evaluation. For example, one can use a linear parametric family to approximate μ, $\mu(r,z) = w^\top \phi(r,z)$, where $\phi(r,z)$ denotes a feature mapping of (r,z) to a vector of features, and w denotes parameters that can be estimated using S. Another example is to use an RBF kernel to approximate k_1, $k_1(r,r') = \exp\left\{-d_1(r,r')^2\right\}$, where $d_1(r,r')$ is a distance function that and can be selected from a family of functions $d_1 \in \mathcal{D}_1$.[6] This approach has the advantage of compactly representing μ, k_1 and k_2 in a way that does not require memorizing S. However, designing suitable functional forms that well approximate Eq. (1), Eq. (5) and Eq. (6) can be difficult.

4. We can alternatively use a feature representation of the context $\phi(z)$ and a parametric model to estimate $\mu(r,z)$.

5. I.e., they are the data-driven mean and covariance functions estimated directly from historical traffic data.

6. One can also define an analogous kernel model for k_2 (using a notion of temporal distance rather than spatial distance).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

The second strategy involves sub-sampling S when evaluating Eq. (3), Eq. (5) and Eq. (6). Compared to the first approach described above, this second approach has the advantage of not requiring a suitable functional form for μ, k_1, and k_2. One disadvantage is that one may still require a sizable fraction of S in order to achieve a good approximation. We empirically test how much sub-sampling is required in our experiments.

3.3.2 Posterior Inference

$K_{Y,X}$ matrix in Eq. (2) grows quadratically in size w.r.t. the number of observations, which makes inference costly as we collect more observations. Intuitively, we should expect that very old observations do not impact our model much when trying to predict future conditions (since the time difference is large, the k_2 component from very old observations should be close to 0). As such, we should be able to discard old observations from our model without compromising model accuracy. We empirically verify this effect in our experiments.

4 Adaptive Collective Routing

We consider the setting where N cooperative vehicles must travel from source locations $A = \{a_1, \ldots, a_N\}$ to destinations $B = \{b_1, \ldots, b_N\}$ Each b_n can contain multiple destinations that vehicle n must visit. We discretize time into intervals (e.g., every minute), and assume that congestion conditions behave according to a GPDCM.[7] We define our routing problem in Figure 2 below, where `Planner` denotes the routing algorithm.

Define $Y_0 \leftarrow \emptyset$ and $X_0 \leftarrow \emptyset$. For each time step $t = 1, \ldots, \infty$ the following occurs:

- Receive state of vehicles $L_t = \{\ell_{t,1}, \ldots, \ell_{t,N}\}$
- If all vehicles have reached their destinations, then terminate
- Define $O_{t-1} = (X_{t-1}, Y_{t-1})$
- Compute routing $\Psi_t \leftarrow \texttt{Planner}(L_t, O_{t-1})$
- Execute routing Ψ_t for one time step
- Receive congestion measurements $y_t = \{y_{t,1}, \ldots, y_{t,N}\}$
- Define $Y_t = Y_{t-1} \cup \{y_t\}$
- Define $X_t = X_{t-1} \cup \{L_t\}$

Fig. 2. Adaptive Collective Routing Problem Definition

In our setting, each measurement $y_{t,n}$ corresponds to the observed travel speed for vehicle for time step t, and each state $\ell_{t,n}$ corresponds to a road/context pair (r, z) for vehicle n at time step t. The goal then is to develop a route planning algorithm (`Planner`) to minimize the collective travel time of all N vehicles.

The basic structure of our approach is shown in Algorithm 1. The key input is a utility function $\delta U_n(r|\Psi)$ that describes the utility of any route r for vehicle n given partial routing solution Ψ (which contains routes for a subset of the N vehicles, and is initially empty). For example, for simple static routing, δU_n can be defined simply as the negative expected travel time according the prior GPDCM distribution.

Algorithm 1 proceeds by sequentially adding routes to Ψ. For each vehicle not yet routed, the optimal route is computed according to δU_n (Line 6). Note that we will employ standard routing algorithms to solve this step. The route that provides the maximum utility is chosen and added to the solution Ψ (Lines 7-9). This process continues until all vehicles have been routed.

In the following, we first show in Section 4.1 how to quantify the long-term value of exploration using information gain. Using information gain, we show in Section 4.2 how to specify δU_n in order to balance the exploration/exploitation tradeoff. This leads to a variant of the conventional Gaussian Process Upper Confidence Bound (GP-UCB) approach [3] extended to the routing setting. In Section 4.3, we present a more accurate way to quantify the value of exploration, which focuses on routes that are likely to be optimal – we call these the *canonical routes*. Leveraging canonical routes, we arrive at an improved approach, which we call Planning using Canonical Routes (PCR). The two approaches, GP-UCB and PCR, are described in Algorithms 2 and 3, respectively. Note that both approaches specify δU_n and then run a routing subroutine such as Algorithm 1. We finally present more efficient routing subroutines in Section 4.4.

4.1 Information Gain as Value of Exploration

The most accurate approach for quantifying the value of exploration is using the expected utility gain of future routing decisions [7]. However, given the exponentially many possible future routing decisions, computing this quantity is intractable using naive approaches.

We instead develop our exploration criterion based on uncertainty reduction of our congestion model f. One natural approach is to use information gain [8]:

$$IG(y_\Psi; f) \equiv H(y_\Psi) - H(y_\Psi|f) \equiv H(f) - H(f|y_\Psi), \quad (7)$$

where $H(\cdot)$ denotes the standard information entropy, $H(\cdot|\cdot)$ denotes the standard conditional entropy, and $y_\Psi = \{y_{r,\tau}\}_{(r,\tau)\in\Psi}$ denotes the observations made by some set of road segments traveled. In other words, $IG(y_\Psi; f)$ is the reduction in entropy of our congestion model f given observations y_Ψ. Thus, a reasonable exploration criterion is to select the route Ψ that maximizes $IG(y_\Psi; f)$ (we will discuss an alternative in Section 4.3).

As we will discuss in Section 4.4, $IG(y_\Psi; f)$ is monotone submodular with respect to y_Ψ, which implies that simple greedy forward-selection can be effective in maximizing $IG(y_\Psi; f)$. Given an intermediate Ψ, we are thus interested in identifying the observation $\hat{y}$ that maximizes the incremental information gain:

$$\delta IG(\hat{y}|y_\Psi, f) = IG(y_\Psi \cup \{\hat{y}\}; f) - IG(y_\Psi; f). \quad (8)$$

7. Since we discretize time into intervals, we use the average travel speed of each road segment for routing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1 GREEDYROUTING: Greedy Route Planning

Input: $L = \{\ell_1, \ldots, \ell_N\}$ *//current locations of each vehicle*
Input: $\delta U_1, \ldots, \delta U_N$ *//utility functions of each vehicle to maximize for during routing*
Initialize $W \leftarrow \{1, \ldots, N\}$
Initialize $\Psi_0 \leftarrow \emptyset$ *//routing solution, initially empty*
for $j = 1, \ldots, N$ **do**
  For each vehicle $n \in W$, compute $\vec{r}_n \leftarrow \text{argmax}_{\vec{r} \in P_n} \delta U_n(\vec{r}|\Psi_{j-1})$ *//compute maximum utility route for vehicle n*
  $\hat{n} \leftarrow \text{argmax}_n \delta U_n(\vec{r}_n|\Psi_{j-1})$
  $W \leftarrow W \setminus \{\hat{n}\}$
  $\Psi_j \leftarrow \Psi_{j-1} \cup \{\vec{r}_{\hat{n}}\}$
end for
Return Ψ_N

Algorithm 2 GP-UCB for Routing

Input: $L = \{\ell_1, \ldots, \ell_N\}$ *//current locations of each vehicle*
Input: $O \equiv (X, Y)$ *//recent observations*
Input: α *//trade-off between exploration vs exploitation*
Input: ROUTINGALG *//structured routing algorithm to use, e.g.,* GREEDYROUTING
Define posterior mean $\forall (r, z) : \ \mu_O(r, z)$ *//see Eq.* (1) *and Eq.* (3)
Define posterior cov. $\forall (r, z), (r', z') : \ k_O((r, z), (r', z'))$ *//see Eq.* (2)*, Eq.* (4)*, Eq.* (5) *and Eq.* (6)
For each vehicle n, define $\delta U_n(\vec{r}|\Psi) \equiv c_n(\vec{r}) + \alpha k(\vec{r}, \vec{r}|O, \Psi)$ *//see Eq.* (12) *and Eq.* (10)
Return ROUTINGALG$(L, \delta U_1, \ldots, \delta U_N)$

In the case where we are purely exploring, we can define δU to be Eq. (8).

