The Online Shortest Path Problem: Learning Travel Times Using A Multi-Armed Bandit Framework

Tomás Lagos[1], Ramón Auad[2], and Felipe Lagos[3]

[1]Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA
[2]Department of Industrial Engineering, Universidad Católica del Norte, Antofagasta, Chile
[3]Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Santiago de Chile 7941169, Chile

Abstract

In the age of e-commerce, many logistic companies must operate in large road networks without accurate knowledge of travel times for their specific fleet of vehicles. Moreover, millions of dollars are spent on routing services that do not accurately capture the specific characteristics of the companies' drivers and the types of vehicles they must use. In this work, we consider a logistic operator that has limited information about travel times in a network, and who seeks to find the optimal expected shortest path of origin-destination pairs. We model this problem as an Online Shortest Path Problem, common to many last mile routing settings: given a graph whose arcs' travel times are stochastic and follow an unknown distribution, the planner seeks to find a vehicle route of minimum travel time from an origin to a destination. We assume the planner progressively collects information on the travel conditions as the drivers keep serving requests. Inspired by combinatorial multi-armed bandit and Kriging literature, we propose three methods with different characteristics that effectively learn the optimal shortest path and illustrate the practical advantages of incorporating spatial correlation in the learning process. Our approach balances the trade-off of exploration (better estimates for unexplored arcs) vs exploitation (executing the minimum expected time path) by using the Thompson Sampling algorithm. In each iteration, our algorithm executes the path minimizing the expected time with data from a posterior distribution of the arcs' speeds. We carry out a computational study comprising two settings: a set of four artificial instances and a real-life case study. The case study uses empirical data of taxis in the 17km-radius area of the center of Beijing, completely enveloping Beijing's "5th Ring Road". In both settings, we observe that our algorithms efficiently and effectively balance the exploration-exploitation trade-off.

Keywords: Last-Mile Logistics; Machine Learning; Multi-Armed Bandits; Thompson Sampling; Online Shortest Path; Kriging.

1 Introduction

Urban logistics continuously and deeply impacts our society (Cattaruzza et al. 2017). Over the past decade, factors such as the steady population growth, the importance of fast transportation in supply chains, and the interest in sustainability have reshaped how logistics works; moreover, the advent of the Internet and the accelerated adoption of mobile devices have driven new trends, such as the boom of e-commerce and the sharing economy, in so allowing the creation of new business models, *e.g.*, ultra-fast parcel delivery, ride-hailing, and meal delivery (Savelsbergh and Van Woensel 2016, Lafkihi et al. 2019, Al Mashalah et al. 2022).

Specifically, e-commerce sales have experienced double-digit yearly growth over the past years, with an explosive adoption boost in 2020 due to the COVID-19 pandemic (Insider Intelligence 2021, Euromonitor 2021). Between 2018 and 2020, US grocery e-commerce sales percentage relative to US total grocery sales has grown from 2.7% to 10.2%, and it is projected to reach 21.5% by 2025, estimated in $250 billion, *i.e.*, 60% over pre-pandemic estimates (Mercatus and Incisiv 2021); moreover, worldwide retail e-commerce sales have increased from $1.3 trillion in 2014 to over $5.5 trillion in 2022, with sales expectations of $8.1 trillion in 2026 (Statista 2022).

That exist considerable potential gains in e-commerce is beyond doubt. Yet, the sustained rise in adoption of e-commerce, the steadily increasing service expectations from customers, and the stiff competition between logistics service providers lead to new and complex logistics challenges (Savelsbergh and Van Woensel 2016). Many of these challenges are encountered in last-mile delivery, *i.e.*, the last segment of the logistics service where goods are delivered at the customer location; last-mile logistics is typically considered to be the least efficient and most expensive portion of the supply chain, with associated costs amounting up to 60% of the total logistics costs (Vanelslander et al. 2013, Lafkihi et al. 2019). Providing a fast, reliable, and efficient service are among the key factors for e-commerce success, and attaining these greatly depends on how transportation decisions are made (Mangiaracina et al. 2019).

In e-commerce logistics, customers typically place requests ahead of time and select a delivery location, and a decision maker coordinates the preparation and delivery processes. As requests become ready for dispatch, the decision maker decides which routes to follow to complete their deliveries. These decisions are executed using a fleet of drivers, which may be composed of a combination of direct employees and independent contractors. One of the challenges encountered when making these decisions is incorporating travel time uncertainty. Network travel times can be highly variable (*e.g.*, due to traffic congestion, and road closures); and failing at accounting for these effects may translate into poor routing decisions and unreliable service quality, potentially deteriorating the business reputation in the long-term. Conversely, travel times information allows decision makers to make informed routing decisions, which ultimately can provide substantial benefits to their business, such as increased drivers' productivity and reduced fuel consumption by selecting shorter routes, and faster and more reliable service for customers by choosing less congested routes (Szeto et al. 2013, Manufacturing and Logistics IT 2020, Kou et al. 2022).

Although in practice there exist multiple navigation tools that can provide real-time

travel time estimations, these solutions come with critical limitations for companies that have specialized operations. First, they provide time estimations that are primarily based on historical data on conventional users whose driving behaviors very likely differ significantly from a company's specific operators. Second, these solutions are typically designed for standard vehicles (*e.g.*, cars) whose eligible roads and travel times differ from more specialized vehicles (*e.g.*, , cars may swiftly traverse through a passageway that a delivery truck cannot due to size or weight restrictions). Third, these solutions do not scale well with the size of the system (*e.g.*, the number of drivers and the number of requests) as most of them are not designed to construct efficient multi-stop routes but are designed to calculate simple origin-destination (O-D) routes. Lastly, most specialized solutions are commercial and their consistent usage may raise considerable operational costs, especially as business operations grow (Fu et al. 2020, Manufacturing and Logistics IT 2020).

Motivated by this, in this paper we propose a learning-based framework that allows transportation companies to learn over time which are the best routes to traverse as they operate their own fleet. We consider a setting where the decision maker coordinates the operations of a fleet of drivers across a transportation network and has limited or no initial information about the travel times of the network's arcs; as they make routing decisions and collect travel time information when serving customer requests, they can improve their predictions of the network travel conditions. In the literature, this setting is known as the Online Shortest Path Problem (OSPP), a problem common to many last mile routing settings: given a time period (*e.g.*, a day of operations) and a graph whose arcs' travel times are stochastic, the decision maker must find a vehicle route from a source to a destination to minimize the total travel time. The travel times distribution parameters are unknown to the decision maker, but they have the ability to observe the realized travel times when one of its vehicles completes the route and use this gained knowledge to update the information about the graph's travel times. At every time period, the decision maker faces the well-known "exploration-exploitation trade-off": if the decision maker only considers the information obtained up to the current time period and executes the route with the shortest expected time, they might miss alternative, not yet-visited routes with shorter expected travel time; if instead, they explore new routes too frequently, the decision maker learns the distribution parameters but the routing operation becomes inefficient due to the possibility of finding sub-optimal alternative routes. We formulate this problem as a Combinatorial Multi-Armed Bandit (CMAB) problem (Cesa-Bianchi and Lugosi 2012), an extension of the more traditional Multi-Armed Bandit (MAB) setting (Robbins 1952). To address this trade-off, our framework adapts Thompson Sampling (Thompson 1933), an efficient machine learning algorithm that effectively balances investment to acquire new information that may improve future performance, and exploitation of current knowledge to maximize immediate performance (Chapelle and Li 2011). Moreover, we propose two extensions that leverage the concept of Kriging (Krige 1951) to incorporate spatial correlation between the travel time of nearby arcs, a phenomenon that is common in practice but generally disregarded in the literature to facilitate analysis. We conduct an exhaustive experimental study comprising synthetic and real-world data to empirically prove the effectiveness of our methods in finding

travel time-efficient paths.

Overall, the proposed approach offers multiple practical advantages, such as (i) except for the cost of exploring the network, it is cost-free for the retailer; (ii) estimations are based purely on data specific to its own fleet of drivers; (iii) its accuracy improves as the retailer keeps operating its delivery network and gathers new data; and (iv) it offers adaptability to potential changes in the travel conditions throughout the network. Furthermore, this work adds to the developing line of research that combines machine learning with logistics and illustrates the potential of how learning methods can be leveraged to effectively solve more complex logistics problems.

We summarize the contributions of this paper as follows:

- To the best of our knowledge, we present the first study of the OSPP in a setting that considers a general network topology, and without assumptions on the true distribution of rewards.
- We propose an algorithm that adapts Thompson Sampling to iteratively learn cost-effective paths across an operating network.
- We further propose two algorithms that build upon our first, which combine Thompson Sampling and Kriging to incorporate the spatial correlation between the travel times of close-by arcs, effectively speeding up the learning process.
- We demonstrate the capabilities of our methods by conducting a set of experiments that compare their performance against a benchmark policy inspired by the existing literature; and analyze the sensitivity of the performance of our methods with respect to the selection of their parameters.

The rest of the paper is organized as follows. Section 2 reviews existing literature pertinent to our work. Section 3 provides a mathematical definition of the OSPP and its corresponding CMAB formulation. Section 4 presents a Thompson Sampling-based learning algorithm to solve the OSPP, and proposes two extensions to handle the OSPP with spatially-correlated travel times. Section 5 presents an experimental study obtained from simulations using synthetic and real-world data. Section 6 summarizes the findings of the paper and proposes directions for future research.

2 Literature Review

The classical MAB problem (Robbins 1952, Lai et al. 1985) consists of a sequential decision problem where at each decision epoch (typically referred to as *rounds*), a decision maker selects an arm (decision) to pull from a pool of available arms. Based on the selected arm, the decision maker receives a corresponding reward whose distribution is unknown. The evaluation of the performance of a given policy is typically done by means of its *pseudo-regret*, defined as the difference between the expected total reward obtained from following the policy, and the optimal expected reward attainable if all arms' reward distribution were

known by the decision maker. Solving a MAB problem consists of finding an optimal policy that minimizes the regret. The MAB problems have gained considerable attention over the last few years, with recent references describing in detail the problem, its variants and algorithms, such as the books Lattimore and Szepesvári (2020), Slivkins (2019) and the monograph Bubeck and Cesa-Bianchi (2012). This problem has arisen in multiple domains (Bouneffouf et al. 2020), such as online advertising (Chakrabarti et al. 2008), recommendation services (Zhou et al. 2017), revenue management (Ferreira et al. 2018), and even, in Artificial Intelligence applications such as AlphaGo (Silver et al. 2016).

One of the variants of the MAB is the CMAB problem (Chen et al. 2013, Cesa-Bianchi and Lugosi 2012). In contrast to classical MAB problems, in CMAB settings the decision maker pulls, at each round, a subset of the available arms (*i.e.*, in the literature this subset is called *super-arm*). The subsets that can be selected at each round are commonly required to satisfy a set of constraints, *e.g.*, the set of selected arms cannot exceed a given budget (Nuara et al. 2022), and the goal is to determine the optimal policy that minimizes the regret. The OSPP corresponds to a CMAB problem (Chen et al. 2013), wherein the super-arm selected at every round consists of a set of arcs in the network that describe a path connecting a given O-D pair, and the associated reward corresponds to the length of the path.

The OSPP has been studied in different contexts. In the transportation domain, it has shown up when solving reliable routing in stochastic network problems (Khani 2019). In this problem, an agent must travel from an origin to a destination using a routing policy with the minimum expected time on a graph. Information about travel times is revealed as the agent travels along the arcs, and authors solve it using an online shortest path model. This work differs from ours in that the path is dynamically adapted as it is executed and in that there is no learning of the distribution of random variables. Instead, the objective is to find a policy for deciding how to update the routing plan. The OSPP we studied is more similar to the problem studied in Talebi et al. (2017); however, in this work, the routing problem is not for vehicles, but for packets transmission in wireless networks, where the packets are either transmitted or not (Bernoulli distributions) and where the challenge is to learn the arcs probabilities for packets transmission. The authors consider a stochastic semi-bandit combinatorial setting for the problem. Another recent work that is similar to ours is Chen et al. (2020), in which it is studied the problem of routing electric vehicles that must travel from an origin to a destination. The energy consumption of the vehicles is learned, considering a semi-bandit setting using an Upper Confidence Bound (UCB) policy. Zou et al. (2014), Liu and Zhao (2012) also study the OSPP for packet transmission in wireless networks. To the best of our knowledge, the OSPP for vehicle routing minimizing travel times has not been studied.

The most widely adopted approaches to solve MAB problems include the UCB (Auer et al. 2002) and Thompson Sampling (Thompson 1933). UCB-based algorithms can be very effective with proper selection of UCB, yet their performance is highly dependent on these bounds, and obtaining high-quality bounds without a closed-form posterior distribution can be computationally expensive (Russo and Van Roy 2014). By contrast, Thompson Sampling-based algorithms in general admit an efficient and straightforward implementation, provide

theoretical performance guarantees comparable to the UCB approach (Kaufmann et al. 2012, Agrawal and Goyal 2013), and often attain better empirical results (Chapelle and Li 2011). Due to these advantages, in this work, we adapt the combinatorial variant of Thompson Sampling to a shortest path setting to effectively learn travel times in a given transportation network. To solve CMAB problems, authors have proposed extensions of the UCB and Thompson Sampling methods from traditional MAB literature, namely Combinatorial UCB (Chen et al. 2013) and Combinatorial Thompson Sampling (Komiyama et al. 2015).

A distinctive characteristic of the OSPP is the fact that unlike most of the MAB settings found in the literature, travel times from arcs (*i.e.*, rewards from arms) that are close in proximity can be correlated, *e.g.*, due to traffic. To model this spatial element, we construct learning algorithms borrowing ideas from the Kriging literature (Krige 1951). Kriging is a spatial prediction method that seeks to infer (or *krige*) unknown values based on observations at nearby locations (Cressie 1990, 2015, Cressie and Johannesson 2008). Correspondingly, we formulate the OSPP as a Gaussian Process (GP) where rewards of arcs follow normal distributions that are spatially correlated with nearby arcs, and progressively learn the unknown parameters of arcs' distributions as we traverse paths across the network. Exploiting the spatial correlations allows us to deduce information about unfrequented arcs based on the visited paths and improve subsequent routing decisions, ultimately making the overall learning process more sample-efficient.

GP models are in general a powerful tool to model uncertainty over functions and are broadly used in the MAB literature (Rasmussen and Williams 2005, Srinivas et al. 2009, Hoffman et al. 2011, Chowdhury and Gopalan 2017). For example, in Nuara et al. (2022) the authors study the pay-per-click advertising with budget constraints problem over multiple channels, and formulate the problem as a combinatorial semi-bandit problem, in which the dependency of the number of clicks on the bid and daily budget is captured by a Gaussian process. Upper bounds for the regret and experimental evaluation of the algorithm are provided. To the best of our knowledge, algorithms combining a Kriging updating approach with Thompson Sampling have not been proposed and the posterior distribution updating strategies of our algorithms has not been studied.

Random travel times are usually considered in vehicle routing problems (Toth and Vigo 2014). In general, probability distributions with a positive skew and long tails are assumed, such as the Log-Normal, Gamma, Weibull, and Burr distributions (Ulmer 2017, Susilawati et al. 2013). The Log-Normal distribution is considered for different vehicle routing settings, such as routing on urban corridors or under congestion (Gómez et al. 2016), and it has been established that it is a distribution that accurately models travel times (Lecluyse et al. 2009, Chen et al. 2018, Srinivasan et al. 2014): it reflects that there is a minimum time required to cover the distance, after which the probability increases rapidly to a maximum, and then it decreases slowly with a long tail. Other studies supporting this distribution include Kharoufeh and Gautam (2004), Taniguchi et al. (2001).

3 Problem Formulation

This section formally defines the OSPP. We first provide a mathematical description of the OSPP setting and introduce necessary notation, and then formulate the OSPP as a CMAB problem.