We make two approximations in order to efficiently approximate Eq. (8) during routing. Both approximations stem from the fact that we must compute Eq. (8) for an entire route $\vec{r}$, rather than a single observation:

$$\delta IG(y_{\vec{r}}|y_\Psi, f) = IG(y_\Psi \cup y_{\vec{r}}; f) - IG(y_\Psi; f). \tag{9}$$

First, we cannot determine exactly when a vehicle will traverse each road segment of a route. As such, for each route $\vec{r} \in \Psi$, we approximate $y_{\vec{r}}$ using $y_{\vec{r}} = \{y_{r,\tau}\}_{(r,\tau) \in E[\vec{r}]}$, where

$$E[\vec{r}] = \langle (r_1, \tau_1), \ldots, (r_{|\vec{r}|}, \tau_{|\vec{r}|}) \rangle$$

contains the expected times τ of traveling each $r \in \vec{r}$.

Second, we approximate Eq. (9) using variance reduction, which is closely related to information gain when f is a Gaussian process [3], [9]. We further decompose variance reduction additively along each road segment $r \in \vec{r}$. Thus, the incremental variance reduction of running any route $\vec{r}$ given intermediate solution Ψ and recent observations $O \equiv (X, Y)$ can be written as

$$k(\vec{r}, \vec{r}|O, \Psi) = \sum_{(r,\tau) \in E[\vec{r}]} k((r, \tau), (r, \tau)|O, \Psi, E[\vec{r}]_{\prec(r,\tau)}), \tag{10}$$

where $k(\cdot, \cdot|O, \Psi) \equiv k_{O \cup \Psi}(\cdot, \cdot)$ Eq. (2), and $\vec{r}_{\prec(r,\tau)}$ denotes the road segments in $\vec{r}$ that a traversed before (r, τ).

Maximizing Eq. (9) is in general intractable, and is related to the submodular orienteering problem [10], [11], [12]. In practice, we employ conventional routing algorithms such as depth-first routing,[8] which perform very well in our experiments and are fast to compute.

8. In the case where the utilities are additive (rather than subadditive as in our case), depth-first routing is identical to Dijkstra's.

In summary, we can use Eq. (9) and Eq. (10) to efficiently quantify the long-term value of exploration when planning routes. In Section 4.2, we show how to balance exploration and exploitation for effective non-myopic adapting routing. In Section 4.3, we will show how to extend Eq. (9) to more accurately quantify the value of exploration for our problem setting.

4.2 Balancing Exploration versus Exploitation

We now show how to balance exploration/exploitation trade-off for effective non-myopic adaptive routing (see Algorithm 2). This approach can be viewed as an extension of the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm [3] to the routing setting.

GP-UCB is run once per time step (i.e., it instantiates `Planner` in Figure 2), and uses a utility function U that trades off between exploration and exploitation:

$$U(\Psi) = \sum_{n=1}^{N} c_n(\Psi \cap P_n) + \alpha IG(y_\Psi, f), \tag{11}$$

where IG is the information gain Eq. (7) that represents the exploration term, $\alpha \geq 0$ is a parameter that trades off between exploration and exploitation, P_n is the set of all feasible routes for vehicle n (i.e., any route for vehicle n that ends at its target B_n), and c_n is the exploitation term that measures the difference in expected travel time between the selected route for vehicle n and the best route $\vec{r}_n$ (according to the current posterior distribution):

$$c_n(\Psi) = \begin{cases} 0 & \text{if } |\Psi| = 0 \\ \max_{\vec{r} \in \Psi} \texttt{time}(\vec{r}_n^*) - \texttt{time}(\vec{r}) & \text{otherwise} \end{cases} \tag{12}$$

When vehicle n has no route assigned, we define $c_n = 0$. When vehicle n has multiple routes assigned (which is used in the analysis only), we choose the best route

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

Algorithm 3 Planning using Canonical Routes (PCR)

1: **Input**: $L = \{\ell_1, \ldots, \ell_N\}$ *//locations of each vehicle*
2: **Input**: $O \equiv (X, Y)$ *//recent observations*
3: **Input**: M *//number of future scenarios to sample*
4: **Input**: α *//trade-off between exploration vs exploitation*
5: **Input**: ROUTINGALG *//structured routing algorithm to use, e.g.,* GREEDYROUTING
6: Define posterior mean $\forall(r,z):\ \mu_O(r,z)$ *//see Eq.* (1) *and Eq.* (3)
7: Define posterior cov. $\forall(r,z),(r',z'):\ k_O((r,z),(r',z'))$ *//see Eq.* (2)*, Eq.* (4)*, Eq.* (5) *and Eq.* (6)
8: Sample M traffic scenarios $\{\hat{f}_k\}_{k=1}^M$ from posterior using $\mu_O(r,z)$ and $k_O((r,z),(r',z'))$.
9: For each vehicle n, compute all optimal routes $\{\vec{r}^*_{n,i}\}_{i=1}^M$ from the M traffic scenarios.
10: For each vehicle n, define $\delta U_n^{(j)}(\vec{r}) \equiv c_n(\vec{r}) + (\alpha/M)\sum_{n',i}\delta k(\vec{r}, \vec{r}^*_{n',i}|O, \Psi_{j-1})$ *//see Eq.* (12) *and Eq.* (17)
11: **Return**: ROUTINGALG$(L, \delta U_1, \ldots, \delta U_N)$

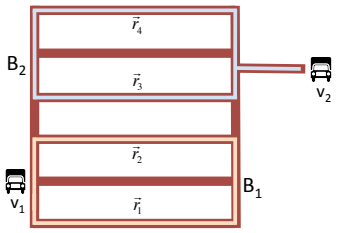


Fig. 3. Simple example of canonical routes. Vehicles v_1 and v_2 must travel to B_1 and B_2, respectively. There are two canonical routes, $\vec{r}_1$ and $\vec{r}_2$ (depicted in light orange), for v_1, and two canonical routes, $\vec{r}_3$ and $\vec{r}_4$ (depicted in light blue), for v_2. Canonical routes correspond to routes that could be optimal depending on what the traffic conditions actually are, and can be computed by first sampling probable traffic scenarios from a GPDCM fitted to historical traffic data, and then computing the optimal routes for each vehicle under those sampled scenarios.

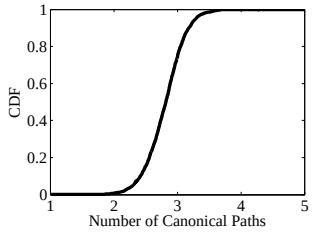


Fig. 4. Distribution of the number of canonical routes for each individual vehicle routing problem (see Section 4.3). On average, there are 3.2 canonical routes per vehicle.

for c_n. Note that $c_n \leq 0$ always, since $\vec{r}^*_n$ has the minimal travel time under the current posterior.

Our goal is to find a Ψ that (approximately) solves:

$$\underset{\Psi, \forall n: |\Psi \cap P_n| = 1}{\operatorname{argmax}} \quad U(\Psi). \tag{13}$$

We can define the incremental utility gain:

$$\delta U(\vec{r}|\Psi) = U(\Psi \cup \{\vec{r}\}) - U(\Psi) = \sum_{n=1}^{N} \mathbf{1}_{[\vec{r} \in P_n]} \delta U_n(\vec{r}|\Psi),$$

where

$$\delta U_n(\vec{r}|\Psi) = c_n(\vec{r}) + \alpha \delta IG(y_{\vec{r}}|y_\Psi, f), \tag{14}$$

for δIG defined as in Eq. (9). Using the approximation described in Eq. (10), we arrive at the final incremental gain function used in Algorithm 2:

$$\delta U_n(\vec{r}|\Psi) \equiv c_n(\vec{r}) + \alpha k(\vec{r}, \vec{r}|O, \Psi). \tag{15}$$

4.3 Planning using Canonical Routes

One limitation of conventional information gain Eq. (7) is that it equally values all road segments in the network. However, it can often be that only a few routes could ever be optimal for a given vehicle's routing task. For instance, given a GPDCM, one can sample probable future traffic scenarios. Each sample contains *complete information*, i.e., the travel speeds of all routes at all times (cf. [13], [14]), which allows us to compute the optimal route for that scenario. We call such optimal routes the *canonical routes*.[9]

Figure 3 shows a simple example depicting the canonical routes for two vehicles. Each vehicle has two canonical routes, i.e., two routes that could possibly be optimal depending on what the traffic conditions actually are. As mentioned above, these canonical routes are computed by first sampling probable traffic scenarios from a GPDCM fitted to historical traffic data, and then computing the optimal routes under those scenarios.

Figure 4 shows the distribution of the number of canonical routes per routing task on a real road network.[10] We see that the number of canonical routes is often quite small – about 3.2 on average. Thus, we can typically efficiently enumerate all canonical routes by computing the optimal routes under sampled scenarios from our GPDCM (see Lines 8-9 in Algorithm 3).

If a routing algorithm *knew exactly* which of the sampled scenarios reflects future traffic conditions, then that corresponding canonical route would be the optimal action to take. As such, it is intuitively beneficial to focus exploration on reducing uncertainty as to which canonical route is actually optimal. For example, suppose that routes $\vec{r}_2$ and $\vec{r}_3$ in Figure 3 are highly correlated in travel

9. Note that canonical routes can range from being disjoint to only varying slightly from each other, depending on the routing task, the topology of the road network, and the distribution of travel speeds.

10. We computed Figure 4 by sampling from a GPDCM estimated using real traffic data from a real road network (see Section 5), and using a diverse set of source and destination locations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

Algorithm 4 LAZYGREEDYROUTING: Lazy Greedy Route Planning

Input: $L = \{\ell_1, \ldots, \ell_N\}$ *//current locations of each vehicle*
Input: $\delta U_1, \ldots, \delta U_N$ *//utility functions of each vehicle to maximize for during routing*
Initialize $W \leftarrow \{1, \ldots, N\}$
Initialize $\Psi_0 \leftarrow \emptyset$ *//routing solution, initially empty*
Initialize $\forall n \in W: \ u_n \leftarrow \infty$ *//the last evaluated utility maximum utility route for vehicle n, initially negative infinity*
Initialize $\forall n \in W: \ \tau_n \leftarrow 0$ *//the iteration when vehicle n was last evaluated, initially 0*
Define priority queue $Q \leftarrow \texttt{sort}\{(n, u_n, \tau_n, \emptyset) : n \in W\}$ *//sort in decreasing order of u_n, most negative to most positive*
for $j = 1, \ldots, N$ **do**
  $(n, u_n, \tau_n, \vec{r}_n) \leftarrow$ pop from top of Q
  while $\tau_n < j$ **do**
    $\vec{r}_n \leftarrow \text{argmax}_{\vec{r} \in P_n} \delta U_n(\vec{r}|\Psi_{j-1})$ *//compute maximum utility route for vehicle n*
    $u_n \leftarrow \delta U_n(\vec{r}_n|\Psi_{j-1})$
    $\tau_n \leftarrow j$
    Sorted insert $(n, u_n, \tau_n, \vec{r}_n)$ into Q *//sort in decreasing order of u_n, most positive to most negative*
    $(n, u_n, \tau_n, \vec{r}_n) \leftarrow$ pop from top of Q
  end while
  $W \leftarrow W \setminus \{n\}$
  $\Psi_j \leftarrow \Psi_{j-1} \cup \{\vec{r}_n\}$
end for
Return Ψ_N

speed (i.e., both tend to be congested or free-flowing at the same time). Then it would be beneficial to route vehicle v_1 along $\vec{r}_2$, since that would reveal information regarding the current travel speed along $\vec{r}_3$, and thus allow vehicle v_2 to be routed more intelligently.

Let $R = \{\vec{r}'_n\}$ denote the set of canonical routes for all vehicles. Then we can quantify the value of exploration using the expected information gain between our routing solution and R, $E_R[IG(y_\Psi; f_R]$, where

$$IG(y_\Psi; f_R) \equiv H(y_\Psi) - H(y_\Psi|f_R) \equiv H(f_R) - H(f_R|y_\Psi),$$

and $f_R = \{f_{r,\tau}\}_{(r,\tau)\in R}$ denotes the values of f only at R (rather than all of f).

Analogous to Section 4.1, we are interested in approximating the incremental information gain for f_R:

$$\delta IG(y_{\vec{r}}|y_\Psi, f_R) = IG(y_\Psi \cup \{y_{\vec{r}}\}; f_R) - IG(y_\Psi; f_R). \quad (16)$$

Using a similar approach as Eq. (10), we can approximate Eq. (16) as the variance reduction in each $\vec{r}' \in R$:

$$\delta k(\vec{r}, \vec{r}'|O) = k(\vec{r}', \vec{r}'|O) - k(\vec{r}', \vec{r}'|O, \vec{r}), \quad (17)$$

for $K(\vec{r}', \vec{r}'|O)$ defined as in Eq. (10), and $K(\vec{r}', \vec{r}'|O, \vec{r})$ defined as:

$$\sum_{(r,\tau)\in E[\vec{r}']} k((r,\tau),(r,\tau)|O, \Psi, E[\vec{r}']_{\prec(r,\tau)}, E[\vec{r}]_{\prec(r,\tau)}). \quad (18)$$

Following Eq. (13), we can simply define a new utility function to optimize for:

$$U(\Psi) = \sum_{n=1}^{N} c_n(\Psi \cap P_n) + \alpha E_R\left[\sum_{\vec{r}'\in R} IG(y_\Psi; f_{\vec{r}'})\right]. \quad (19)$$

Following Eq. (15), we can specify the corresponding incremental gain:

$$\delta U_n(\vec{r}|\Psi) = c_n(\vec{r}) + \alpha E_R\left[\sum_{\vec{r}'\in R} \delta IG(y_{\vec{r}}|y_\Psi, f_{\vec{r}'})\right]. \quad (20)$$

We approximate the expectation $E_R[\ldots]$ using an empirical sample. We first sample M traffice scenarios $\{\hat{f}_k\}_{k=1}^M$ from posterior using $\mu_O(r,z)$ and $k_O((r,z),(r',z'))$. For each vehicle n, we then compute all canonical routes $\{\vec{r}^*_{n,i}\}_{i=1}^M$ from the M traffic scenarios. Finally, we can define the sample approximation to Eq. (20) as:

$$\delta U_n^{(j)}(\vec{r}) \equiv c_n(\vec{r}) + (\alpha/M) \sum_{n',i} \delta k(\vec{r}, \vec{r}^*_{n',i}|O, \Psi_{j-1}). \quad (21)$$

Algorithm 3 describes this approach, which we call Planning using Canonical Routes (PCR). Algorithm 3 is more computationally intensive than Algorithm 2, which we will empirically compare.