3.1 Online Shortest Path Formulation

Let $\mathcal{G}$ be a connected and directed network with V nodes, indexed by $[V] \coloneqq \{1, \ldots, V\}$, and arcs indexed by $k \in [A] \coloneqq \{1, \ldots, A\}$. The planning horizon is $T \in \mathbb{Z}^+$, with time periods $t \in [T] \coloneqq \{1, \ldots, T\}$. At each time $t \in [T]$, a decision maker requires to travel from an origin location $o_t \in [V]$ to a destination $d_t \in [V]$, for which they select a *simple* path in the network P_t connecting both nodes. As a consequence of selecting path P_t, the decision maker incurs a (negative) reward equivalent to the total travel time required to traverse through the arcs of P_t at time $t \in [T]$. The goal of the decision maker at every time period $t \in [T]$ is to follow the path from o_t to d_t of minimum expected travel time. Such a path can in principle be determined by solving a standard shortest path problem. Let μ_k be the true expected travel time of arc $k \in [A]$, and let $\mu \coloneqq (\mu_k)_{k \in [A]}$ be the vector of expected travel times of arcs $[A]$; if w_k denotes a binary decision variable that takes the value 1 if and only if arc $k \in P_t \subseteq [A]$, and $\delta_v^-, \delta_v^+ \subseteq [A]$ are the sets of arcs inbound and outbound to node $v \in [V]$, respectively, then the shortest path optimization problem can be formulated as follows.

$$\mathrm{SP}(o_t, d_t, \mu) \coloneqq \min \quad \sum_{k \in [A]} \mu_k w_k \tag{1a}$$

$$s.t. \quad \sum_{k \in \delta_{o_t}^+} w_k = 1 \tag{1b}$$

$$\sum_{k \in \delta_{d_t}^-} w_k = 1 \tag{1c}$$

$$\sum_{k \in \delta_v^+} w_k - \sum_{k \in \delta_v^-} w_k = 0 \qquad \forall v \in [V] \setminus \{o_t, d_t\} \tag{1d}$$

$$w_k \in \{0, 1\} \qquad \forall k \in [A] \tag{1e}$$

Objective (1a) corresponds to the expected total travel time of the path selected by the decision maker at time t, which is modeled as the sum of the expected travel times $\mu_k, k \in [A]$, associated with each of the arcs in the path. Constraints (1b) and (1c) require that the path chosen starts at origin o_t and ends at destination d_t, respectively, while Constraints (1d) impose flow conservation at every node visited by the path. Finally, Constraints (1e) specify the binary nature of each decision variable $w_k, k \in [A]$. This problem can be efficiently solved using linear programming or dynamic programming, *e.g.*, Dijkstra's algorithm (Dijkstra 1959).

3.2 Formulation as a CMAB Problem

In practice, however, the decision maker does not have access to either the expectation or the realization of the travel times across the different arcs of $\mathcal{G}$ at each period, instead having to estimate them online. Achieving this requires that the decision maker faces the well-known exploration-exploitation dilemma: if they only consider the information obtained up to the current time period and execute the route with the shortest estimated expected time, then they might miss alternative routes whose expected times are better; if instead they explore new routes too frequently, they may learn more about the travel times distribution, albeit the routing operation may become inefficient. Hence, we formulate the OSPP as a CMAB problem. The word "combinatorial" references the fact that in the CMAB setting, the decision maker sequentially selects from a set of available options or *arms*, a subset or *superarm* that satisfies some combinatorial constraints. More precisely, we study a "semi-bandit" setting where once a superarm is selected, the decision maker observes only the payoffs corresponding to the arms contained in the selected superarm and gets the associated reward (as opposed to other settings where the decision maker observes the payoff of every arm, or only the superarm's total reward is observed after making a decision Audibert et al. (2011)). In the OSPP, each arm corresponds to an arc of the network, and at each time $t \in [T]$, a superarm consists of a set of arcs describing a path from the origin o_t to the destination d_t. The payoff of every arc is given by its travel time realization, and the reward of a path is the sum of the payoffs of the arcs defining it.

For $t \in [T]$, let $\mathcal{P}_t$ be the set of all simple directed paths connecting o_t to d_t. The decision maker learns over time which are the most efficient arcs to travel by choosing, at each period $t \in [T]$, a simple path (superarm) $P_t \in \mathcal{P}_t$. By choosing path P_t at time $t \in [T]$, the decision maker obtains a (negative) reward equivalent to the total travel time incurred when traversing the arcs in P_t. We denote the random variable corresponding to the travel time of arc $k \in [A]$ at time $t \in [T]$ by X_k^t, and the vector of random variables corresponding to the travel time of the arcs in $[A]$ by $X^t := (X_k^t)_{k\in[A]}$. Consequently, the negative reward of selecting path P_t is the random variable given by $\sum_{k\in p} X_k^t$. Furthermore, we assume that the random travel times X^t are jointly distributed with a cumulative distribution (CDF) $F(x;\theta)$, $x \in \mathbb{R}^A$, and where θ represents a parameter vector whose true values are unknown to the decision maker and takes values in the parameter space Θ, and with $\mu := E(X^t) \in \theta, \forall t \in [T]$.

To measure the quality of the decisions made by the decision maker, we compare the average travel time of the path P_t from o_t to d_t selected at time t by the decision maker against the shortest (*i.e.*, optimal) average travel time among all the paths in $\mathcal{P}_t$ (provided by a clairvoyant algorithm that has perfect knowledge of the expected travel times across the entire network). For each time period $t \in [T]$, let P_t^* be the path that yields the optimal average travel time; we define the *marginal pseudo-regret* at t as

$$\Delta_t := \sum_{k\in P_t} \mu_k - \sum_{k\in P_t^*} \mu_k \tag{2}$$

and the *total pseudo-regret* as

$$\mathcal{R}_t \coloneqq \sum_{t' \in [t]} \Delta_{t'} \tag{3}$$

Lastly, we define the *time-average pseudo-regret* at time $t \in [T]$ as the average marginal regret over all time periods up to t, *i.e.*, $\frac{\mathcal{R}_t}{t}$. The goal is to design an online routing policy that minimizes the time-average pseudo-regret at the last time period, $\frac{\mathcal{R}_T}{T}$.

4 Thompson Sampling-based Shortest Path Learning Algorithm

In this section, we propose a Thompson Sampling-based learning algorithm to solve the OSPP. Our algorithm combines Thompson Sampling with a standard shortest path optimization routine that determines the best path (*i.e.*, the best action) for a realization of the mean travel times of the arcs in A. The key idea behind our algorithm is that the information collected when following sample-optimal paths at each time period is later used to update the posterior distribution of travel times, thereby improving future decisions. Subsequently, we propose two extensions to the algorithm that incorporates spatial correlation between the travel times of nearby arcs, by incorporating ideas from Kriging; as we will see in Section 5, these extensions can significantly speed up the learning process, improving the algorithm performance.

4.1 General Framework

For period $t \in [T]$, let $\tilde{\theta}$ be a random travel time parameter vector sampled from the distribution Θ^t at time t. We denote the mean travel time of arc $k \in [A]$ under distribution parameter vector $\tilde{\theta}$ by $\overline{X}_k(\tilde{\theta})$, and define the corresponding vector of means $\overline{X}(\tilde{\theta}) \coloneqq (\overline{X}_k(\tilde{\theta}))_{k \in [A]}$. Additionally, we denote the true realization of the travel time across arc $k \in [A]$ at time $t \in [T]$ by x_k^t, and define the history of observations up to time period t as $\mathcal{H}_t \coloneqq \{(P_{t'}, \{x_k^{t'}\}_{k \in P_{t'}})\}_{t' \in [t]}$.

We propose a Thompson Sampling-based learning procedure (in the sequel referred to as the *main algorithm*) to progressively learn the travel times across the network. The main algorithm first initializes a parameter vector $\tilde{\theta}$ sampled from the prior distribution Θ^0, and then at each time period $t \in [T]$ performs an iteration comprising the following four steps:

(i) Compute the mean travel times with respect to $\tilde{\theta}$, $\overline{X}_k(\tilde{\theta})$, for each arc $k \in [A]$.

(ii) Select path P_t from origin o_t to destination d_t by solving the shortest path problem $\mathrm{SP}(o_t, d_t, \overline{X}(\tilde{\theta}))$.

(iii) Observe travel times $\{x_k^t\}_{k \in P_t}$ and update history of observations $\mathcal{H}_t$ by incorporating the newly collected information $\{(P_t, \{x_k^t\}_{k \in P_t})\}$.

(iv) Given $\mathcal{H}_t$, compute the posterior distribution Θ^t, and sample a parameter vector $\tilde{\theta}$ from Θ^t.

Steps (i), (iii), and (iv) are similar to the standard Thompson Sampling algorithm used for traditional MAB problems. In turn, step (ii) enforces that the selected super-arm defines a simple directed path connecting o_t with d_t. This is done by solving shortest path problem $\text{SP}(o_t, d_t, \overline{X}(\tilde{\theta}))$, which finds a (o_t, d_t)-path of minimum mean travel time with respect to sampled parameter vector $\tilde{\theta}$; given an optimal solution $w^* \in \{0,1\}^A$ for $\text{SP}(o_t, d_t, \overline{X}(\tilde{\theta}))$, the decision maker selects path $P_t = \{k \in [A] : w_k^* = 1\}$. Then in step (iii) the decision maker observes the realized travel times $x_k^t, k \in P_t$, and stores these observations in the history $\mathcal{H}_t$. Finally, step (iv) updates the posterior distribution Θ_t based on $\mathcal{H}_t$, and samples a new parameter vector $\tilde{\theta}$ from the updated posterior distribution Θ^t for computing $\overline{X}(\tilde{\theta})$ at the next iteration. Step (iv) is a key part of the algorithm since as the decision maker continues to visit different arcs, the posterior distribution gradually becomes more informative of travel times across the network, thereby improving the quality of the paths computed in step (ii) at future iterations.

4.2 Uncertainty Modeling and Parameters Updating

We devote the rest of this section to describe in detail step (iv). In particular, we discuss how we model uncertainty, and how we use new observations to update the probabilistic models, and then present two mechanisms that incorporate spatial correlation between nearby arcs; as we will see, this can significantly speed up the learning of more efficient paths across the network.

4.2.1 Random Variables Distribution

Following results from the transportation literature on how to model the distribution of travel times (Lecluyse et al. 2009, Srinivasan et al. 2014, Chen et al. 2018), we opt to model the travel time of each arc as a Log-Normal random variable. Although we are interested in ultimately finding paths that yield the shortest travel time, learning the distribution of travel time presents some practical challenges; specifying prior information about the travel times from each arc (*i.e.*, an initial "guess" of the parameters of its distribution) is not trivial as setting an acceptable prior value for the variance of its travel time might depend on arc-specific attributes unknown to the decision maker. To avoid this issue, we choose to instead focus the learning process on random variables related to the travel *speed* across arcs, as for these the decision maker can readily initialize a common prior value for all arcs. Modeling travel time using Log-Normal distributions allows us to easily relate travel speed with travel times, making it possible to rely on observed travel times to update speed estimates: if $\ell_k > 0$ is the length of arc $k \in [A]$ and $X_k \sim \text{LogN}(\mu_k^{travel}, \lambda_k)$ represents the random travel time across k, then the random travel speed across k, S_k, satisfies $S_k := \frac{\ell_k}{X_k} \sim \text{LogN}(-\mu_k^{travel} + \ln(\ell_k), \lambda_k)$, namely the travel speed can also be modeled as a Log-Normal random variable. Furthermore, Log-Normal random variables have the useful property of being related to normal distribution; if $Z \sim \text{LogN}(\mu_Z, \lambda_Z)$, then $\ln(Z) \sim \mathcal{N}(\mu_Z, \lambda_Z)$. This allows us to exploit properties of normal random variables during the learning process, thereby making it more efficient as we will later show.

Consequently, the random variables we consider in the learning process are the arcs' log-speeds, *i.e.*, $Y_k \coloneqq \ln(S_k) \sim \mathcal{N}(\mu_k, \lambda_k), \forall k \in [A]$ (where $\mu_k \coloneqq -\mu_k^{travel} + \ln(\ell_k)$), and the parameters to be learned are $\theta = (\mu, \lambda)$, with $\lambda \coloneqq (\lambda_k)_{k\in[A]}$. For simplicity, in the sequel we assume the decision maker directly observes log-speeds when traversing an arc, noting that the log-speed of arc $k \in [A]$ observed at time $t \in [T]$, y_k^t, can be easily computed from the corresponding observed travel time x_k^t as $y_k^t = \ln\left(\frac{\ell_k}{x_k^t}\right)$.

4.2.2 Independent Gaussian Model (IGM)

Let the set of random log-speeds $\{Y_k\}_{k\in[A]}$ be a real-value spatial random process, where we first assume that random variables Y_k are independent, and each has unknown mean μ_k and variance λ_k. To learn these parameters, we derive prior distributions for each and propose a Bayesian framework that updates them as new realizations of Y_k become available, for all $k \in [A]$.

Due to the assumption that arcs are independent, we present the Bayesian framework for a generic arc $k \in [A]$. Let Y_k be a Normal distributed random variable such that $Y_k \sim \mathcal{N}(\mu_k, \lambda_k)$. For time period $t \in [T]$, let $T_{k,t} \subseteq [t]$ be the set of time periods up to (and including) t such that arc k was part of the corresponding selected path, *i.e.*, $T_{k,t} \coloneqq \{t' \in [t] : k \in P_{t'}\}$, and let $\{y_k^{t'}\}_{t'\in T_{k,t}}$ be the observed realizations of log-speed Y_k up to time t. We assume the variance λ_k follows an Inverse-Gamma prior distribution, *i.e.*, $\frac{1}{\lambda_k} \sim \text{Ga}(\alpha_k^0, \beta_k^0)$, with known $\alpha_k^0 > 1$ and $\beta_k^0 > 0$. We also assume a prior Normal distribution for μ_k, *i.e.*, $\mu_k \sim \mathcal{N}(\eta_k^0, \frac{\lambda_k}{\kappa_k^0})$, with $\kappa_k^0 > 0$ a known parameter. Note that the initial values for the mean and variance of Y_k are provided by, respectively, parameters η_k^0 and $\frac{\beta_k^0}{\alpha_k^0 - 1}$.

It follows that for each arc $k \in [A]$, the prior distribution of (μ_k, λ_k) is a Normal-Inverse Gamma distribution, *i.e.*, $\mu_k, \lambda_k | \eta_k^0, \kappa_k^0, \alpha_k^0, \beta_k^0 \sim \Theta_k^0 \coloneqq \text{NIG}(\eta_k^0, \kappa_k^0, \alpha_k^0, \beta_k^0)$. Hence, for each iteration $t \in [T]$ of the main algorithm (*i.e.*, after observing t paths), the posterior distribution of (μ_k, λ_k) is given by

$$\begin{aligned} p(\mu_k, \lambda_k | \{y_k^{t'}\}_{t'\in T_{k,t}}, \eta_k^0, \kappa_k^0, \alpha_k^0, \beta_k^0) &= p(\{y_k^{t'}\}_{t'\in T_{k,t}} | \mu_k, \lambda_k)\; p(\mu_k, \lambda_k | \eta_k^0, \kappa_k^0, \alpha_k^0, \beta_k^0) \\ &= \mathcal{N}(\{y_k^{t'}\}_{t'\in T_{k,t}} | \mu_k, \lambda_k)\; \mathcal{N}\left(\mu_k \middle| \eta_k^0, \tfrac{\lambda_k}{\kappa_k^0}\right) \text{IGa}\left(\lambda_k | \alpha_k^0, \beta_k^0\right) \\ &\propto \left(\frac{1}{\lambda_k}\right)^{\alpha_k^t + \frac{3}{2}} \exp\left(-\frac{1}{2\lambda_k}\left(2\beta_k^t + \kappa_k^t(\mu_k - \eta_k^t)^2\right)\right) \end{aligned} \tag{4}$$

with

$$\begin{aligned} \eta_k^t &= \frac{\kappa_k^0 \eta_k^0 + \sum_{t'\in T_{k,t}} y_k^{t'}}{\kappa_k^0 + |T_{k,t}|} \\ \kappa_k^t &= \kappa_k^0 + |T_{k,t}| \\ \alpha_k^t &= \alpha_k^0 + \tfrac{|T_{k,t}|}{2} \\ \beta_k^t &= \beta_k^0 + \tfrac{1}{2}\sum_{t'\in T_{k,t}} (y_k^{t'} - \bar{y}_k)^2 + \frac{\kappa_k^0 |T_{k,t}| (\eta_k^0 - \bar{y}_k^t)^2}{2(\kappa_k^0 + |T_{k,t}|)} \end{aligned} \tag{5}$$

and $\bar{y}_k^t = \frac{1}{|T_{k,t}|}\sum_{t' \in T_{k,t}} y_k^{t'}$ if $T_{k,t} \neq \emptyset$, else 0. Subsequently,

$$\mu_k, \lambda_k | \{y_k^{t'}\}_{t' \in T_{k,t}}, \eta_k^0, \kappa_k^0, \alpha_k^0, \beta_k^0 \sim \Theta_k^t := \text{NIG}(\eta_k^t, \kappa_k^t, \alpha_k^t \beta_k^t), \tag{6}$$

Algorithm 1 describes in detail how the main algorithm updates the posterior distribution of (μ, λ) and how it samples parameters from it at a specific iteration $t \in [T]$ (*i.e.*, , step (iv)). For each arc $k \in [A]$, Algorithm 1 first updates the set of time periods at which k has been visited and the corresponding average of observed log-speed (lines 3 - 5), then computes the updated parameters for the posterior distribution by evaluating the set of Equations (5), and finally samples a mean and variance for the arcs' log-speed (lines 6 and 7; we provide the derivation of Equations (4) - (6) in Appendix A). Note that in Algorithm 1 (and subsequent algorithms) we omit the time index t in the history, the vector of average log-speeds, and the set of visiting times; the reason is that in practice, these are implemented as global variables and are incrementally updated as new paths are observed. In the rest of the paper, we refer to this updating and sampling approach as the Independent Gaussian Model (IGM).