4.4 Connection to Submodular Optimization

The reason for recomputing the highest utility routes each iteration (e.g., Lines 6-9 in Algorithm 1) is because the incremental utility δU changes after committing to a route. For instance, if the two best routes from the current iteration provide uncertainty reduction to the same regions of f, then after the algorithm commits to one of those routes, the value of exploration for the other route has decreased dramatically. This notion of diminishing returns is often modeled using submodularity [15].

In fact, one can show that U in Eq. (11) is a quasi-monotone submodular utility function. We say that U is **quasi-monotone** if

$$\forall n, \Psi, \vec{r} \in P_n: \ |\Psi \cap P_n| > 0 \Rightarrow U(\Psi \cup \{\vec{r}\}) \geq U(\Psi),$$

and that U is approximately **submodular** if $\forall \Psi, \Psi', \vec{r}:$

$$U(\Psi \cup \{\vec{r}\}) - U(\Psi) \geq U(\Psi \cup \Psi' \cup \{\vec{r}\}) - U(\Psi \cup \Psi') - \epsilon,$$

where $\epsilon \geq 0$ and $\epsilon = 0$ if U is exactly submodular.

Lemma 1. *U in Eq. (11) is quasi-monotone submodular.*

All proofs are deferred to Section 9. The optimization problem in Eq. (13) is a submodular maximization

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering

Algorithm 5 RANDOMIZEDGREEDYROUTING: Randomized Greedy Route Planning

1: **Input**: $L = \{\ell_1, \ldots, \ell_N\}$ *//current locations of each vehicle*
2: **Input**: $\delta U_1, \ldots, \delta U_N$ *//utility functions of each vehicle to maximize for during routing*
3: Initialize $\Psi_0 \leftarrow \emptyset$ *//routing solution, initially empty*
4: Define random permutation π over the N vehicles
5: **for** $j = 1, \ldots, N$ **do**
6: $\quad \vec{r}_{\pi(j)} \leftarrow \text{argmax}_{\vec{r} \in P_n} \delta U_{\pi(j)}(\vec{r}|\Psi_{j-1})$ *//compute maximum utility route for vehicle $\pi(j)$*
7: $\quad \Psi_j \leftarrow \Psi_{j-1} \cup \{\vec{r}_{\pi(j)}\}$
8: **end for**
9: **Return** Ψ_N

problem under matroid constraints for U defined as Eq. (11) [16], [17], and we can show that any algorithm that greedily selects the route with (approximately) maximal incremental gain achieves a constant factor approximation to the global optimum (for that U).

Lemma 2. *Let Ψ^* denote the global maximizer of Eq.* (13) *for U quasi-monotone and approximately submodular. Let Ψ_i denote the intermediate solution of a greedy algorithm after selecting i routes. Suppose the greedy algorithm next selects a route $\vec{r}_{i+1}$ to construct $\Psi_{i+1} \leftarrow \Psi_i \cup \{\vec{r}_{i+1}\}$ satisfying:*

$$U(\Psi_i \cup \{\vec{r}_i\}) - U(\Psi_i) \geq \beta \max_{\vec{r}^*} \left(U(\Psi_i \cup \{\vec{r}^*\}) - U(\Psi_i)\right),$$

for $\beta \in [0, 1]$, and that $\forall \Psi', \vec{r}$:

$$U(\Psi_{i-1} \cup \{\vec{r}\}) - U(\Psi_{i-1}) \geq U(\Psi_{i-1} \cup \Psi' \cup \{\vec{r}\}) - U(\Psi_{i-1} \cup \Psi') - \epsilon_i.$$

Then we have

$$U(\Psi_N) \geq \frac{\beta}{1+\beta} \left(U(\Psi^*) - \sum_{i=1}^{N} \epsilon_i\right),$$

where Ψ_N is the final solution of the greedy algorithm.

Although we cannot provide precise guarantees on the behavior of Algorithm 2 due to the approximations described in Section 4.1, intuitively we expect Algorithm 2 (using Algorithm 1 for ROUTINGALG) to be effective at maximizing the incremental gain of Eq. (11) at each iteration. We expect β to be close to 1 and ϵ_i to be close to 0, which implies that Algorithm 2 is close to a $1/2$ approximation of the maximizer of Eq. (11).

The utility function based on canonical routes Eq. (19) is not submodular, since information gain is not submodular. However, we observe in practice that Eq. (19) behaves very similarly to a submodular function (i.e., each ϵ_i is small) , and so we should expect Algorithm 3 (using Algorithm 1) to perform near-optimally as well.

We empirically validate both algorithms and show that both perform quite well, with Algorithm 3 approaching near-optimal performance relative to an omniscient routing algorithm with full knowledge of future congestion information. It would be interesting to develop a posteriori guarantees on optimality based on the behavior of the greedy algorithm in a specific problem instance.

4.4.1 More Efficient Routing

We now present more efficient routing subroutines that (in some cases) provably behave similarly Algorithm 1. It is known that a lazy variant of Algorithm 1 can compute identical solutions for monotone submodular utility functions [18]. Algorithm 4 describes this lazy variant, which can also be used to instantiate ROUTINGALG in Algorithms 2 & 3. In Algorithm 1, each round of optimization (Lines 6-9) requires computing the routing solution of *every unassigned vehicle* (Line 6). In contrast, in Algorithm 4, each round of optimization (Lines 9-18) only considers a subset of the unassigned vehicles.

Lemma 3. *Algorithm 2 using Algorithm 1 and Algorithm 2 using Algorithm 4 return identical solutions.*

Furthermore, a perhaps surprising property of the proof of Lemma 2 is that it applies to *any* ordering of the vehicles to be routed (and not necessarily the one chosen by Algorithm 1). Thus, Lemma 2 also applies when we greedily route the vehicles in an arbitrary order, which leads to further computational savings since we only need to solve the routing problem for each vehicle once. We chose to use a random ordering, and this routing subroutine is described in Algorithm 5.

5 EMPIRICAL EVALUATION

We conduct two types of empirical evaluation. First, we evaluate whether our GPDCM can accurately predict future traffic scenarios given recent observations. Second, we compare the performance of our routing algorithms versus conventional baselines as well as omniscient routing with perfect knowledge of traffic conditions.[11]

We obtained two datasets consisting of GPS traces from a large number vehicles:

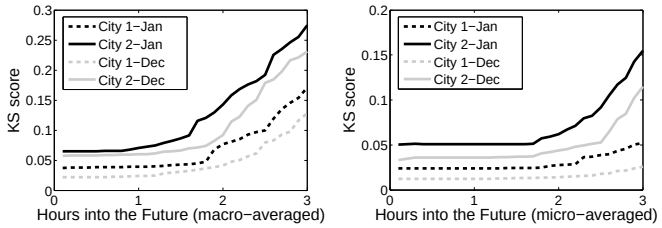
- **City 1**: One year (from January 2008 to December 2008) of GPS data from approximately 15,000 taxis in Shenzhen, China.
- **City 2**: One year (from January 2006 to December 2006) of GPS data from approximately 5,600 taxis in Shanghai, China.

5.1 Gaussian Process Validation

We first validate the effectiveness of our GPDCM, and verify that it can accurately predict the distribution of future traffic scenarios given real-time observations.

11. All of our experiments were conducted using a workstation with a Intel Core Quad CPU (Q9550 2.83 GHz) and 32 GB of main memory. All the algorithms are implemented with C/C++. Unless specified otherwise, all the experiments were conducted in the time period of 8am to 10am on each day from Monday to Friday and then we took the mean value.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering



(a) Macro-averaged KS score (b) Micro-averaged KS score

Fig. 5. (a) Showing the macro-averaged KS score (computing KS score for each day separately) on the two datasets for months of January and December. (b) Showing the micro-averaged KS score (by aggregating all days into a single day).

5.1.1 Kolmogorov-Smirnov Test

In order to test how well our GPDCM predicts *a distribution* of future traffic conditions, we require a measure of how well a predicted distribution matches the true or empirical distribution observed in nature (i.e., the observed distribution of travel speeds). We choose to use the Kolmogorov-Smirnov test [19], which has a long tradition in the statistics community for quantifying the distance between two distributions.[12]

Let F_n denote the empirical cumulative distribution function (CDF) for n i.i.d. random variables X_i:

$$F_x(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \leq x} \tag{22}$$

where $I_{X_i \leq x}$ is an indicator function (i.e., 1 if $X_i \leq x$, and 0 otherwise). The Kolmogorov-Smirnov score for $F(x)$ and an empirical CDF $F_n(x)$ is

$$Z_n = \sup_x \left| F_n(x) - F(x) \right|, \tag{23}$$

which corresponds to the supremum over all absolute distances between the two distributions. The lower the KS score, the better the distributional fit. Two identical distributions will have a KS score of 0, and typically a KS score of less than 0.1 is indicative of a strong distributional fit [19].

5.1.2 Main Results

For each city, we fit a GPDCM (μ and k) using six months of data (from February to June). We then evaluate the accuracy of our GPDCM via the KS score on held-out data. For each day, we give the GPDCM with two hours of observations from 8am-10am. We then measure the KS score between the GPDCM posterior and the future empirical distribution partitioned to one hour windows.

Figure 5(a) shows the KS score averaged over multiple days (i.e., macro-averaged) for both datasets for the months of January and December (which are the furthest apart months in both datasets). We observe that the KS score is quite low for the first three hours, indicating that our GPDCM predicts accurate near-term traffic distributions. The KS score degrades as the GPDCM predicts traffic distributions further into future, and eventually decays to the "background" prior KS score.

The performance difference for different cities and months in Figure 5(a) can be explained as follows. Since we have more data from City 1, we should expect a better model fit. Likewise, our datasets also contain more data from December, which explains why our model makes more accurate (from a distributional point of view) predictions for the month of December. As such, one can interpret this difference in performance between December and January as being primarily attributed to data sparsity in the test set.

Figure 5(b) shows the KS score computed over all days (i.e., micro-averaged) for both datasets for the months of January and December. The results are qualitatively similar to Figure 5(a). Because the empirical distribution is aggregated over all days (e.g., all [10am-11am] time windows for each day in January), the KS score is lower than the results in Figure 5(a) due to lower data sparsity.

These results suggest that the GPDCM is an effective model for predicting the distribution of traffic conditions given real-time observations. In the following, we evaluate how our routing algorithms can utilize a GPDCM for effective non-myopic routing under uncertainty.

5.2 Routing Validation

We now evaluate the effectiveness of our GP-UCB and PCR routing approaches for adaptive collective routing under uncertainty. We compare against both conventional baselines as well an omniscient lower bound:

- **Static Routing**: Each vehicle is routed (via Dijkstra's) on GPDCM prior mean. Routes are not updated as vehicles collect real-time observations.
- **Myopic Routing**: Each vehicle is routed (via Dijkstra's) on GPDCM posterior mean. Routes update in response to real-time observations, but the exploration value is not considered (hence myopic).[13]
- **Omniscient Routing**: Each vehicle is routed (via Dijkstra's) according to **perfect information** of traffic conditions (and thus precluding the need to explore). Omniscient routing is an idealized approach that can only run during simulation, and can bound the performance of any other approach i.e., serve as a competitive analysis).

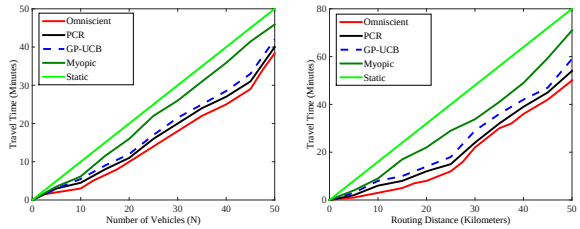
All approaches utilize the same GPDCM for modeling travel times, as well as the same routing algorithm (e.g., Dijkstra). More sophisticated routing algorithms can be used as well; however we defer such an evaluation to future work since the primary goal here is to evaluate the benefits of incorporating exploration into the routing utility function δU.

We run all routing approaches via simulation using congestion conditions mined from the same day in our

12. Other options include the Berk-Jones test, the score test and their integrated versions [19], [20].

13. Myopic routing is essentially how conventional adaptive routing approaches behave (cf. [21]).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering



(a) Travel time versus number vehicles to be routed (b) Travel time versus routing distance

Fig. 6. Travel time versus (a) number of vehicles to be routed (b) routing distance. PCR performs nearly as well as the omniscient lower bound.

historical data [6],[14] and assign each vehicle a unique source and destination pair (A, B).[15] We also evaluate multi-destination routing scenarios in Section 5.2.7.

5.2.1 Main Results

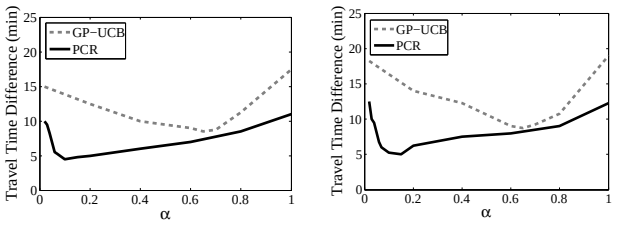
Figure 6(a) compares the total travel time with respect to the number of vehicles. We observe that the GP-UCB and PCR both significantly outperform Static and Myopic routing, with PCR nearly matching the performance of Omniscient routing. Figure 6(b) shows a comparison of total travel time with respect to the routing distance. We again observe both GP-UCB and PCR performing near-optimally relative to Omniscient routing. PCR offers about a 50% relative reduction in residual routing time between GP-UCB and Omniscient routing.

The main difference between GP-UCB and PCR is the different definition of information gain to characterize the value of exploration. GP-UCB uses a uniform information gain across the entire GP congest model (Eq. (15)), whereas PCR only evaluates information gain for road segments that appear in canonical routes (Eq. (21)).

These results suggest that both GP-UCB and PCR are effective at balancing the exploration/exploitation trade-off in order to quickly identify the best possible routing paths for the majority of the vehicles in the system. In the following, we present additional empirical evaluations, including parameter tuning, different ways of trading off between performance and computational cost, a preliminary field study, and simulation results for the multi-destination routing problem.

5.2.2 Parameter Tuning

We set the α parameter for both GP-UCB and PCR using a validation set of 30 routing tasks. In practice, we found that both GP-UCB and PCR are mildly sensitive to the choice of α, although parameter tuning is fairly straightforward. We tuned using the mean travel time difference versus Omniscient routing, shown in Figure 7(a). For completeness, Figure 7(b) shows the the max travel time



(a) Mean Travel Time Difference (b) Max Travel Time Difference

Fig. 7. Showing the effect of α on performance. Both plots depict the difference between GP-UCB and PCR versus Omniscient routing. (a) shows the mean difference. (b) shows the max difference.

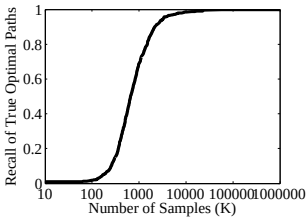


Fig. 8. Recall (or coverage) of the optimal paths with respect to number of samples M.

difference. For our experiments, we set $\alpha = 0.65$ for GP-UCB and $\alpha = 0.1$ for PCR.

Another design decision is the number of samples M for PCR (Line 4 in Algorithm 3). We would like to choose M sufficiently large so that all canonical paths are covered. Figure 8 shows the collective recall (or coverage) of the canonical paths of 50 vehicles. In our experiments, we chose $M = 5000$.[16] We will also empirically evaluate the trade-off of using smaller M (and thus reducing computational cost) versus routing performance.