Algorithm 1: Posterior Update and Parameter Sampling – IGM

Input : Time period t, set of arcs $[A]$, history of observed paths and log-speeds $\mathcal{H} = \{(P_{t'}, \{y_k^{t'}\}_{k \in P_{t'}})\}_{t' \in [t]}$, prior distributions $\Theta^0 := (\Theta_k^0)_{k \in [A]}$, vector of average observed log-speeds $\bar{y} := (\bar{y}_k)_{k \in [A]}$, sets of time periods each arc has been visited at $\{T_k\}_{k \in [A]}$.

Output: Sampled travel time distribution parameters $(\tilde{\mu}, \tilde{\lambda})$.

1 Initialize $A-$dimensional vectors $\tilde{\mu} := (\tilde{\mu}_k)_{k \in [A]}, \tilde{\lambda} := (\tilde{\lambda}_k)_{k \in [A]}$;
2 **forall** $k \in [A]$ **do**
3 **if** $k \in P_t$ **then**
4 $\bar{y}_k \leftarrow \frac{|T_k| \bar{y}_k + y_k^t}{|T_k|+1}$;
5 $T_k \leftarrow T_k \cup \{t\}$;
6 Compute posterior distribution Θ_k^t by evaluating set of Equations (5) using history $\mathcal{H}$ parameters from prior distribution Θ_k^0;
7 $(\tilde{\mu}_k, \tilde{\lambda}_k) \leftarrow$ Sample from posterior distribution Θ_k^t;
8 **return** $(\tilde{\mu}, \tilde{\lambda})$

4.3 Spatial Gaussian Model

Across the literature, random variables associated with arcs in a network are typically assumed to be independent in order to simplify the analysis. In practice, however, vehicles that traverse through arcs close in proximity at a given time commonly exhibit similar speeds (*e.g.*, heavier than usual traffic at an artery may decrease the speed of vehicles in the artery and in nearby arcs). In this section, we incorporate this spatial effect into the travel log-speeds distributions.

Note that in our setting, learning the correlation between every pair of arcs in the network can become impractical: the decision maker would need to explore multiple (likely sub-optimal) paths to determine the inter-dependencies of the arcs, ultimately incurring a considerable total regret over the planning horizon. To keep the learning process efficient, we assume a special structure for the correlation between arcs' log-speeds. Particularly, we model log-speed distributions as a spatial Gaussian process that assumes a degree of correlation between each pair of arcs based on the distance between them. Then, when the decision maker selects a path and observes the travel times (and thus log-speeds) of the associated arcs, they can use these observations to update not only the distribution parameters of arcs in the traversed path but also of arcs in its vicinity: for a given travel time realization, since the random variables follow a Gaussian distribution, we can easily compute the conditional distribution for the unobserved variables. Overall, the spatial Gaussian model we propose offers two main advantages: it reduces the number of parameters to be learned (*i.e.*, the inter-dependencies of arcs); and it allows indirect learning, *i.e.*, the decision maker can update unobserved arcs' distribution parameters.

Let Z be a stationary Gaussian process with support in $\mathbb{R}^A$ and with zero mean that exhibits spatial correlation. We model the travel log-speed Y_k of an arc $k \in [A]$ as

$$Y_k = \mu_k + Z_k + \varepsilon_k^{error} \tag{7}$$

where $\mu_k \in \mathbb{R}$ is a constant, Z_k is a Gaussian process, and ε_k^{error} is a Normal distributed error, $\varepsilon_k^{error} \sim \mathcal{N}(0, \sigma^2)$.

Let $c_k \coloneqq (c_k^x, c_k^y)$ be a two-dimensional vector denoting the position of the center of arc $k \in [A]$. To model the spatial correlation, we define $\rho_{kk'} = \|c_k - c_{k'}\|$ as the distance between the centers of arcs $k, k' \in [A]$, and assume that variables Z_k and $Z_{k'}$ (and hence Y_k and $Y_{k'}$) are more correlated as $\rho_{kk'}$ is closer to zero (and conversely, the correlation decreases as $\rho_{kk'}$ increases). Specifically, for each pair of arcs $k, k' \in [A]$ we denote the correlation between random variables Y_k and $Y_{k'}$ by $\phi_{kk'}$, and correspondingly define it as $\phi_{kk'} = K(\rho_{kk'}, \varphi)$, where $\varphi > 0$ is a spatial influence parameter that controls the effect of the distance between two arcs' centers on their interdependence, and $K(\cdot, \cdot)$ denotes the kernel function used to model the correlation between arcs. Furthermore, we encode all the pairwise correlations into a correlation matrix $\Phi \coloneqq (\phi_{kk'})_{k \in [A], k' \in [A]}$. Given variances $\lambda_k > 0$ for each arc $k \in [A]$, we define the spatial covariance between Z_k and $Z_{k'}$ as

$$\mathrm{Cov}(Z_k, Z_{k'}) \coloneqq \phi_{kk'} \sqrt{\lambda_k \lambda_{k'}} \tag{8}$$

and subsequently, define the log-speed spatial covariance matrix as

$$\Sigma \coloneqq (\mathrm{Cov}(Y_k, Y_{k'}))_{k \in [A], k' \in [A]} = \Lambda^{\frac{1}{2}} \Phi \Lambda^{\frac{1}{2}} + \sigma^2 I \tag{9}$$

where $\Lambda = \mathrm{diag}(\lambda)$, and I denotes the identity matrix of size A.

To incorporate spatial correlations, we slightly modify step (iv) of the main algorithm. At each $t \in [T]$, right after sampling distribution parameters $(\tilde{\mu}, \tilde{\lambda})$ from posterior distribution Θ^t, we apply an additional update on $(\tilde{\mu}, \tilde{\lambda})$ using conditional Normal distributions (*i.e.*, conditional on history $\mathcal{H}_t$), based on the following theorem.

Theorem 1 (Murphy (2012) Theorem 4.3.1)**.** *Suppose the components of the random vector* $X = (X_1, X_2) \in \mathbb{R}^{n_1+n_2}$ *are jointly Gaussian, with parameters*

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.$$

Then the conditional posterior of X_1 *given* $X_2 = x_2$ *satisfies*

$$X_1 | X_2 = x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$$

where

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$$
$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

Specifically, we propose two alternative strategies for updating the parameters: a path-based strategy that independently uses the information of each path observed so far to perform the update; and an aggregated strategy that performs updates based on the average value of observations. For both strategies, we assume Φ is known. To mathematically describe these strategies, for a subset of arcs $Q \subseteq [A]$, we define $\tilde{\mu}[Q] \coloneqq (\tilde{\mu}_k)_{k \in Q}$, and denote the covariance sub-matrix of arcs in Q by $\Sigma[Q] \coloneqq (\Sigma_{kk'})_{k \in Q, k' \in Q}$. We also define, for each $k \in [A] \setminus Q$, the vector $\Sigma_k[Q] \coloneqq (\Sigma_{k'k})_{k' \in Q}$.

4.3.1 Path-Based (PB) Spatial Updating Strategy

The key idea of this strategy is that right after sampling distribution parameters in step (iv), the decision maker iteratively adjusts the sample based on paths in the historical information (*i.e.*, one path at a time, in chronological order); specifically, the decision maker uses log-speed observations of arcs in each observed path to update the remaining arcs based on spatial correlations.

Let $t \in [T]$ be the current iteration of the main algorithm. For each time period $t' \in [t]$, we define the vector $y^{t'} \coloneqq (y_k^{t'})_{k \in P_{t'}}$, and for subset of path's arcs $Q \subseteq P_{t'}$ we let the vector $y^{t'}[Q] \coloneqq (y_k^{t'})_{k \in Q}$ (for $Q = P_{t'}$ we simply write $y^{t'}$); then given traversed path $P_{t'}$ and vectors of means and variances $\tilde{\mu}$ and $\tilde{\lambda}$ sampled in step (iv) of the current iteration t, it follows from Theorem 1 that for each arc $k \in [A]$, $Y_k | \bigcup_{k' \in P_{t'}} \{Y_{k'} = y_{k'}^{t'}\} \sim \mathcal{N}(\mu_k^{post}(\tilde{\mu}, \Sigma, P_{t'}, y^{t'}), \lambda_k^{post}(\tilde{\lambda}, \Sigma, P_{t'}) + \sigma^2)$, where σ^2 is the unknown internal variance of the Gaussian process (as defined in Equation (7)), and μ_k^{post} and λ_k^{post} are functions defined as

$$\mu_k^{post}(\tilde{\mu}, \Sigma, P_{t'}, y^{t'}) = \begin{cases} \tilde{\mu}_k + \Sigma_k[P_{t'}]^\top \Sigma[P_{t'}]^{-1}(y^{t'} - \tilde{\mu}[P_{t'}]) & \text{if } k \notin P_{t'} \\ \tilde{\mu}_k & \text{otherwise} \end{cases} \tag{10a}$$

$$\lambda_k^{post}(\tilde{\lambda}, \Sigma, P_{t'}) = \begin{cases} \tilde{\lambda}_k - \Sigma_k[P_{t'}]^\top \Sigma[P_{t'}]^{-1}\Sigma_k[P_{t'}] & \text{if } k \notin P_{t'} \\ \tilde{\lambda}_k & \text{otherwise} \end{cases} \tag{10b}$$

Thus at each $t \in [T]$, the PB strategy iteratively performs "spatial" updates on sampled vectors $\tilde{\mu}$ and $\tilde{\lambda}$ by applying Equations (10a) and (10b) for each $(P_{t'}, y^{t'}) \in \mathcal{H}_t$.

Note that in practice, computing the covariance matrix Σ for spatially updating $\tilde{\mu}$ and $\tilde{\lambda}$ requires estimating σ^2. We do so for each $(P_{t'}, y^{t'}) \in \mathcal{H}_t$, prior to spatially updating $\tilde{\mu}$ and $\tilde{\lambda}$. Specifically, we first compute a covariance matrix assuming zero internal variance, $\hat{\Sigma} := \tilde{\Lambda}^{\frac{1}{2}} \Phi \tilde{\Lambda}^{\frac{1}{2}}$, where $\tilde{\Lambda} = \text{diag}(\tilde{\lambda})$; then for each arc $k \in P_{t'}$, we use $\hat{\Sigma}$ to estimate its log-speed based on k's spatial correlation with the remaining observed arcs $P_{t'} \setminus \{k\}$, as

$$\hat{y}_k^{t'} := \tilde{\mu}_k + \hat{\Sigma}_k[P_{t'} \setminus \{k\}]^\top \hat{\Sigma}[P_{t'} \setminus \{k\}]^{-1}(y^{t'}[P_{t'} \setminus \{k\}] - \tilde{\mu}[P_{t'} \setminus \{k\}])$$

and then estimate the internal variance as the mean squared error between predicted and observed log-speeds for arcs in $P_{t'}$, *i.e.*, $\hat{\sigma}^2 := \frac{1}{|P_{t'}|} \sum_{k \in P_{t'}} (y_k^{t'} - \hat{y}_k^{t'})^2$. Finally, we apply Equations (10a) and (10b) using $\Sigma = \hat{\Sigma} + \hat{\sigma}^2 I$.

Algorithm 2 describes in detail how the PB spatial strategy updates $\tilde{\mu}$ and $\tilde{\lambda}$. First, the algorithm initializes $\tilde{\mu}$ and $\tilde{\lambda}$ by executing Algorithm 1, and then updates both vectors based on the PB spatial strategy: for each past time period $t' \in [t]$, the algorithm computes $\tilde{\Lambda}$ and $\hat{\Sigma}$, and use them to compute log-speed predictions of arcs observed in path $P_{t'}$ based on current estimation of arcs' log-speed means and variances (lines 3 - 6). These predictions are in turn used to estimate the internal variance $\hat{\sigma}^2$ (line 7). Lastly, lines 8 and 9 update $\tilde{\mu}$ and $\tilde{\lambda}$ according to Equations (10a) and (10b), respectively. After the last spatial update, the algorithm returns updated vectors $\tilde{\mu}$ and $\tilde{\lambda}$.

Algorithm 2: Posterior Update and Parameter Sampling – PB Spatial Strategy

Input : Time period t, set of arcs $[A]$, history of observed paths and log-speeds $\mathcal{H} = \{(P_{t'}, \{y_k^{t'}\}_{k \in P_{t'}})\}_{t' \in [t]}$, prior distributions Θ^0, vector of average observed log-speeds $\bar{y} := (\bar{y}_k)_{k \in [A]}$, sets of time periods each arc has been visited at $(T_k)_{k \in [A]}$, correlation matrix Φ.

Output: Updated travel time distribution parameters $(\tilde{\mu}, \tilde{\lambda})$.

1 $(\tilde{\mu}, \tilde{\lambda}) \leftarrow$ Algorithm 1$(t, [A], \mathcal{H}, \Theta^0, \bar{y}, (T_k)_{k \in [A]})$;
2 **forall** $t' \in [t]$ **do**
3 $\quad \tilde{\Lambda} \leftarrow \text{diag}(\tilde{\lambda})$;
4 $\quad \hat{\Sigma} \leftarrow \tilde{\Lambda}^{\frac{1}{2}} \Phi \tilde{\Lambda}^{\frac{1}{2}}$;
5 $\quad$ **forall** $k \in P_{t'}$ **do**
6 $\quad\quad \hat{y}_k^{t'} \leftarrow \tilde{\mu}_k + \hat{\Sigma}_k[P_{t'} \setminus k]^\top \hat{\Sigma}[P_{t'} \setminus k]^{-1}(y^{t'}[P_{t'} \setminus k] - \tilde{\mu}[P_{t'} \setminus k])$;
7 $\quad \hat{\sigma}^2 \leftarrow \frac{1}{|P_{t'}|} \sum_{k \in P_{t'}} (y_k^{t'} - \hat{y}_k^{t'})^2$;
8 $\quad \tilde{\mu} \leftarrow \mu^{post}(\tilde{\mu}, \hat{\Sigma} + \hat{\sigma}^2 I, P_{t'}, y^{t'})$;
9 $\quad \tilde{\lambda} \leftarrow \lambda^{post}(\tilde{\lambda}, \hat{\Sigma} + \hat{\sigma}^2 I, P_{t'})$;
10 **return** $(\tilde{\mu}, \tilde{\lambda})$

4.3.2 Path-Aggregated (PA) Spatial Updating Strategy

The PB spatial strategy is able to accelerate the learning process (thus reducing the overall regret, as we will show later in Section 5). However, this strategy can be computationally expensive, primarily due to having to invert t correlation matrices at every execution. Hence, in this section, we propose a less expensive alternative, the Path-Aggregated (PA) spatial strategy. In contrast to its PB counterpart, the PA strategy only updates arcs that have never been visited up to the time of the update; furthermore, instead of applying the spatial effect iteratively over each past time period, this approach averages all observations for each observed arc and updates all non-observed arcs in one update step, thus requiring inverting a covariance matrix only once per execution.