5.2.3 How Does the Size of Historical Data S Impact Performance?

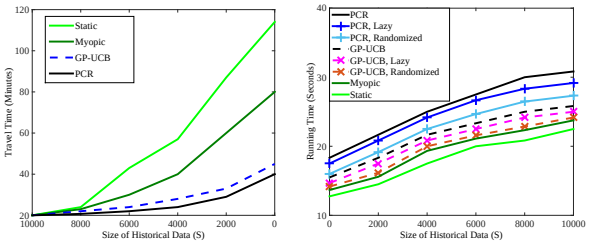
Our use of non-parametric mean μ and kernel k functions leads to more computationally expensive posterior inference as the training data S grows. Figure 9 shows how the total travel time and computation cost (for 20 vehicles) vary with the size of S. We see that GP-UCB and PCR are relatively robust to using smaller S (which leads to less reliable estimates of μ and k), which can lead to significant computational and storage savings for relatively small sacrifices in performance. We observe that both GP-UCB and PCR are computationally efficient, with GP-UCB being about 25% more efficient than PCR. In fact, GP-UCB matches the computational cost of Myopic routing (which is widely deployed). These results suggest that our approach is viable for real-time adaptive routing, since more optimized implementations can likely reduce the overhead of computing routes to be

14. We compute the "crowdedness" metric [6] for each road, which is essentially the complement of the travel speed CDF (i.e., crowdedness of 0.9 means the road is at its 10th percentile traveling speed).

15. Each vehicle was assigned a unique routing task selected from a large set of real routing tasks mined from our traffic datasets [6].

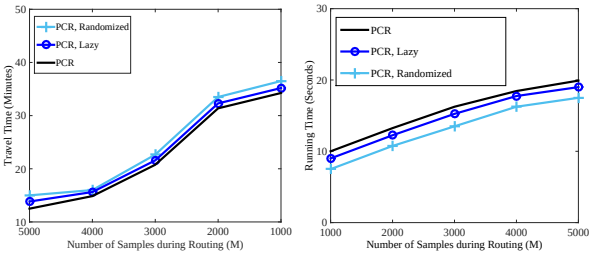
16. Note that sampling is efficient (due to the GPDCM allowing sampling in closed form) and routing is efficient (due to it being a modified Dijkstra's algorithm). Thus choosing $M = 5000$ still allows Algorithm 3 to be run efficiently.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering



(a) Travel time versus size of historical data S (b) Computational cost versus the size of historical data S

Fig. 9. Showing how travel time and computational cost vary with the size of the historical data S.



(a) Travel time versus the number of samples M (b) Computational cost versus the number of samples M

Fig. 10. Showing how travel time and computational cost vary with the number of samples M when running PCR.

negligible compared to the actual routing intervals (e.g., the routes are updated every 2 minutes).

5.2.4 How Does the Number of Samples M in PCR Impact Performance?

Another way to trade off performance with efficiency for PCR is to reduce the posterior samples M. Figure 10 shows how the total travel time and computational cost vary with M. We observe that PCR can tolerate a small degree of undersampling of traffic scenarios.

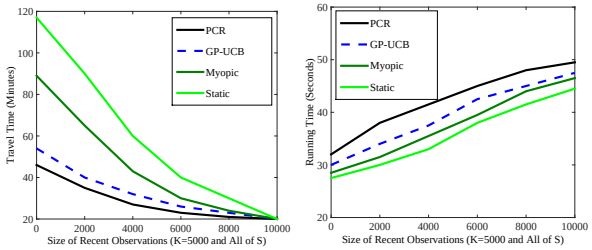
5.2.5 How Does the Number of Recent Observations Y Impact Performance?

We next evaluate the impact from varying the amount of recent observations Y that we keep for computing the GPDCM posterior (smaller Y can lead to less reliable GPDCM posterior distributions). Figure 11 shows this evaluation, and we see that both GP-UCB and PCR can tolerate a significant reduction in Y.

5.2.6 Preliminary Field Study

We conducted a preliminary field study using twelve vehicles in City 1.[17] Each routing trial comprises two groups of six cooperative vehicles, with each vehicle being assigned identical source and destination locations. The first group employs a version of Myopic routing, where the drivers are instructed to route according to their best knowledge and can communicate with each other. The second group employs a restricted version of PCR where three drivers are routed to explore three

17. These vehicles are identical to those captured in the GPS data.



(a) Travel time versus size of recent observations Y (b) Computational cost versus size of recent observations Y

Fig. 11. Showing how travel time and computational cost vary with the number of recent observations Y.

TABLE 1
Travel Time (minutes) from Field Study

Trial	Myopic	PCR	Improvement
1	28.0	22.3	25.7%
2	36.2	27.6	31.2%
3	12.7	11.2	13.4%
4	51.0	25.7	42.9%
5	32.5	24.9	30.6%

canonical routes, and the other three are routed according to updated knowledge.

Table 1 shows the results from five trials. We observe that PCR routing consistently outperforms Myopic routing (30% median reduction in travel time), which lends evidence that PCR can be broadly deployed to improve routing performance for larger fleets of cooperative vehicles. The improvement in Trial 3 is small, which is due to low congestion during that time.

5.2.7 Routing in Multiple Destinations

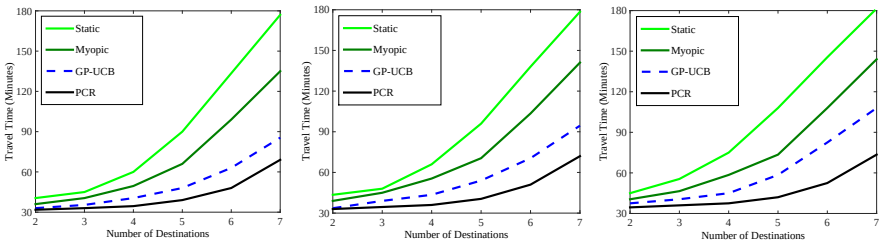
We finally evaluate the multi-destination routing scenario, where each vehicle must visit a set of destinations rather than a single one (i.e., Traveling Salesperson). We considered three different routing heuristics, divide and conquer (DC) [22], nearest neighbor (NN) [23], and pairwise exchange (PE) [24]. The destinations were generated based on the most frequently visited destinations for the vehicles in the historical dataset. Figure 12 shows the results for 100 randomly selected vehicles. We see that PCR consistently outperforms the other approaches across all routing heuristics. These results suggest that our approach can be applied to more complex tasks, so long as an appropriate routing algorithm is used.

6 Discussion

Our approach is related to other approaches that first sample from a probabilistic POMDP and then optimize for the resulting empirical distribution [13], [14]. One important theoretical question is what number of samples M guarantees good performance within one iteration.

Another theoretical question is whether one can prove guarantees for the entire execution of GP-UCB and PCR over all iterations. Such analysis may rely on using variance reduction to bound the performance gap between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2411278, IEEE Transactions on Knowledge and Data Engineering



(a) Divide and conquer (b) Nearest neighbor (c) Pairwise exchange

Fig. 12. Routing in multiple destinations which is solved in three popular heuristic and approximation algorithms.

omniscient routing, which is related to analysis for GP-UCB in unstructured settings [3], [9]. Two important differences between [3], [9] and our setting are that (1) uncertainty reduction decays over time (i.e., past observations have low value) and (2) we must optimize over a structured action space (i.e., all possible routings).

From a game-theoretic perspective, it would be interesting to consider cases where vehicles were not assumed to cooperative, but rather could be incentive to cooperate. One possibility is to properly price the value of exploration.