Given current iteration $t \in [T]$, let $Q_t \subseteq [A]$ be the set of arcs that have been visited at least once, *i.e.*, $Q_t := \{k \in [A] : T_{k,t} \neq \emptyset\}$; and let $\bar{y}^t := (\bar{y}^t_k)_{k \in Q_t}$. Then by Theorem 1, we have that the random log-speed Y_k for each $k \in [A]$ satisfies $Y_k | \bigcup_{k' \in Q_t} \{Y_{k'} = \bar{y}^t_{k'}\} \sim \mathcal{N}(\mu^{post}_k(\tilde{\mu}, \Sigma, Q_t, \bar{y}^t), \lambda^{post}_k(\tilde{\lambda}, \Sigma, Q_t) + \sigma^2)$, with

$$\mu^{post}_k(\tilde{\mu}, \Sigma, Q_t, \bar{y}^t) = \begin{cases} \tilde{\mu}_k + \Sigma_k[Q_t]^\top \Sigma[Q_t]^{-1}(\bar{y}^t - \tilde{\mu}[Q_t]) & \text{if } k \notin Q_t \\ \tilde{\mu}_k & \text{otherwise} \end{cases} \tag{11a}$$

$$\lambda^{post}_k(\tilde{\lambda}, \Sigma, Q_t) = \begin{cases} \tilde{\lambda}_k - \Sigma_k[Q_t]^\top \Sigma[Q_t]^{-1} \Sigma_k[Q_t] & \text{if } k \notin Q_t \\ \tilde{\lambda}_k & \text{otherwise} \end{cases} \tag{11b}$$

We design the PA strategy to be more consistent with IGM samples by setting $\sigma = 0$, which translates into getting $\tilde{\mu}_k = \mu^{post}_k(\tilde{\mu}, \Sigma, Q_t, \bar{y}^t)$ and $\tilde{\lambda}_k = \lambda^{post}_k(\tilde{\lambda}, \Sigma, Q_t)$ after the update. This setting, also known as noiseless Kriging (Cressie 1990), assumes the average of observed values as ground truth log-speeds for each visited arc, and estimates the log-speeds of unobserved arcs based on their spatial correlation.

Algorithm 3 shows the PA spatial updating strategy for $\tilde{\mu}$ and $\tilde{\lambda}$. Compared to Algorithm 2, it requires one additional input, Q_t, which in our implementation is iteratively updated every time a new path is observed (and thus we omit the time index t). The algorithm initializes $\tilde{\mu}$ and $\tilde{\lambda}$ by executing Algorithm 1 and then computes $\hat{\Sigma}$. Lastly, vectors $\tilde{\mu}$ and $\tilde{\lambda}$ are updated using Equations (11a) and (11b), respectively. The algorithm returns updated vectors $\tilde{\mu}$ and $\tilde{\lambda}$.

Algorithm 3: Posterior Update and Parameter Sampling – PA Spatial Strategy

Input : Time period t, set of arcs $[A]$, history of observed paths and log-speeds $\mathcal{H} = \{(P_{t'}, \{y_k^{t'}\}_{k \in P_{t'}})\}_{t' \in [t]}$, prior distributions Θ^0, vector of average observed log-speeds $\bar{y}$, sets of time periods each arc has been visited at $(T_k)_{k \in [A]}$, correlation matrix Φ, set of traversed arcs Q.

Output: Updated travel time distribution parameters $(\tilde{\mu}, \tilde{\lambda})$.

1 $(\tilde{\mu}, \tilde{\lambda}) \leftarrow$ Algorithm 1$(t, [A], \mathcal{H}, \Theta^0, \bar{y}, (T_k)_{k \in [A]})$;
2 $\tilde{\Lambda} \leftarrow \text{diag}(\tilde{\lambda})$;
3 $\hat{\Sigma} \leftarrow \tilde{\Lambda}^{\frac{1}{2}} \Phi \tilde{\Lambda}^{\frac{1}{2}}$;
4 $\tilde{\mu} \leftarrow \mu^{post}(\tilde{\mu}, \hat{\Sigma}, Q, \bar{y}^t)$;
5 $\tilde{\lambda} \leftarrow \lambda^{post}(\tilde{\lambda}, \hat{\Sigma}, Q)$;
6 **return** $(\tilde{\mu}, \tilde{\lambda})$

4.4 Main Algorithm

Algorithm 4 shows the implementation of the main algorithm described in Section 4.1. It first initializes the necessary vectors and sets (lines 1 - 5); in particular, it initializes vectors $\tilde{\mu}$ and $\tilde{\lambda}$ by sampling from the log-speed priors (line 5). Then for each iteration (*i.e.*, time period), in lines 7 and 8 the algorithm computes the mean log-speeds of each arc (step (i)) and correspondingly determines the shortest $(o_t, d_t)-$path by solving Model (1) (step (ii)). Subsequently, the decision maker observes the log-speeds from arcs in the shortest path and updates the history accordingly (step (iii); lines 9 - 11), and ends the iteration by obtaining the posterior of arcs' log-speeds based on history, either using only the IGM or applying one of the proposed spatial updating strategies (step (iv); lines 12 - 18).

5 Computational Experiments

In this section, we show computation results from applying the proposed algorithms to artificial and real-world instances. The goal of these experiments is to (i) identify the advantages and disadvantages of each method; (ii) analyze the effect of parameter value selection on their performances; and (iii) demonstrate the effectiveness of our learning-based algorithms. We first present results on a set of artificial instances, which allows us to study the methods in a more controlled setting, and then we show results on an instance constructed using real taxi data from the city of Beijing.

Algorithm 4: Thompson Sampling-based Shortest Path Learning Algorithm

Input : Set of time periods $[T]$, set of arcs $[A]$, prior distributions Θ^0, correlation matrix Φ, set of O-D pairs $\{(o_t, d_t)\}_{t\in[T]}$.

$\mathcal{H} \leftarrow \emptyset, Q \leftarrow \emptyset$;
Initialize vectors $\tilde{\mu} \coloneqq (\tilde{\mu}_k)_{k\in[A]}, \tilde{\lambda} \coloneqq (\tilde{\lambda}_k)_{k\in[A]}, \overline{X} \coloneqq (\overline{X}_k)_{k\in[A]}, \bar{y} \coloneqq (\bar{y}_k)_{k\in[A]}$;
forall $k \in [A]$ **do**
    $T_k \leftarrow \emptyset$;
    $(\tilde{\mu}_k, \tilde{\lambda}_k) \leftarrow$ Sample from prior distribution Θ_k^0;
forall $t \in [T]$ **do**
    $\overline{X} \leftarrow (\exp(\tilde{\mu}_k + \frac{\tilde{\lambda}_k}{2}))_{k\in[A]}$;
    $P_t \leftarrow$ Shortest path from solving optimization problem $\mathrm{SP}(o_t, d_t, \overline{X})$;
    forall $k \in P_t$ **do**
        $y_k^t \leftarrow$ Observed log-speed for arc k;
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(P_t, \{y_k^t\}_{k\in P_t})\}$;
    if *using* ***IGM*** **then**
        $(\tilde{\mu}, \tilde{\lambda}) \leftarrow$ Algorithm 1$(t, [A], \mathcal{H}, \Theta^0, \bar{y}, \{T_k\}_{k\in[A]})$;
    else if *using* ***PB Spatial Updating Strategy*** **then**
        $(\tilde{\mu}, \tilde{\lambda}) \leftarrow$ Algorithm 2$(t, [A], \mathcal{H}, \Theta^0, \bar{y}, \{T_k\}_{k\in[A]}, \Phi)$;
    else if *using* ***PA Spatial Updating Strategy*** **then**
        $Q \leftarrow Q \cup P_t$;
        $(\tilde{\mu}, \tilde{\lambda}) \leftarrow$ Algorithm 3$(t, [A], \mathcal{H}, \Theta^0, \bar{y}, \{T_k\}_{k\in[A]}, \Phi, Q)$;

5.1 Performance on Artificial Instances

The experiments on artificial instances consider the IGM, as well as the PB and PA spatial updating strategies. For each spatial strategy, we test different configurations, each given by a combination of values of distance norm $\rho_{kk'}$, spatial influence parameter φ, and kernel function K from the ones shown in Table 1. We compare all three approaches in terms of pseudo-regret. Our goal with these controlled instances is to show which algorithms have the best performance, as well as to analyze the effect of the algorithmic configurations on their performance.

Parameter	Values
Distance norm $\rho_{kk'}$	Manhattan ($\|c_k - c_{k'}\|_1$), Euclidean ($\|c_k - c_{k'}\|_2$)
Spatial influence φ	$0.67, 1.33, 2, 2.67, 3.33, 4$
Kernel $K(\rho_{kk'}, \varphi)$	Exponential ($\exp(-\frac{\rho_{kk'}}{\varphi})$), Gaussian ($\exp(-(\frac{\rho_{kk'}}{\varphi})^2)$)

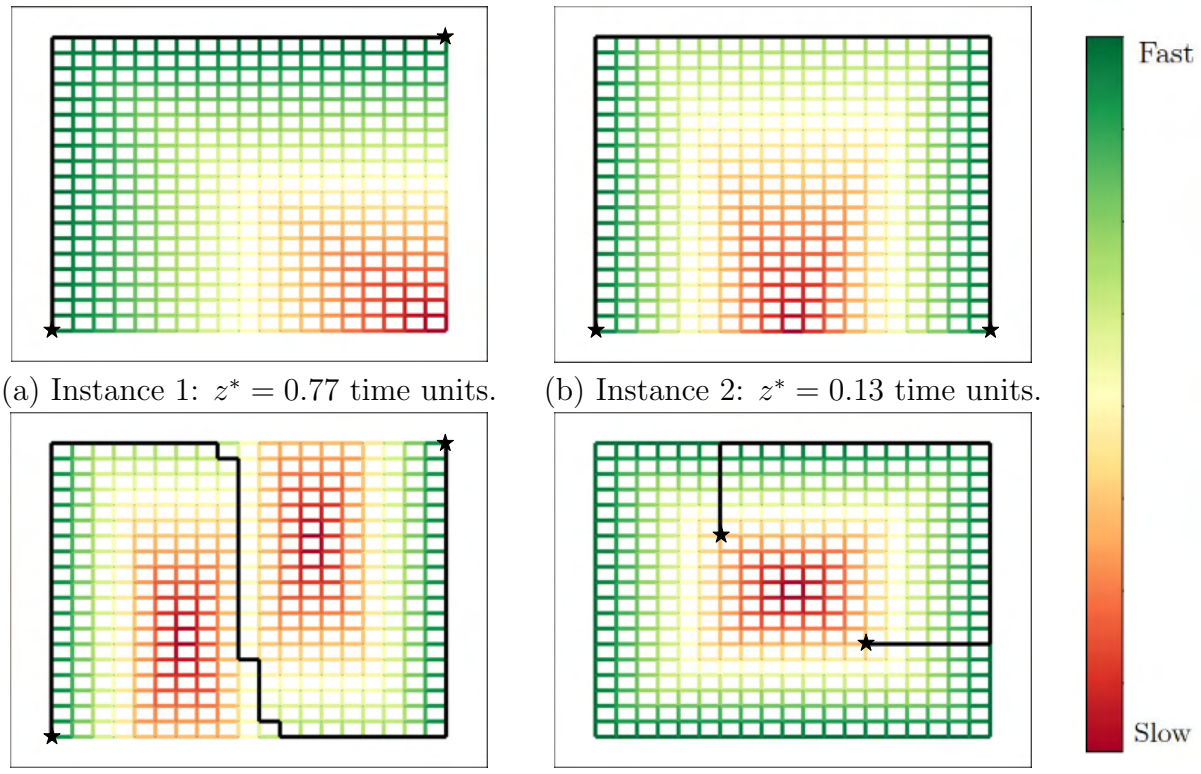
Table 1: Configurations for PB and PA spatial updating strategies tested on synthetic instances

In order to test the effectiveness of our methods to find the optimal path (in the sequel also referred to as *expert's solution*), we consider a total of four artificial instances, each defined by a set of different distributions and magnitudes of arcs' speeds, and a corresponding expert's solution (whose expected travel time we denote by z^*), see Figure 1. Based on the complexity of the expert's solution, we classify the instances into "simple" and "difficult".

Instances 1 and 2 correspond to simple instances. In instance 1, the origin and destination nodes are respectively located at the bottom left and top right corners of the network, and arcs are constructed such that they become more expensive (slower) the closest they are to the right and bottom borders; correspondingly, the expert's solution consists of all arcs in the top and left borders of the network. Instance 2 shares the same origin node, but the destination node is now located at the bottom right corner of the network. Furthermore, this instance considers an obstacle (*i.e.*, set of very expensive arcs) that covers most of the center of the network except its upper border, hence the optimal path corresponds to all arcs in the left, top, and right borders of the graph.

In turn, instances 3 and 4 correspond to difficult instances. In instance 3, the origin and destination nodes are located at the bottom left and top right corners, respectively. The expert's solution circumvents two obstacles, describing a "snake-shaped" path. In instance 4, both origin and destination nodes are in opposite corners of a very slow zone in the interior of the network, and the fastest arcs are located in the network's borders. Thus instead of traversing directly towards the destination, the optimal path requires first leaving the slow zone towards the graph's top border (in the opposite direction of the destination), and then traversing clockwise through the network's borders in order to reach the destination node from the right. For replication purposes, we include the pseudo-codes used to generate these instances in Appendix B.1.

For each instance, we construct 500 episodes of $T = 50$ time periods, each defined by



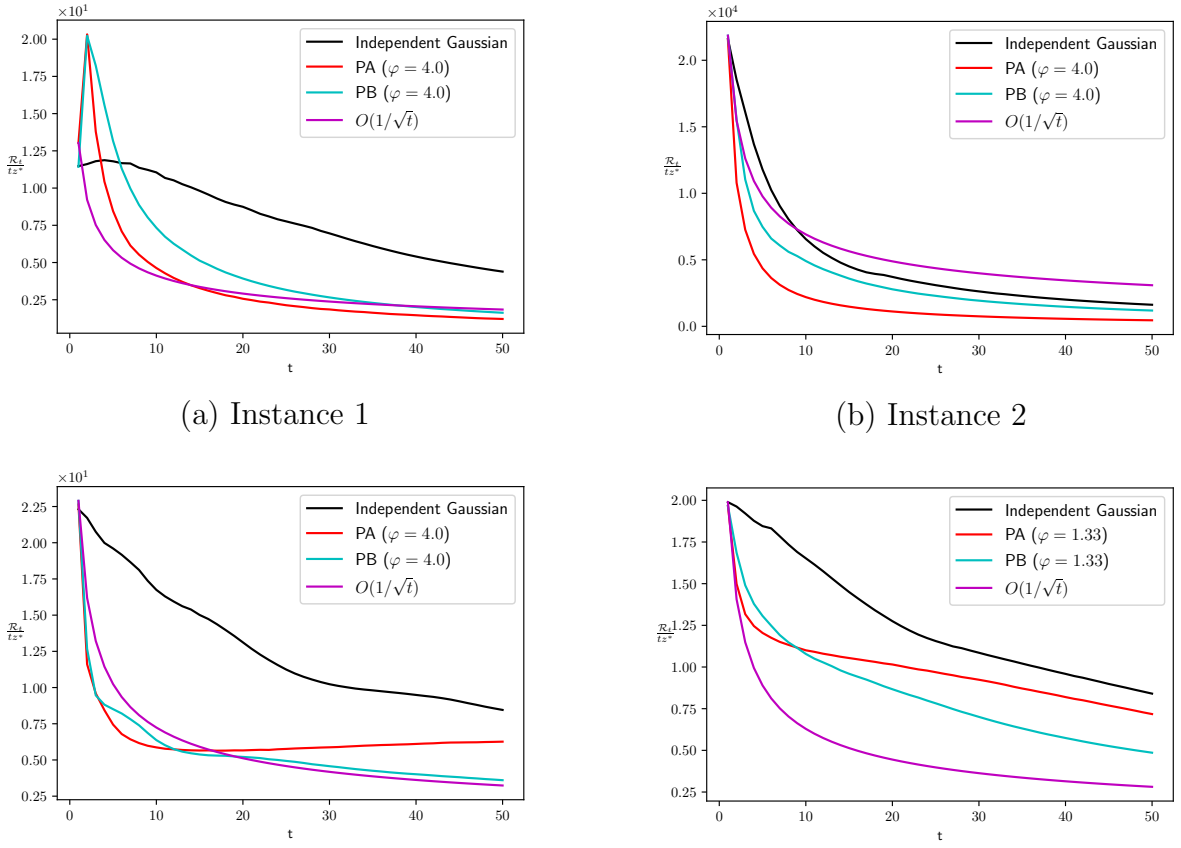
(a) Instance 1: $z^* = 0.77$ time units. (b) Instance 2: $z^* = 0.13$ time units.

(c) Instance 3: $z^* = 202.49$ time units. (d) Instance 4: $z^* = 1.44$ time units.

Figure 1: Illustration of artificial instances. The colors represent the true mean of the arcs' log-speeds. Instances 1 and 2 correspond to simple instances, and instances 3 and 4 to difficult instances. Paths corresponding to expert's solutions are depicted using black lines.

an initial prior distribution and realization of arcs' speeds across all time periods, and then execute the IGM and all possible configurations of PA and PB spatial strategies on all episodes. We measure the performance of each approach as its attained average total pseudo-regret, $\frac{\mathcal{R}_T}{T}$, averaged across all 500 episodes. For the spatial strategies, we use the kernel functions most used in the literature and specifically in the context of Kriging (Rasmussen and Williams 2005, Lantuéjoul 2013), although other kernel functions could be readily used. The values of the parameter φ can be interpreted as the maximum distance within which two arcs are spatially correlated (which we define as having a spatial correlation that is greater than 5%). For example, when using the Exponential kernel, for tested values $\varphi \in \{0.67, 1.33, \ldots, 4\}$, the correlation is considered relevant only for arcs separated by a distance of $2, 4, \ldots, 12$ arcs, respectively.

For all four instances, Figure 2 shows the evolution over time of the time-average total pseudo-regret (normalized by z^*) attained by each proposed method. Each data point corresponds to the normalized total pseudo-regret at each time period, averaged over the 500 corresponding episodes. For the spatial strategies, these results are obtained using the



(a) Instance 1

(b) Instance 2

(c) Instance 3

(d) Instance 4

Figure 2: Average time-average total pseudo-regret over 500 episodes, for all artificial instances. Spatial strategies use the Exponential kernel and Euclidean distance, and their corresponding value of φ is set to the one that yields the lowest average total pseudo-regret by time period $t = 50$.

Exponential kernel with Euclidean distance, and the spatial influence parameter value that yields the best average total pseudo-regret for each specific instance.

We observe that in general, the PA and PB strategies outperform the IGM, showing the most significant performance differences in instances 1, 2, and 3. In terms of performance relative to z^*, all proposed methods perform better in the difficult instances (see the order of magnitude on the vertical axes of Figure 2). When comparing both spatial strategies, we observe that the PA strategy achieves a lower total pseudo-regret by the last iteration for the simple instances; however, the PB strategy attains better results for the difficult instances. In particular, for instance 3, the total pseudo-regret from the PA strategy starts to increase at iteration 25. Our conjecture is that the PB strategy has optimal regret guarantees, whereas the PA method does not; still, PA reports an overall good empirical performance in our simulations. We also note that the spatial approaches have decreasing curves in the number

of iterations, with the exception of instance 1 where both PA and PB methods experience an initial increase in their pseudo-regret (iterations 2-3) before trending downwards. Although we do not provide theoretical guarantees, we observe in our results that the pseudo-regret curves for both PA and PB describe a pattern close to $\mathcal{O}(1/\sqrt{T})$, just as the optimal methods for MAB problems (Bubeck and Cesa-Bianchi 2012).

Kernel	φ	**Norm**	**Pseudo-regret**				**% of wins**			
			$t = 10$		$t = 50$		$t = 10$		$t = 50$	
			PA	**PB**	**PA**	**PB**	**PA**	**PB**	**PA**	**PB**
Exponential	0.67	1	1.01	1.25	0.72	1.05	0.48	0.13	0.87	0.06
		2	0.93	1.30	0.62	1.00	0.60	0.10	0.88	0.08
	1.33	1	0.77	1.17	0.48	0.76	0.65	0.17	0.85	0.14
		2	0.74	1.05	0.43	0.64	0.67	0.18	0.79	0.21
	2	1	0.68	0.99	0.40	0.58	0.67	0.23	0.74	0.26
		2	0.66	0.89	0.40	0.49	0.64	0.29	0.63	0.37
	2.67	1	0.65	0.89	0.41	0.49	0.61	0.30	0.61	0.39
		2	0.56	0.82	0.40	0.45	0.67	0.27	0.56	0.44
	3.33	1	0.55	0.85	0.37	0.46	0.68	0.27	0.62	0.38
		2	0.49	0.74	0.35	0.41	0.67	0.29	0.59	0.41
	4	1	0.48	0.80	0.33	0.42	0.71	0.24	0.64	0.36
		2	**0.44**	0.70	**0.28**	0.38	0.66	0.30	0.63	0.37
Gaussian	0.67	2	1.09	1.20	0.85	1.08	0.43	0.18	0.71	0.10
	1.33	2	**0.87**	1.24	**0.66**	0.90	0.66	0.12	0.78	0.18
	2	2	0.99	1.23	1.24	0.84	0.55	0.17	0.08	0.72
	2.67	2	1.47	1.24	1.55	0.88	0.25	0.27	0.03	0.70
	3.33	2	1.80	1.29	1.71	0.94	0.16	0.29	0.02	0.61
	4	2	2.14	1.26	1.82	0.98	0.11	0.31	0.01	0.52
IGM			11.05		4.39					

Table 2: For instance 1, average (over all episodes) of the time-average total pseudo-regret, and percentage of episodes in which each method achieves the lowest total pseudo-regret among all three methods, at iterations $t = 10$ and $t = 50$. The pseudo-regret reported for IGM is normalized by z^*. The pseudo-regret reported for the spatial methods is normalized by the reported pseudo-regret for the IGM.

Table 2 reports the results obtained for instance 1. Each row represents the results of a spatial configuration. The first three columns indicate the kernel function, the influence parameter, and the distance metric used in each spatial configuration. The next four columns report the average total pseudo-regret obtained by PA and PB strategies for time periods 10 and 50; for an easier comparison, these values are normalized by the total pseudo-regret attained by the IGM at the corresponding time period, shown in the bottom row. For each spatial configuration, the last four columns show the percentage of episodes in which each spatial approach attains the lowest total pseudo-regret of the three methods, at iterations 10 and 50 (the percentage corresponding to the IGM can be computed as the difference between 1 and the sum of the corresponding percentages of PA and PB).

Based on instance 1, the approach with the overall best performance corresponds to

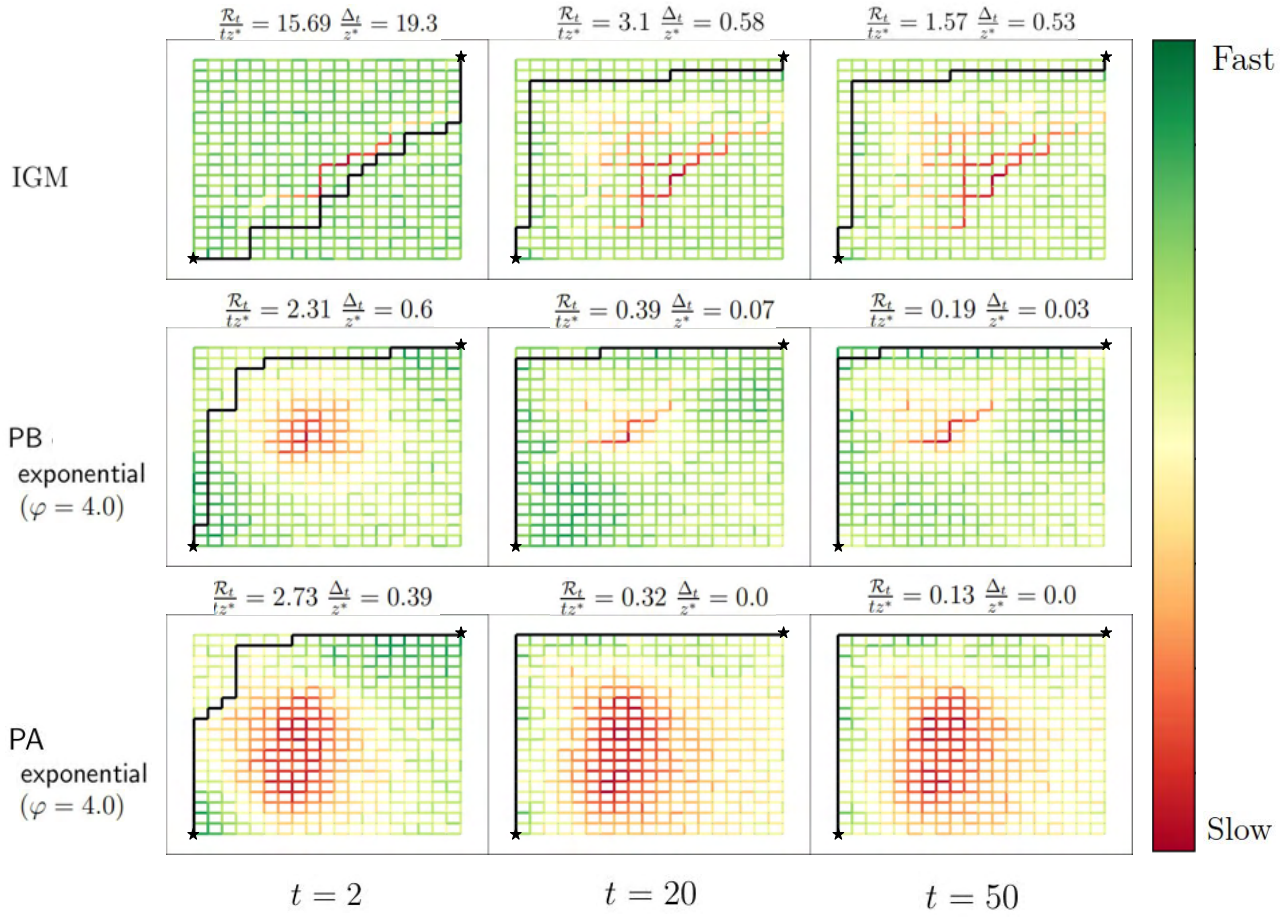


Figure 3: For instance 1, selected path and associated normalized pseudo-regrets for all algorithms at different time periods.

the PA strategy with Exponential kernel, with an average total pseudo-regret that is lower than the IGM and the PB strategy with Exponential kernel, considering all values of φ and distance metrics. Moreover, in all configurations, by the last iteration the PA strategy with Exponential kernel wins over the IGM and the PB exponential model more than 56% of the episodes (see column % of wins for PB at $t = 50$, in Table 2). For instance 1, the Gaussian kernel is generally not as competitive as the Exponential kernel. Still, both approaches and kernels outperform the IGM (for some φ), attaining better total pseudo-regret values even at early iterations. For example, for $\varphi = 1.33$ and a distance norm of 2, the total pseudo-regret attained by the PA strategy with either the Exponential or Gaussian kernels corresponds to only a fraction of the total pseudo-regret of the IGM (43% for the Exponential kernel, and 66% for the Gaussian kernel). For the Exponential kernel, the value of φ that shows the best total pseudo-regret is 4; we also observe that for this kernel, the Euclidean distance performs better than the Manhattan distance. Moreover, our results suggest that for the Exponential kernel, the correlation between arcs in instance 1 is properly captured when

setting a value of $\varphi \geq 1.33$ (*i.e.*, an influence distance of at least 4 arcs), which is consistent with the instance's speeds: the arc speeds are similar for arcs separated by a distance of at least 4-arcs both in the vertical and horizontal direction (on the horizontal direction, this number is even higher since on some directions the speed does not change at all). For the Gaussian kernel, $\varphi = 1.33$ yields the best total pseudo-regret.

Figure 3 shows the progression over time of the estimated speeds means of the posterior distributions (average of samples) and origin-destination paths, selected by the IGM and spatial strategies (with Exponential kernel) as they explore the network. For each approach, we show its inferred arcs' speeds (average of the posterior distribution) and the paths it selects at time periods 2, 20, and 50 of an episode. Note that the spatial strategies achieve a significantly higher performance than the IGM. At iteration 2, while the IGM is still exploring paths strictly within the grid, the PA and PB models are already exploring the border of the graph (where the fastest arcs are located). At iteration 20, the PA model has found the shortest path while the PB and independent models are still choosing sub-optimal paths. Notably, the marginal pseudo-regret attained by the IGM at iteration 50 is comparable to the average total pseudo-regret attained by the PA and PB strategies at iteration 20.

Kernel	φ	**Norm**	**Pseudo-regret**				**% of wins**			
			$t = 10$		$t = 50$		$t = 10$		$t = 50$	
			PA	**PB**	**PA**	**PB**	**PA**	**PB**	**PA**	**PB**
Exponential	0.67	1	0.75	0.90	0.92	0.93	**0.99**	0.01	0.60	0.38
		2	0.72	0.84	0.90	0.90	0.97	0.03	0.51	0.48
	1.33	1	0.68	0.70	0.88	0.66	0.63	0.37	0.01	0.99
		2	0.67	**0.65**	0.86	**0.58**	0.43	0.57	0.01	0.99
	2	1	0.68	0.66	0.87	0.66	0.39	0.61	0.09	0.91
		2	0.67	0.67	0.85	0.77	0.49	0.51	0.38	0.61
	2.67	1	0.69	0.69	0.89	0.87	0.52	0.48	0.49	0.49
		2	0.69	0.76	0.85	1.15	0.75	0.25	0.86	0.13
	3.33	1	0.71	0.75	0.90	1.16	0.68	0.32	0.82	0.15
		2	0.70	0.82	0.86	1.49	0.84	0.16	0.97	0.02
	4	1	0.73	0.81	0.91	1.45	0.78	0.22	0.93	0.03
		2	0.72	0.87	0.87	1.66	0.86	0.14	0.98	0.00
Gaussian	0.67	2	0.76	0.93	0.93	0.96	0.99	0.01	0.71	0.25
	1.33	2	0.74	0.77	0.97	0.81	0.65	0.35	0.01	0.99
	2	2	0.82	0.73	1.06	0.72	0.19	0.81	0.00	1.00
	2.67	2	0.91	0.71	1.12	0.72	0.08	0.92	0.00	1.00
	3.33	2	0.98	0.74	1.12	0.77	0.05	0.95	0.00	0.98
	4	2	1.04	0.77	1.14	0.84	0.03	0.96	0.00	0.89
IGM			1.65		0.84					

Table 3: For artificial instance 4, average (over all episodes) of the time-average total pseudo-regret, and percentage of episodes in which each method achieves the lowest total pseudo-regret among all three methods, at iterations $t = 10$ and $t = 50$. The pseudo-regret reported for IGM is normalized by z^*. The pseudo-regret reported for the spatial methods is normalized by the reported pseudo-regret for the IGM.

Table 3 shows results for instance 4. The columns, parameters, and configurations dis-

played are the same as in Table 2. Unlike instance 1, in this instance the best performance is attained by the PB strategy. In particular, the lowest regret is observed when using the Exponential kernel with Euclidean norm and $\varphi = 1.33$. This value for φ is consistent with the instance arc speeds distribution: the correlation is meaningful for arcs within 4-arc distance. As for the PA strategy in instance 1, the Euclidean norm achieves the best performance for the Exponential kernel for the PB strategy. Overall, most configurations of both spatial strategies outperform the IGM.