7 Related Work

Traffic data analysis: Traffic modeling is a very diverse research area, which is primarily due to there being a large variety of measurement types (e.g., traffic cameras, GPS traces) as well as modeling goals. Our work is most closely related to the area of congestion estimation.

Congestion or traffic estimation has been studied using a variety of mathematical tools, ranging from flow patterns [25], to per-individual statistical models [26], to Markov chain forecasting [27], to path oracles for spatial networks [28], and to shortest path and distance queries on road networks [29]. The two main types of measurement data are GPS or low-bandwidth cellular updates which are associated with individual vehicles [30], [31], and static traffic cameras which are associated with known location [32].

In contrast to most prior work, our goal is to derive a holistic generative probabilistic model of dynamic traffic conditions. Since our goal is to incorporate such a traffic model into a real-time routing algorithm, we require our model to not only accurately capture the distribution of future traffic scenarios (given real-time observations), but also be sufficiently fast to evaluate. For measurement, we use GPS traces collected from thousands of taxis from two large Asian cities (see Section 5).

Other work on traffic modeling have studied phenomenon such as traffic accidents or other outlier activity [33], [34], evaluating the overall health of a road network design [35], enhancing driving directions with taxi drivers' intelligence [36] and providing dynamic taxi ridesharing service [37].

Adaptive routing: Although there have been some work on personalized or adaptive routing or time dependent road networks [38], [39] based on historical and real-time traffic conditions, prior work did not model the benefits of real-time exploration, and thus must resort to myopic routing [21], [11], [1].

The routing problem we study is an instance of the general problem of decision-making under uncertainty. Such problems are typically cast as partially observable Markov decision processes (POMDPs) [40], where the world (i.e., road congestion conditions) is assumed to behave according to Markovian dynamics, and actions (i.e., routing) reveal partial observations regarding the state of the world (i.e., local congestion measurements).

We cast our problem as a POMDP with a continuous state space modeled using a Gaussian Process. In contrast to previous work on continuous POMDPs (e.g., [41]), we are focused on large structured action spaces (i.e., all possible routings of the cooperative vehicles).

There are three common types of approaches for solving discrete POMDPs with large action spaces. The first is to simplify the problem using canonical or macro actions [42]. After identifying the best plan consisting of macro actions, the planner can optionally refine at finer granularities. The second type is to always plan at finer granularities and use pruning approaches to avoid enumerating the full exponentially large decision tree (e.g., Monte-Carlo tree search [43]). The third, more generic, type of approach is to sample from a generative distribution of the POMDP and then find a good plan that optimizes for this empirical distribution [44], [45], [13], [14].[18] The most general approaches are called stochastic planning with recourse [44], [45], [46], which is a general framework[19] where the goal is to compute a good initial plan while also allowing adaptive behavior, or recourse, during operation. Stochastic planning with recourse approaches are often focused on planning for relatively short time horizons due to efficiency reasons. Conversely, approaches such as PEGASUS [14] plan for longer time horizons but typically optimize over restricted classes of policies.

Our second proposed algorithm can be viewed as a stochastic planning approach (over a continuous state space) that utilizes canonical actions. In contrast to previous work, our approach optimizes both for the *long-*

18. It can be shown that a finite number of samples is sufficient for any planning problem (cf. [13], [14]).

19. It can be shown that all MDP planning problems can be instantiated as a (large) stochastic planning with recourse problem (cf. Chapter 4 of [47]).

term plan and over the *full action space of all possible routes*. To control the complexity of long-term planning, we leverage the observation that typically only a few routes can be optimal for any vehicle – we call these the canonical routes (see Section 4.3). Our approach efficiently trades off between exploration and exploitation by choosing routes that optimize the combination of expected travel time (exploitation) and uncertainty reduction (exploration to determine which canonical route is the best) – this bears affinity to algorithms for Gaussian Process bandit optimization [3], [9].

From a combinatorial routing perspective, our work is focused on a relatively simple setting where each vehicle has a pre-defined destination. More complicated settings typically assume that each vehicle is "exchangeable" and can be routed to any target destination (cf. [48]). This leads to a challenging combinatorial optimization problem even without considering uncertainty. We instead focus on the complementary problem of how best to collectively route each vehicle to their pre-assigned destination while accounting for uncertain traffic conditions.

Our approach is also related to work on adaptive path planning in robotics [2] and the m-vehicle dynamic traveling repairman problem [2]. One important difference is that their goal is to maximize some other notion of utility (e.g., number of interesting events discovered along a path) for a given budget (e.g, paths no longer than 100 meters), whereas our goal is to minimize the collective travel time.

Another line of related work is motivated from the network packet routing problem [49]. A key difference is that [49] is focused on the repeated games problem for a single routing problem at a time, whereas we seek to solve a single-shot multi-agent routing problem.

8 Conclusion

We proposed a Gaussian Process Dynamic Congestion Model to capture both the dynamics and the uncertainty of congestion conditions, and the routing algorithms to balance the exploration/exploitation trade-off for entire fleets of vehicles. We conducted extensive evaluations using GPS traces collected from two large Asian cities. Our results show that our GPDCM effectively predicts the distribution of future congestion conditions, and that our routing algorithms can achieve near-optimal performance relative to omniscient routing. We also conducted a preliminary field study, where we found our approach to significantly outperform conventional myopic routing.

Acknowledgements. This research was supported by Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Carnegie Mellon University's Technologies for Safe and Efficient Transportation, The National USDOT University Transportation Center for Safety (T-SET UTC) sponsored by the US Department of Transportation, National Basic Research Program of China (973 Program): 2015CB352400, and Basic Research Program of Shenzhen: JCYJ20140610152828686. Yisong Yue was also supported by ONR (PECASE) N000141010672 and ONR Young Investigator Program N00014-08-1-0752. Siyuan Liu would acknowledge Google Faculty Research Award. The authors thank Emma Brunskill, Geoff Gordon, Sue Ann Hong, Lavanya Marla, and Lionel Ni for valuable discussions and support.

9 Supplementary Proofs

Proof of Lemma 1: $IG(\Psi; f)$ Eq. (7) is known to be monotone submodular in Ψ [3], and monotone functions are also quasi-monotone. We can see that $c_n(\Psi \cap P_n)$ Eq. (12) is also monotone in Ψ except when $|\Psi \cap P_n| = 0$. We next note that the set maximum function $\max_{v \in V} v$ is submodular in the input set V [50], which implies that c_n is submodular. Thus, both components of Eq. (11) are quasi-monotone submodular, which implies that Eq. (11) is quasi-monotone submodular. □

Proof of Lemma 2: We adapt from the proof of Lemma 1 in [17]. Let $\Psi^* = \{\vec{r}^*_1, \ldots, \vec{r}^*_n\}$. We can write $U(\Psi^*)$ as

$$\leq U(\Psi_N \cup \Psi^*) \tag{24}$$

$$= U(\Psi_N) + \sum_{i=1}^{N} \left(U(\Psi_N \cup \Psi^*_{1:i}) - U(\Psi_N \cup \Psi^*_{1:i-1}) \right)$$

$$\leq U(\Psi_N) + \sum_{i=1}^{N} \left(U(\Psi_{i-1} \cup \{\vec{r}^*_i\}) - U(\Psi_{i-1}) - \epsilon_i \right) \tag{25}$$

$$\leq U(\Psi_N) + \sum_{i=1}^{N} \left(\frac{1}{\beta} \left(U(\Psi_i) - U(\Psi_{i-1}) \right) - \epsilon_i \right) \tag{26}$$

$$= \frac{\beta + 1}{\beta} U(\Psi_N) - \sum_{i=1}^{N} \epsilon_i$$

where Eq. (24) follows from quasi-monotonicity of U and the fact that $|\Psi^* \cap P_n| = 1$, Eq. (25) follows from approximate submodularity, and Eq. (26) follows from the fact that the greedy algorithm is locally β-approximate optimal. The result naturally follows. □