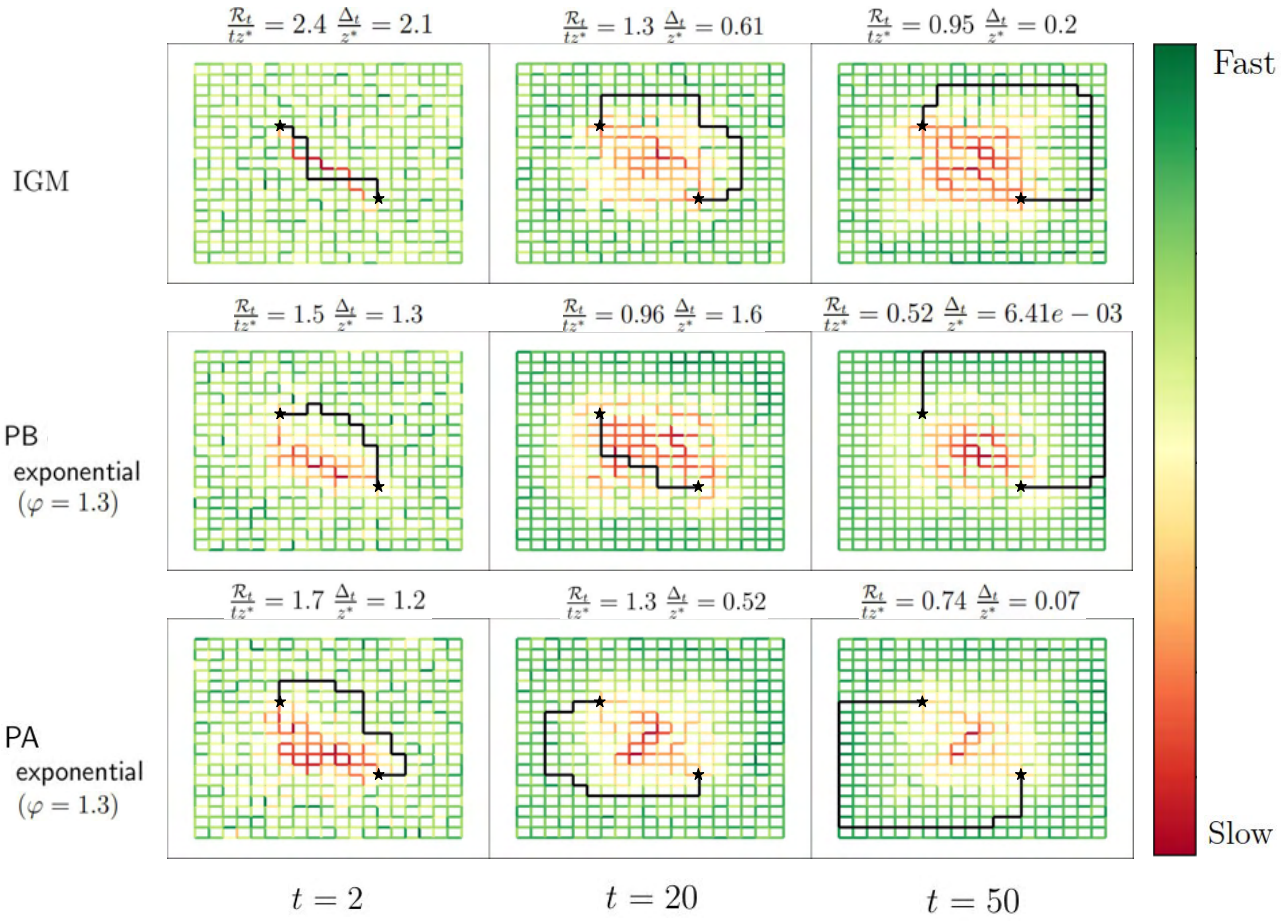


Figure 4: For instance 4, selected path and associated normalized pseudo-regrets for all algorithms at different time periods.

Figure 4 shows the evolution of selected paths and estimated speeds means across the arcs for a simulated episode of instance 4, for the IGM and spatial strategies with Exponential kernel and Euclidean distance, at iterations 2, 20, and 50. By iteration 2, the IGM explores O-D paths that directly cross the slow zone, whereas the spatial models start recognizing the slow zone. At iterations 2 and 20, the PB strategy yields a higher marginal pseudo-regret than the PA strategy, since the PB model still explores paths with high pseudo-regret (for PB, $\frac{\Delta_2}{z^*} = 1.3$, and $\frac{\Delta_{20}}{z^*} = 1.6$). However, by iteration 50 the PB (and to some extent the PA)

strategy has already learned the arcs' speeds distributions and selects paths that are very close to the expert's solution, ultimately attaining the lowest total pseudo-regret among all three algorithms. On the other hand, the IGM continues exploring arcs within the grid.

Aggregate results and examples for instances 2 and 3 can be found in Appendix B.

5.2 Case Study: City of Beijing

The previous section shows that the proposed algorithms perform well on artificial instances under specific distributional assumptions. To empirically demonstrate their practical capabilities, in the remainder of this section, we analyze how our methods perform in a case study using real data from the city of Beijing. Specifically, in Section 5.2.1 we explain how the real-world instance is constructed, and in Section 5.2.2 we analyze how the proposed algorithms perform against a benchmark.

5.2.1 Instance Generation

For our experiments, we seek to model Beijing as a graph with each of its arcs associated with a travel time distribution; we determine these using empirical distributions based on observed times (instead of parametric probability distributions) to prevent performance boosts due to distributional assumptions.

We choose the city of Beijing due to its large population, which causes challenges for the city's transportation, and also due to the large publicly available source of GPS data of vehicles traveling around the city. The graph of Beijing is downloaded from the OpenStreetMap project (OpenStreetMap contributors 2017), and the GPS points are obtained from Microsoft Research's *T-Drive trajectory dataset* in Zheng (2011). This dataset corresponds to one week of trips of 10,357 taxis. The total number of records in this dataset is about 15 million, and the total traveled distance of the trajectories amounts to 9 million kilometers; each record specifies a GPS point within a route and its corresponding timestamp, and each route consists of multiple records. We map these GPS points into O-D data, each with corresponding departure and arrival times. Figure 5 shows the Beijing area considered in the case study.

To obtain travel times, we follow three steps: first, we pre-process the records in the GPS dataset based on geographical projection and average speed thresholds; second, we iteratively estimate the average travel time for each arc in the graph based on the paths' travel times from the dataset; and third, we project the travel time values for each GPS O-D record using the estimated path and times. For the first and second tasks, we follow the methodology proposed in Bertsimas et al. (2019).

Following the pre-processing done in Bertsimas et al. (2019), we first filter out walkways and service roads, and eliminate nodes that do not represent the intersection of two or more arcs. Furthermore, we consider that across each O-D path, the average speed from one point to another must be within an interval; correspondingly, we discard all O-D data records whose average speed is less than 2 kph or greater than 110 kph. We then project each remaining GPS O-D record to corresponding nodes of the graph so both origin and

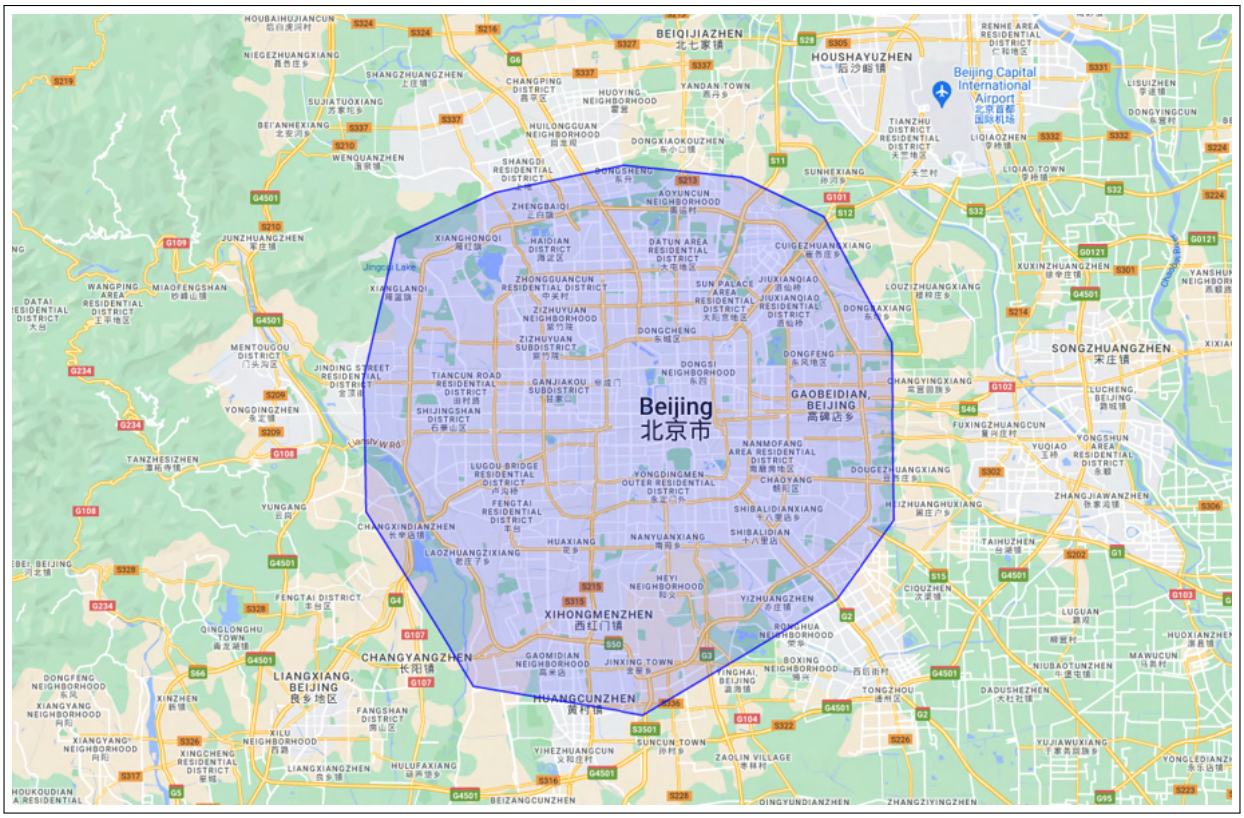


Figure 5: Beijing area selected for the case study.

destination are respectively associated with one of the graph's nodes, and drop records whose projected origin (destination) node is not within 250 meters of the origin's (destination's) GPS coordinates specified in the dataset. The pre-processed instance comprises 34,850 nodes, 42,892 arcs, and 238,000 GPS records. For the GPS timestamps, we select taxi trips completed between 7:00 am and 11:00 am on weekdays as this corresponds to a time of the day with high traffic. The average GPS travel time between each O-D pair is 356 seconds.

To estimate the average travel times and paths followed by vehicles for each O-D pair, we use the Bertsimas et al. (2019) methodology. This is an iterative algorithm: first, it estimates average times for each arc using as input the paths followed by vehicles; then, it finds the shortest paths using the estimated averages, which in turn are used to update the average travel time estimations. Our implementation of the algorithm stops when either (i) the difference in the average number of arcs between the paths found in one iteration and the next one is less than 0.5, or (ii) after completing 10 iterations. From the algorithm parameters used in their work, we choose the same regularization parameter value $\lambda = 1000$ and set the path memory limit to $\Pi = 2$.

Once the average times for each arc and the vehicles' path are estimated, we project the time values of each GPS O-D pair. For each arc k in a vehicle's path, we generate a travel

time value $\tilde{x}_k$ of the form,

$$\tilde{x}_k = \frac{\hat{x}_k T_{od}}{\hat{T}_{od}}, \tag{12}$$

where $\hat{x}_k$ is the estimated average time for arc k, $\hat{T}_{od}$ is the estimated travel time for the path between o and d, and T_{od} is the travel time specified in the corresponding GPS O-D record. We perform this projection for each GPS O-D record and obtain a set of travel time values for each arc. For our experiments, we use each arc's empirical distribution defined by these time values. Thus, when an algorithm selects a path containing a given arc, the observed travel time for that arc is sampled (with replacement) from the set of time values associated with that arc.

The generated instance (graph and time values per arc) can be found in the following link https://github.com/felipelagos/beijing-instance. Figure 6 shows the Beijing graph, with colored arcs representing the estimated average speeds between 7:00 am to 11:00 am.

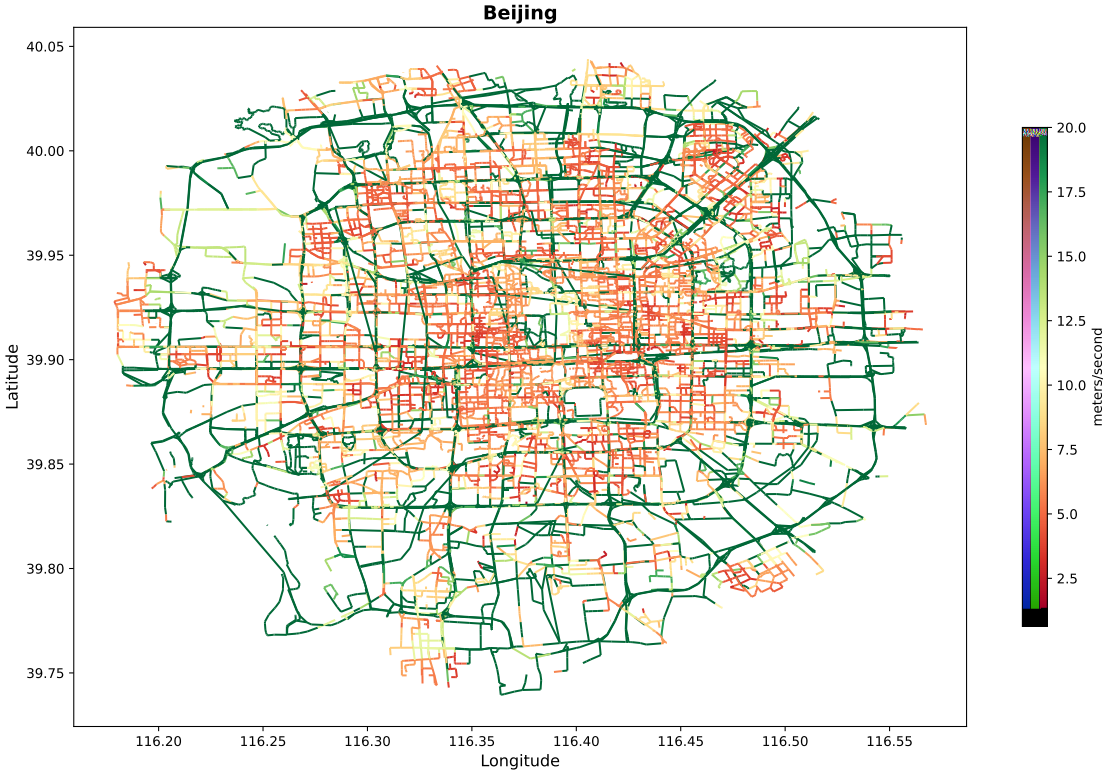


Figure 6: Average speed between 7:00 am and 11:00 am for each arc in the graph.

5.2.2 Results and Analysis

These experiments comprise several O-D pairs, which are selected as the pairs that yield the worst pseudo-regrets from a pool of 100 O-D pairs when using a uniform prior-based approach (we describe this procedure in detail in Appendix C). Due to the increased size of

the real-world instances (relative to the artificial instances), we increase the number of time periods to $T = 150$; in general, we observe that this value is large enough for our approach to visit the expert's solution at least once in most of the episodes.

Importantly, these instances result to be too large for the PA and PB strategies due to requiring considerable memory and computational resources. Hence, out of the three proposed approaches, we omit the spatial extensions in this section and limit the analysis to the IGM. Additionally, we provide an alternative baseline approach to test the value of the IGM. This baseline policy is inspired by the well-known ε-greedy algorithm of the Reinforcement Learning community (Sutton and Barto 2018); it considers each arc's travel time distribution to have a single point estimate, computed as the mean of the observed values; and for arcs that currently have no observations, it instead uses the mean prior value (*i.e.*, η_k^0 for arc k). To add an exploration component, after the decision maker selects a first route and all its arcs have already been visited at least once, with probability ε we allow the baseline policy to forbid one arc uniformly at random and compute an alternative route. In the sequel, we refer to this baseline approach as the ε-greedy to facilitate association. Unless stated otherwise, all comparisons between IGM and the ε-greedy algorithm are done using the value of ε that yields the best final time-average total pseudo-regret $\frac{\mathcal{R}_T}{T}$.

Priors values used by the IGM and the ε-greedy are initialized by inspection. The average speed in the city is initialized to $\eta_k^0 = 40$ kph, considering that this is typically between 20 and 60 kph depending on the type of road. However, to make the priors not informative enough for the IGM, we set the initial values of the other parameters to $\alpha_k^0 = \kappa_k^0 = 1$ and $\beta_k^0 = 3$; doing so introduces enough variance to the initial samples of the mean log-speed prior distribution so that algorithms are able to find main roads and highways.

Figure 7 compares IGM and ε-greedy in terms of marginal and average total pseudo-regret per time period for the instance where ε-greedy attains its best average total pseudo-regret per time period (see Figure 12 in Appendix C for the tuning of the exploration parameter ε). These results suggest that in general IGM outperforms the ε-greedy algorithm. The pseudo-regret of IGM starts higher but improves faster than the pseudo-regret of ε-greedy , which results in IGM attaining a 47% better time-average total pseudo-regret by the last time period. In the first periods ($t \in [0, 40]$), we observe that the IGM explores enough to find high-quality solutions, on average approaching the expert's optimal value (see the marginal pseudo-regret curve in Figure 7). By contrast, during these time periods, the ε-greedy algorithm exploits its incumbent solutions as its exploration mechanism is unlikely to be triggered (due to most arcs having no observations). By iteration $t = 40$ the IGM starts exploiting its incumbent solution and converging, and by iteration $t = 60$ this exploitation pays off as IGM surpasses the ε-greedy in terms of average total pseudo-regret per time period. After $t = 30$, we occasionally observe pronounced peaks in the marginal pseudo-regret of ε-greedy due by policy exploration, which slightly deteriorates the time-average total pseudo-regret. By contrast, the IGM experiences considerably smaller peaks in its marginal pseudo-regret from exploration, and these do not affect its time-average total pseudo-regret.

Table 4 reports the average performance of IGM and ε-greedy by the last time period over all episodes. In more than 70% of the cases, the IGM is able to either find the expert's

solution or a solution that yields a marginally higher pseudo-regret (attaining a pseudo-regret within .5% from the expert's solution in 9 out of 10 instances). For the ε-greedy approach, we observe that the values $\varepsilon \in \{0.1, 0.3\}$ yield better performance; however, they do so by exploiting their corresponding incumbent solution, being unable to find the expert's solution in any of the episodes. As higher values of ε are considered, *i.e.*, $\varepsilon \in \{0.5, 0.7, 0.9\}$, we observe a tendency to find the expert's solution more often, and in some cases even decrease the pseudo-regret; however, in most cases the more frequent exploration considerably deteriorates the overall performance of the ε-greedy approach.

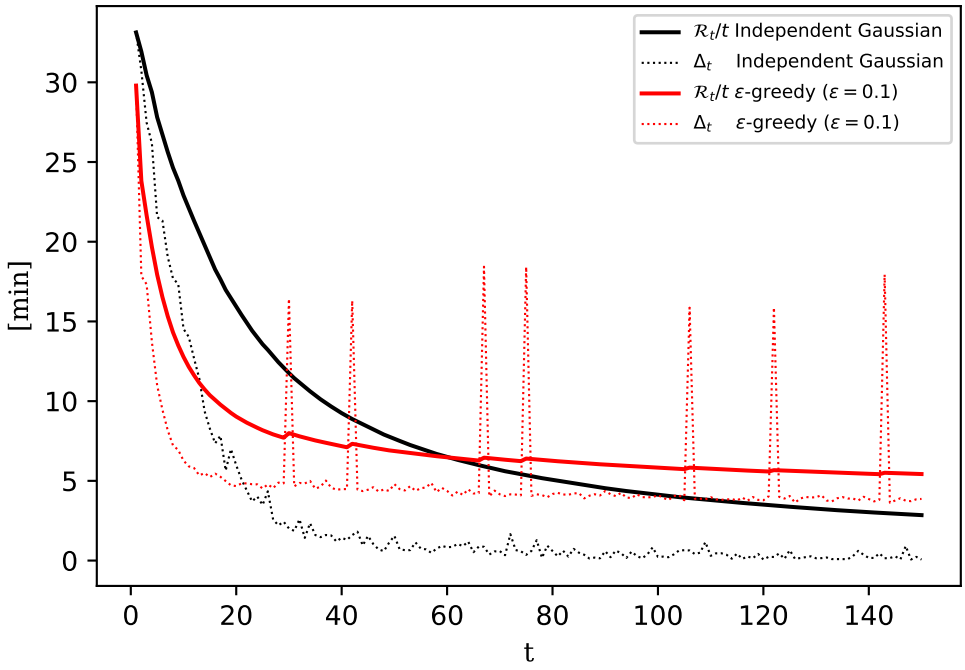


Figure 7: Average pseudo-regrets obtained from 100 episodes of the Beijing case study with multiple origin-destination pairs (10 episodes from each of the 10 instances presented in Appendix C).

6 Conclusions

Motivated by logistic problems currently faced by companies, we studied the OSPP. In this problem, a planner must find a vehicle route from an origin to a destination that minimizes the total travel time. Since in practice travel times are stochastic and follow a probability distribution that is unknown to the planner, they must balance the exploration of new paths and the exploitation of the most up to date travel time information. This problem has been

$z^*[min]$	IGM			ε-greedy														
				$\varepsilon = 0.1$			$\varepsilon = 0.3$			$\varepsilon = 0.5$			$\varepsilon = 0.7$			$\varepsilon = 0.9$		
	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$	#v.e.	$\frac{\overline{\Delta}_T}{z^*}$	$\frac{\overline{\mathcal{R}}_T}{Tz^*}$
28.99	7	0.01	0.12	0	0.09	0.11	0	0.11	0.12	1	0.09	0.13	0	0.08	0.14	2	0.08	0.14
29.50	6	0.00	0.12	0	0.06	0.14	0	0.06	0.11	0	0.04	0.12	0	0.04	0.11	1	0.04	0.11
30.02	5	0.00	0.10	0	0.13	0.17	0	0.14	0.17	1	0.16	0.19	1	0.15	0.2	0	0.16	0.21
30.17	5	0.00	0.10	0	0.12	0.16	0	0.12	0.17	0	0.09	0.18	0	0.11	0.18	0	0.17	0.18
31.92	9	0.00	0.09	0	0.14	0.17	0	0.14	0.16	0	0.16	0.18	0	0.13	0.17	0	0.17	0.18
32.40	9	0.00	0.08	0	0.08	0.29	0	0.08	0.68	0	0.07	1.72	0	4.49	3.01	0	8.87	3.16
32.67	5	0.00	0.10	0	0.14	0.16	0	0.13	0.16	0	0.14	0.17	0	0.19	0.17	0	0.16	0.19
33.91	8	0.00	0.08	0	0.05	0.07	0	0.05	0.08	0	0.06	0.09	0	0.07	0.10	0	0.09	0.11
34.26	8	0.00	0.07	0	0.19	0.24	0	0.19	0.21	0	0.14	0.20	0	0.16	0.21	0	0.16	0.22
36.60	9	0.00	0.04	0	0.18	0.17	0	0.13	0.16	0	0.10	0.16	0	0.11	0.16	0	0.14	0.16

Table 4: Average performance per instance. Each row aggregates the results of 10 episodes: #v.e. denotes the number of episodes each algorithm visits the expert's solution at least once, $\overline{\Delta}_T$ denotes the average (over episodes) marginal pseudo-regret at time period T, and $\overline{\mathcal{R}}_T$ denotes to the average (over episodes) total pseudo-regret at time period T.

mainly studied for packet transmission in wireless networks but not for logistics operation (vehicle routing and travel times estimation). Along with introducing a new problem, we formulate it as a CMAB problem and propose Thompson Sampling-based algorithm that effectively finds policies that balances the trade-off between exploration and exploitation.

Our work first proposes a Thompson Sampling-based algorithm to find the best paths within an operating network, the IGM. Then, it develops two new extensions of the Thompson Sampling which exploit the spatial information of the graph to learn parameters more efficiently, the PB and the PB spatial updating strategies; both of these strategies use Gaussian modeling concepts inspired by the Kriging literature to refine the samples generated by Thompson Sampling using spatial information, attaining better travel time estimations and ultimately determining more efficient vehicle routes. We test our algorithms on a set of simple and difficult artificial instances, and on a case study using real data from the city of Beijing. We show that by incorporating spatial correlation, our algorithms effectively produce policies that outperform the Thompson Sampling-based algorithm in terms of total pseudo-regret. For the case study, the MAB policy is efficient and effective in minimizing pseudo-regret.

There are several extensions and future work to be drawn from this work. We highlight determining whether the algorithms we propose offer theoretical guarantees in terms of pseudo-regret analysis. In our experiments, the PB spatial strategy shows an empirical performance that surpasses the Thompson Sampling-based algorithm in all instances and conducted simulations, which leads us to believe that this algorithm may offer better guarantees than Thompson Sampling in terms of the optimality constant for the pseudo-regret. Another possible extension consists of studying different distributions of travel times or speeds, where our algorithms could be integrated with known methods in Machine Learning to generate samples of these distributions (*e.g.*, Markov Chain Monte Carlo algorithms). In addition, future extensions could incorporate the learning of the value of the spatial influence parameter φ that best fits the underlying correlation between arcs for a given kernel. Finally, we believe that our methods can be used for adaptive problems, namely, in online settings

where at a given time period the planner may decide to deviate from an initially selected path halfway through it.

References

Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135. PMLR.

Al Mashalah, H., Hassini, E., Gunasekaran, A., and Bhatt, D. (2022). The impact of digital transformation on supply chains through e-commerce: Literature review and a conceptual framework. *Transportation Research Part E: Logistics and Transportation Review*, 165:102837.

Audibert, J.-Y., Bubeck, S., and Lugosi, G. (2011). Minimax policies for combinatorial prediction games. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 107–132. JMLR Workshop and Conference Proceedings.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Bertsimas, D., Delarue, A., Jaillet, P., and Martin, S. (2019). Travel time estimation in the age of big data. *Operations Research*, 67(2):498–515.

Bouneffouf, D., Rish, I., and Aggarwal, C. (2020). Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*.

Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79.

Cesa-Bianchi, N. and Lugosi, G. (2012). Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422.

Chakrabarti, D., Kumar, R., Radlinski, F., and Upfal, E. (2008). Mortal multi-armed bandits. *Advances in Neural Information Processing Systems*, 21.

Chapelle, O. and Li, L. (2011). An empirical evaluation of Thompson sampling. *Advances in Neural Information Processing Systems*, 24.

Chen, P., Tong, R., Lu, G., and Wang, Y. (2018). Exploring travel time distribution and variability patterns using probe vehicle data: case study in beijing. *Journal of Advanced Transportation*, 2018.

Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159. PMLR.

Chen, X., Xue, J., Qian, X., Suarez, J., and Ukkusuri, S. V. (2020). Online energy-optimal routing for electric vehicles with combinatorial multi-arm semi-bandit. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE.

Chowdhury, S. R. and Gopalan, A. (2017). On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR.

Cressie, N. (1990). The origins of kriging. *Mathematical Geology*, 22(3):239–252.

Cressie, N. (2015). *Statistics For Spatial Data*. John Wiley & Sons.

Cressie, N. and Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Euromonitor (2021). E-commerce to account for half the growth in global retail by 2025. Accessed March 22, 2023.

Ferreira, K. J., Simchi-Levi, D., and Wang, H. (2018). Online network revenue management using Thompson sampling. *Operations Research*, 66(6):1586–1602.

Fu, K., Meng, F., Ye, J., and Wang, Z. (2020). Compacteta: A fast inference system for travel time prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3337–3345.

Gómez, A., Mariño, R., Akhavan-Tabatabaei, R., Medaglia, A. L., and Mendoza, J. E. (2016). On modeling stochastic travel and service times in vehicle routing. *Transportation Science*, 50(2):627–641.

Hoffman, M., Brochu, E., de Freitas, N., et al. (2011). Portfolio allocation for Bayesian optimization. In *UAI*, pages 327–336. Citeseer.

Insider Intelligence (2021). Worldwide e-commerce continues double-digit growth following pandemic push to online. Accessed March 22, 2023.

Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer.

Khani, A. (2019). An online shortest path algorithm for reliable routing in schedule-based transit networks considering transfer failure probability. *Transportation Research Part B: Methodological*, 126:549–564.

Kharoufeh, J. P. and Gautam, N. (2004). Deriving link travel-time distributions via stochastic speed processes. *Transportation Science*, 38(1):97–106.

Komiyama, J., Honda, J., and Nakagawa, H. (2015). Optimal regret analysis of Thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *International Conference on Machine Learning*, pages 1152–1161. PMLR.

Kou, W., Wang, J., Liu, Y., and Li, C. (2022). Last-mile shuttle planning for improving bus commuters' travel time reliability: A case study of Beijing. *Journal of Advanced Transportation*, 2022:1–15.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139.

Lafkihi, M., Pan, S., and Ballot, E. (2019). Freight transportation service procurement: A literature review and future research opportunities in omnichannel e-commerce. *Transportation Research Part E: Logistics and Transportation Review*, 125:348–365.

Lai, T. L., Robbins, H., et al. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22.

Lantuéjoul, C. (2013). *Geostatistical Simulation: Models and Algorithms*. Springer Science & Business Media.

Lattimore, T. and Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press.

Lecluyse, C., Van Woensel, T., and Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. *4OR*, 7(4):363–377.

Liu, K. and Zhao, Q. (2012). Adaptive shortest-path routing under unknown and stochastically varying link states. In *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 232–237. IEEE.

Mangiaracina, R., Perego, A., Seghezzi, A., and Tumino, A. (2019). Innovative solutions to increase last-mile delivery efficiency in B2C e-commerce: a literature review. *International Journal of Physical Distribution & Logistics Management*.

Manufacturing and Logistics IT (2020). How Amazon manages its delivery routes (and how to copy them). Accessed March 22, 2023.

Mercatus and Incisiv (2021). E-grocery's new reality: The pandemic's lasting impact on U.S. grocery shopping behavior report. Accessed March 22, 2023.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.

Nuara, A., Trovò, F., Gatti, N., and Restelli, M. (2022). Online joint bid/daily budget optimization of internet advertising campaigns. *Artificial Intelligence*, 305:103663.

OpenStreetMap contributors (2017). Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

Russo, D. and Van Roy, B. (2014). Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243.

Savelsbergh, M. and Van Woensel, T. (2016). 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation Science*, 50(2):579–590.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Srinivasan, K. K., Prakash, A., and Seshadri, R. (2014). Finding most reliable paths on networks with correlated and shifted log–normal travel times. *Transportation Research Part B: Methodological*, 66:110–128.

Statista (2022). Retail e-commerce sales worldwide from 2014 to 2026. Accessed March 22, 2023.

Susilawati, S., Taylor, M. A., and Somenahalli, S. V. (2013). Distributions of travel time variability on urban roads. *Journal of Advanced Transportation*, 47(8):720–736.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

Szeto, W., Jiang, Y., Wong, K.-I., and Solayappan, M. (2013). Reliability-based stochastic transit assignment with capacity constraints: Formulation and solution method. *Transportation Research Part C: Emerging Technologies*, 35:286–304.

Talebi, M. S., Zou, Z., Combes, R., Proutiere, A., and Johansson, M. (2017). Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 63(4):915–930.

Taniguchi, E., Thompson, E., Yamada, T., van Duin, J., and Logistics, C. (2001). Network modelling and intelligent transport systems. *City Logistics. Pergamon, Oxford*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. SIAM.

Ulmer, M. W. (2017). *Approximate Dynamic Programming For Dynamic Vehicle Routing*, volume 1. Springer.

Vanelslander, T., Deketele, L., and Hove, D. V. (2013). Commonly used e-commerce supply chains for fast moving consumer goods: comparison and suggestions for improvement. *International Journal of Logistics Research and Applications*, 16(3):243–256.

Zheng, Y. (2011). T-drive trajectory data sample. T-Drive sample dataset.

Zhou, Q., Zhang, X., Xu, J., and Liang, B. (2017). Large-scale bandit approaches for recommender systems. In *International Conference on Neural Information Processing*, pages 811–821. Springer.

Zou, Z., Proutiere, A., and Johansson, M. (2014). Online shortest path routing: The value of information. In *2014 American Control Conference*, pages 2142–2147. IEEE.

Appendices

A Derivation of Prior and Posterior Distributions

This section shows the formal derivation of the posterior distribution of the log-speed of a single arc. For convenience, we perform the following two notation changes compared to the main body of the paper: (i) we omit the arc index k, and (ii) we write the time index as a subscript (instead of using it as a superscript).