Proof of Lemma 3: Alg. 2 utilizes Eq. (9) and Eq. (10) to approximate Eq. (11). We now show that this approximation preserves the properties of Eq. (11) such that Alg. 2 using Alg. 1 and Alg. 2 using Alg. 4 return identical solutions. It suffices to show that, at each iteration j, the route $\vec{r}_{n^*}$ that has optimal $\delta U(\vec{r}_{n^*}|\Psi_{j-1})$ is only sorted behind stale entries (i.e. those with $\tau < j$) in Q. We first note that for all $\vec{r}$, $\delta(\vec{r}|\Psi)$ can only decrease w.r.t. Ψ. Suppose that some entry $(n, v_n, \tau_n, \vec{r}_n) \in Q$ is not stale (i.e., $\tau_n = j$) and is ranked higher than the optimal entry $(n^*, v_{n^*}, \tau_{n^*}, \vec{r}_{n^*})$, i.e., $v_n > v_{n^*}$. Then we have a contradiction because either the optimal entry is not stale in which case we have $v_n < v_{n^*}$, or the optimal entry is stale but then the stored v_{n^*} value is even higher than the non-stale value of v_{n^*} so we must have $v_n < v_{n^*}$. □

References

[1] R. J. Gutman, "Reach-based routing: A new approach to shortest path algorithms optimized for road networks," in *Proc. of ALENEX/ANALC*, 2004, pp. 100–111.

[2] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on Automatic Control*, pp. 1259–1274, 2011.
[3] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. of ICML*, 2010.
[4] S. Liu, Y. Yue, and R. Krishnan, "Adaptive collective routing using gaussian process dynamic congestion models," in *Proc. of KDD*, 2013.
[5] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 1.
[6] S. Liu, Y. Liu, L. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *Proc. of KDD*, 2010.
[7] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, pp. 117–129, 1978.
[8] C. Guestrin, A. Krause, and A. Singh, "Near-optimal sensor placements in gaussian processes," in *Proc. of ICML*, 2005.
[9] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization," in *Proc. of NIPS*, 2011.
[10] C. Chekuri and M. Pál, "A recursive greedy algorithm for walks in directed graphs," in *Proc. of FOCS*, 2005.
[11] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," in *Proc. of SODA*, 2008.
[12] A. Singh, A. Krause, and W. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Proc. of IJCAI*, 2009.
[13] M. Kearns, Y. Mansour, and A. Ng, "Approximate planning in large pomdps via reusable trajectories," in *Proc. of NIPS*, 1999.
[14] A. Y. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Proc. of UAI*, 2000.
[15] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, pp. 265–294, 1978.
[16] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint," *Integer Programming and Combinatorial Optimization*, pp. 182–196, 2007.
[17] D. Golovin, A. Krause, and M. Streeter, "Online learning of assignments that maximize submodular functions," *CoRR*, vol. abs/0t08.0772, 2009.
[18] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. of KDD*, 2007.
[19] A. Koning and L. Peng, "Goodness-of-fit tests for a heavy tailed distribution," Erasmus University Rotterdam, Econometric Institute, Econometric Institute Report, Nov 2005.
[20] M. C. Bryson, "Heavy-tailed distributions: Properties and tests," *Technometrics*, vol. 16, no. 1, 1974.
[21] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. of KDD*, 2011.
[22] Z. Yang, "A divide and conquer algorithm for multi-stop routing," in *Proc. of 2013 Washington GIS Conference*, 2013.
[23] S. S. Ray, S. Bandyopadhyay, and S. K. Pal, "Genetic operators for combinatorial optimization in tsp and microarray gene ordering," *Applied Intelligence*, pp. 183–195, 2007.
[24] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European Journal of Operational Research*, pp. 106–130, 2000.
[25] X. Li, J. Han, J.-G. Lee, and H. Gonzalez, "Traffic density-based discovery of hot routes in road networks," in *Proc. of SSTD*, 2007.
[26] H.-P. Kriegel, M. Renz, M. Schubert, and A. Zuefle, "Statistical density prediction in traffic networks," in *Proc. of IEEE ICDM*, 2007.
[27] K. Sirvio and J. Hollmén, "Spatio-temporal road condition forecasting with markov chains and artificial neural networks," in *Proc. of HAIS*, 2008.
[28] J. Sankaranarayanan, H. Samet, and H. Alborzi, "Path oracles for spatial networks," *PVLDB*, pp. 1210–1221, 2009.
[29] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, "Shortest path and distance queries on road networks: towards bridging theory and practice," in *Proc. of SIGMOD*, 2013, pp. 857–868.
[30] J. Yoon, B. Noble, and M. Liu, "Surface street traffic estimation," in *Proc. of MobiSys*, 2007.
[31] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi gps traces," in *Proc. of Pervasive*, 2012.
[32] J. Bacon, A. I. Bejan, A. R. Beresford, D. Evans, R. J. Gibbens, and K. Moody, "Dependable and historic computing," 2011, ch. Using real-time road traffic data to evaluate congestion.
[33] C. Benjamin, "Detecting the onset of congestion rapidly with existing traffic detectors," *Transportation Research*, 2003.
[34] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *Proc. of KDD*, 2011.
[35] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proc. of UbiComp*, 2011.
[36] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 220–232, 2013.
[37] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. of ICDE*, 2013.
[38] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proc. of AAAI*, 2013, pp. –1–1.
[39] G. V. Batz, D. Delling, P. Sanders, and C. Vetter, "Time-dependent contraction hierarchies," in *Proc. of ALENEX*, 2009, pp. 97–105.
[40] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1995.
[41] P. Dallaire, C. Besse, S. Ross, and B. Chaib-draa, "Bayesian reinforcement learning in continuous pomdps with gaussian processes," in *Proc. of IROS*, 2009.
[42] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," *Journal of Machine Learning Research*, vol. 40, pp. 523–570, 2011.
[43] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Proc. of NIPS*, 2010.
[44] J. Higle and S. Sen, "Stochastic decomposition: An algorithm for two-stage linear programs with recourse," *Mathematics of operations research*, vol. 16, no. 3, pp. 650–669, 1991.
[45] J. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Verlag, 1997.
[46] G. Polychronopoulos and J. Ttsitsiklis, "Stochastic shortest path problems with recourse," *NETWORKS*, vol. 27, pp. 133–143, 1996.
[47] S. A. Hong, "Distributed market-based algorithms for multi-agent planning with shared resources," Ph.D. dissertation, Carnegie Mellon University, 2013.
[48] J. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J. Sormany, "New heuristics for the vehicle routing problem," *Logistics systems: design and optimization*, pp. 279–297, 2005.
[49] B. Awerbuch and R. Kleinberg, "Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches," in *Proc. of STOC*, 2004.
[50] A. Guillory and J. Bilmes, "Interactive submodular set cover," in *Proc. of ICML*, 2010.

Siyuan Liu is a Research Scientist at Carnegie Mellon University. He received his first Ph.D. degree from Department of Computer Science and Engineering at Hong Kong University of Science and Technology in 2011, and the second Ph.D. degree from University of Chinese Academy of Sciences in 2014. His research interests include mobile data analytics and heterogeneous social networks mining.

Yisong Yue is an Assistant Professor at the California Institute of Technology, and received his Ph.D. in Computer Science from Cornell University. His research interests lie primarily in the theory and application of statistical machine learning, with a particular focus on interactive learning systems, structured prediction, and learning with humans in the loop.

Ramayya Krishnan is the W. W. Cooper and Ruth F. Cooper Professor of Management Science and Information Systems and Dean of the Heinz College at Carnegie Mellon University. He has a Ph.D. from the University of Texas at Austin. His current research investigates risk management in business process design and information security, large scale social network analysis, and the design of policies that take into account the competing needs of promoting data access and protecting privacy.