Let Y be a random variable that follows a Normal distribution with mean μ and variance λ, with each parameter being itself a random variable following a prior distribution; specifically, λ following an Inverse-Gamma distribution with parameters α_0 and β_0, and μ following a Normal distribution with mean η_0 and variance $\frac{\lambda}{\kappa_0}$. The values of η_0, κ_0, α_0, and β_0 are defined a priori and represent the initially available information about the population. We have,

$$\begin{aligned} Y &\sim \mathcal{N}(\mu, \lambda) \\ \lambda &\sim \mathrm{IG}(\alpha_0, \beta_0) \\ \mu &\sim \mathcal{N}\left(\eta_0, \frac{\lambda}{\kappa_0}\right) \end{aligned}$$

From these, it follows that

$$\mu, \lambda | \alpha_0, \beta_0, \eta_0, \kappa_0 \sim \mathrm{NIG}(\eta_0, \kappa_0, \alpha_0, \beta_0),$$

where $\mathrm{NIG}(\eta_0, \kappa_0, \alpha_0, \beta_0)$ is the probability distribution function of a Normal-Inverse-Gamma with parameters $(\eta_0, \kappa_0, \alpha_0, \beta_0)$. This function is mathematically given by

$$\begin{aligned} p(\mu|\eta_0, \lambda, \kappa_0) p(\lambda|\alpha_0, \beta_0) &= \sqrt{\frac{\kappa_0}{2\pi\lambda}} \exp\left(-\frac{\kappa_0(\mu-\eta_0)^2}{2\lambda}\right) \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha)} \left(\frac{1}{\lambda}\right)^{\alpha_0+1} \exp\left(-\frac{\beta_0}{\lambda}\right), \\ &= \sqrt{\frac{\kappa_0}{2\pi}} \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha)} \left(\frac{1}{\lambda}\right)^{\alpha_0+\frac{3}{2}} \exp\left(-\frac{\kappa_0(\mu-\eta_0)^2}{2\lambda} - \frac{\beta_0}{\lambda}\right), \\ &\sim \mathrm{NIG}(\eta_0, \kappa_0, \alpha_0, \beta_0). \end{aligned}$$

For $t \in [T]$, let $T_t \subseteq [t]$ be the set of time periods up to t where an observation of Y was collected, and $\{y_{t'}\}_{t' \in T_t}$ be the corresponding set of observations. Using Bayes' theorem, we

derive the posterior distribution for μ and λ,

$$
\begin{aligned}
p(\mu, \lambda | \{y_{t'}\}_{t' \in T_t}, \eta_0, \kappa_0, \alpha_0, \beta_0) &= \frac{p(\{y_{t'}\}_{t' \in T_t} | \mu, \lambda) p(\mu, \lambda | \alpha_0, \eta_0, \kappa_0, \beta_0)}{p(\{y_{t'}\}_{t' \in T_t})} \\
&\propto \left(\frac{1}{2\pi\lambda}\right)^{\frac{|T_t|}{2}} \exp\left(-\frac{1}{2\lambda}\sum_{t' \in T_t}(y_{t'} - \mu)^2\right) \sqrt{\frac{\kappa_0}{2\pi}} \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha)} \left(\frac{1}{\lambda}\right)^{\alpha_0 + \frac{3}{2}} \exp\left(-\frac{\kappa_0(\mu - \eta_0)^2}{2\lambda} - \frac{\beta_0}{\lambda}\right) \\
&\propto \left(\frac{1}{\lambda}\right)^{\alpha_0 + \frac{|T_t|}{2} + \frac{3}{2}} \exp\left(-\frac{\kappa_0(\mu - \eta_0)^2}{2\lambda} - \frac{\sum_{t' \in T_t}(y_{t'} - \mu)^2}{2\lambda} - \frac{\beta_0}{\lambda}\right) \\
&= \left(\frac{1}{\lambda}\right)^{\alpha_0 + \frac{|T_t|}{2} + \frac{3}{2}} \exp\left(-\frac{1}{2\lambda}(\kappa_0 + |T_t|)\left(\mu^2 - 2\mu\left(\frac{\eta_0\kappa_0 + \sum_{t' \in T_t} y_{t'}}{\kappa_0 + |T_t|}\right) + \frac{\eta_0^2\kappa_0 + \sum_{t' \in T_t} y_{t'}^2}{\kappa_0 + |T_t|}\right) - \frac{\beta_0}{\lambda}\right) \\
&= \left(\frac{1}{\lambda}\right)^{\alpha_t + \frac{3}{2}} \exp\left(-\frac{1}{2\lambda}\kappa_t(\mu - \eta_t)^2 - \frac{1}{\lambda}\left(\beta_0 - \frac{\eta_t^2\kappa_t}{2} + \frac{\eta_0^2\kappa_0 + \sum_{t' \in T_t} y_{t'}^2}{2}\right)\right) \\
&= \left(\frac{1}{\lambda}\right)^{\alpha_t + \frac{3}{2}} \exp\left(-\frac{1}{2\lambda}\kappa_t(\mu - \eta_t)^2 - \frac{1}{\lambda}\left(\beta_0 + \frac{1}{2}\sum_{t' \in T_t}(y_{t'} - \bar{y})^2 + \frac{|T_t|\kappa_0(\bar{y} - \eta_0)^2}{2(\kappa_0 + |T_t|)}\right)\right) \\
&= \left(\frac{1}{\lambda}\right)^{\alpha_t + \frac{3}{2}} \exp\left(-\frac{1}{2\lambda}\kappa_t(\mu - \eta_t)^2 - \frac{\beta_t}{\lambda}\right) \\
&\sim \text{NIG}(\eta_t, \kappa_t, \alpha_t, \beta_t),
\end{aligned}
$$

with new values defined as follows,

$$
\begin{aligned}
\eta_t &= \frac{\kappa_0\eta_0 + \sum_{t' \in T_t} y_{t'}}{\kappa_0 + |T_t|}, \\
\kappa_t &= \kappa_0 + |T_t|, \\
\alpha_t &= \alpha_0 + \tfrac{|T_t|}{2}, \\
\beta_t &= \beta_0 + \frac{1}{2}\sum_{t' \in T_t}(y_{t'} - \bar{y})^2 + \frac{\kappa_0|T_t|(\eta_0 - \bar{y})^2}{2(\kappa_0 + |T_t|)},
\end{aligned}
$$

Note that the posterior distribution is also a Normal-Inverse-Gamma distribution, which is a consequence of choosing a conjugate family of distributions for prior distributions.

For computational convenience, the following equations can be used to sample from the posterior distribution,

$$
\begin{aligned}
\frac{1}{\lambda}\Bigg| \{y_{t'}\}_{t' \in T_t}, \alpha_0, \beta_0 &\sim \text{Ga}(\alpha_t, \beta_t), \\
\mu| \{y_{t'}\}_{t' \in T_t}, \lambda, \eta_0, \kappa_0 &\sim \mathcal{N}\left(\eta_t, \frac{\lambda}{\kappa_t}\right).
\end{aligned}
$$

B Artificial Instances

B.1 Instance Generation

This section outlines the procedures used to construct the underlying speed distributions for artificial instances used in Section 5.1. Recall that the networks in these instances are square grids, each with a total of A arcs of equal length ℓ, and with N nodes per side. For artificial instances, we assume that the speed of each arc in the grid follows a Log-Normal distribution. Correspondingly, Algorithms 5 through 8 outline how the arcs' true log-speed means $(\mu_k)_{k\in[A]}$ are computed for artificial instances 1 through 4, respectively. Furthermore, we set the scale parameter of all Log-Normal distributions to $\frac{1}{\sqrt{\lambda_k}} = \log(1.2), \forall k \in [A]$. Then, to sample realizations of the log-speed of arcs, we draw a sample of each of these distributions and compute its logarithm, which then is used to update the posterior distributions.

For all artificial instances, we use the values $N = 20$, $\ell = 1.5$ distance units, and $A = 760$. The output given by these algorithms is plotted in Figure 1.

We calibrate the prior parameters for all algorithms by inspection as follows. For all four instances, we set the values of α_k^0 and κ_k^0 to 50 and 1, respectively. For instance 1, we set β_k^0 to $\frac{7}{4}$, and $\frac{1}{2}$ for instances 2, 3, and 4; and for instances 1 through 4, we set η_k^0 to values 27, 20, 1, and 25, respectively.

Algorithm 5: Travel log-speed generation for artificial instance 1

Input : Set of arcs $[A]$, grid arc length ℓ, grid width N
Output: Vector of true log-speed means $(\mu_k)_{k\in[A]}$.

forall $i \in [2N-1]$ **do**
  forall $k \in [A]$ **do**
    $b_1 \leftarrow \frac{c_k^x}{\ell} = \frac{i-1}{2}$;
    $b_2 \leftarrow \frac{c_k^y}{\ell} = \frac{2N-i-1}{2}$;
    $b_3 \leftarrow \frac{c_k^x + c_k^y}{\ell} \geq N-1$;
    if b_1 **or** (b_2 **and** b_3) **then**
      $\mu_k \leftarrow (i-3)\log\left(\frac{2}{1+\sqrt{2}}\right) + \log 50$;
return $(\mu_k)_{k\in[A]}$

Algorithm 6: Travel log-speed generation for artificial instance 2

Input : Set of arcs $[A]$, grid arc length ℓ, grid width N
Output: Vector of true log-speed means $(\mu_k)_{k\in[A]}$.

$(obstacle\_position_x, obstacle\_position_y) \leftarrow \left(\frac{N-1}{2}, 0\right)$;
$(orientation\_effect_x, orientation\_effect_y) \leftarrow (10, 3.5)$;
forall $k \in [A]$ **do**
  $value_1 \leftarrow orientation\_effect_x \left|\frac{c_k^x}{\ell} - obstacle\_position_x\right|$;
  $value_2 \leftarrow orientation\_effect_y \left|\frac{c_k^y}{\ell} - obstacle\_position_y\right|$;
  $\mu_k \leftarrow (\max\{value_1, value_2\} - 60) \log\left(\frac{1+\sqrt{2}}{2}\right) + \log 50$;
return $(\mu_k)_{k\in[A]}$

Algorithm 7: Travel log-speed generation for artificial instance 3

Input : Set of arcs $[A]$, grid arc length ℓ, grid width N
Output: Vector of true log-speed means $(\mu_k)_{k\in[A]}$.

$(obstacle\_position_x^1, obstacle\_position_y^1) \leftarrow \left(\frac{N-1}{3}, \frac{N-1}{3}\right)$;
$(obstacle\_position_x^2, obstacle\_position_y^2) \leftarrow \left(\frac{2(N-1)}{3}, \frac{2(N-1)}{3}\right)$;
$orientation\_effect \leftarrow (10, 3.5)$;
forall $k \in [A]$ **do**
  $value_{11} \leftarrow orientation\_effect_1 \left|\frac{c_k^x}{\ell} - obstacle\_position_x^1\right|$;
  $value_{12} \leftarrow orientation\_effect_2 \left|\frac{c_k^y}{\ell} - obstacle\_position_y^1\right|$;
  $value_{21} \leftarrow orientation\_effect_1 \left|\frac{c_k^x}{\ell} - obstacle\_position_x^2\right|$;
  $value_{22} \leftarrow orientation\_effect_2 \left|\frac{c_k^y}{\ell} - obstacle\_position_y^2\right|$;
  $\mu_k \leftarrow (\min_{i=1,2} \max_{j=1,2} value_{ij} - 60) \log\left(\frac{1+\sqrt{2}}{2}\right) + \log 50$;
return $(\mu_k)_{k\in[A]}$

Algorithm 8: Travel log-speed generation for artificial instance 4

Input : Set of arcs $[A]$, grid arc length ℓ, grid width N
Output: Vector of true log-speed means $(\mu_k)_{k\in[A]}$.

forall $i \in [N+1]$ **do**
  forall $k \in [A]$ **do**
    $b_1 \leftarrow \left(\frac{c_k^x}{\ell} = \frac{i-1}{2}\right)$ **or** $\left(\frac{c_k^x}{\ell} = N - \frac{i+1}{2}\right)$;
    $b_2 \leftarrow \left(\frac{c_k^y}{\ell} = \frac{i-1}{2}\right)$ **or** $\left(\frac{c_k^y}{\ell} = N - \frac{i+1}{2}\right)$;
    $b_3 \leftarrow \frac{i-1}{2} \leq \frac{c_k^x}{\ell} \leq N - \frac{i+1}{2}$;
    $b_4 \leftarrow \frac{i-1}{2} \leq \frac{c_k^y}{\ell} \leq N - \frac{i+1}{2}$;
    if $(b_1$ **and** $b_4)$ **or** $(b_2$ **and** $b_3)$ **then**
      $\mu_k \leftarrow (i-3)\log\left(\frac{2}{1+\sqrt{2}}\right) + \log 50$;
return $(\mu_k)_{k\in[A]}$

B.2 Artificial Instances: Additional Results

Table 5 reports the results obtained for artificial instance 2 at iterations 10 and 50. In particular, it reports the time-average total pseudo-regret obtained by the IGM, and the spatial strategies at iterations (averaged over all run episodes); the reported regrets for the IGM are normalized by z^*, and the ones reported for PA and PB strategies are normalized by IGM's reported regret. The last four columns show the percentage of episodes in which each spatial strategy attains the lowest total pseudo-regret (the percentage of the IGM can be computed as 1 minus the sum of the percentages of both spatial methods). Similar results for artificial instance 3 are reported in Table 6.

Figure 8 shows the estimated speeds mean (average of samples) of the posterior distributions for the IGM and the spatial strategies with the Exponential kernel, for artificial instance 2. For iterations 2, 20, and 50, we plot the arc's mean speeds of a simulation and the path chosen by each method. Figure 9 shows the simulations for artificial instance 3.

C Case Study: Additional Results

The O-D pairs provided in this section are computed from an initial pool of 100 O-D pairs having at least 25 minutes of expert total travel time. For each of the 100 pairs, we first compute the shortest path based on distance, and then using a uniform-speed prior policy we solve the OSPP and calculate the obtained total pseudo-regret. From these results, we select the 10 pairs for which the policy achieves the worst total pseudo-regret as instances for the case study. These are displayed in Figures 10 and 11.

The calibration of ε for the ε-greedy baseline algorithm considered in Section 5.2.2 is done according to the time-average total pseudo-regret obtained at iteration $t = 150$. Figure 12 shows the marginal and total pseudo-regret curves for different values of $\varepsilon \in$

Kernel	φ	Norm	Pseudo-regret				% of wins			
			$t = 10$		$t = 50$		$t = 10$		$t = 50$	
			PA	**PB**	**PA**	**PB**	**PA**	**PB**	**PA**	**PB**
exponential	0.67	1	0.60	1.00	0.77	1.00	0.95	0.02	0.88	0.06
		2	0.56	1.02	0.66	1.00	0.98	0.01	0.98	0.02
	1.33	1	0.47	0.91	0.50	0.95	0.98	0.02	1.00	0.00
		2	0.41	0.89	0.41	0.92	1.00	0.00	1.00	0.00
	2	1	0.40	0.87	0.39	0.89	0.99	0.01	1.00	0.00
		2	0.36	0.81	0.34	0.84	0.98	0.02	1.00	0.00
	2.67	1	0.38	0.82	0.35	0.84	1.00	0.00	1.00	0.00
		2	**0.35**	0.79	0.31	0.79	0.99	0.01	1.00	0.00
	3.33	1	0.36	0.79	0.33	0.80	0.99	0.01	1.00	0.00
		2	**0.35**	0.76	0.30	0.76	0.99	0.01	1.00	0.00
	4	1	**0.35**	0.77	0.30	0.78	0.99	0.01	1.00	0.00
		2	**0.35**	0.77	**0.28**	0.74	0.99	0.01	1.00	0.00
gaussian	0.67	2	0.72	1.02	0.89	1.02	0.86	0.06	0.75	0.14
	1.33	2	0.52	0.91	**0.52**	0.96	0.97	0.03	0.99	0.01
	2	2	**0.48**	0.91	0.76	0.95	0.95	0.05	0.83	0.11
	2.67	2	0.77	0.95	0.86	0.97	0.67	0.21	0.68	0.21
	3.33	2	0.95	1.00	0.92	0.98	0.50	0.27	0.62	0.22
	4	2	1.02	1.04	1.02	0.98	0.44	0.27	0.42	0.32
Independent			$6.56 \cdot 10^3$		$1.60 \cdot 10^3$					

Table 5: For artificial instance 2, average (over all episodes) of the time-average total pseudo-regret, and percentage of episodes in which each method achieves the lowest total pseudo-regret among all three methods, at iterations $t = 10$ and $t = 50$. The pseudo-regret reported for IGM is normalized by z^*. The pseudo-regret reported for the spatial methods is normalized by the reported pseudo-regret for the IGM.

$\{0.1, 0.3, 0.5, 0.7, 0.9\}$. We observe that ε-greedy attains the best total pseudo-regret for $\varepsilon = 0.1$.

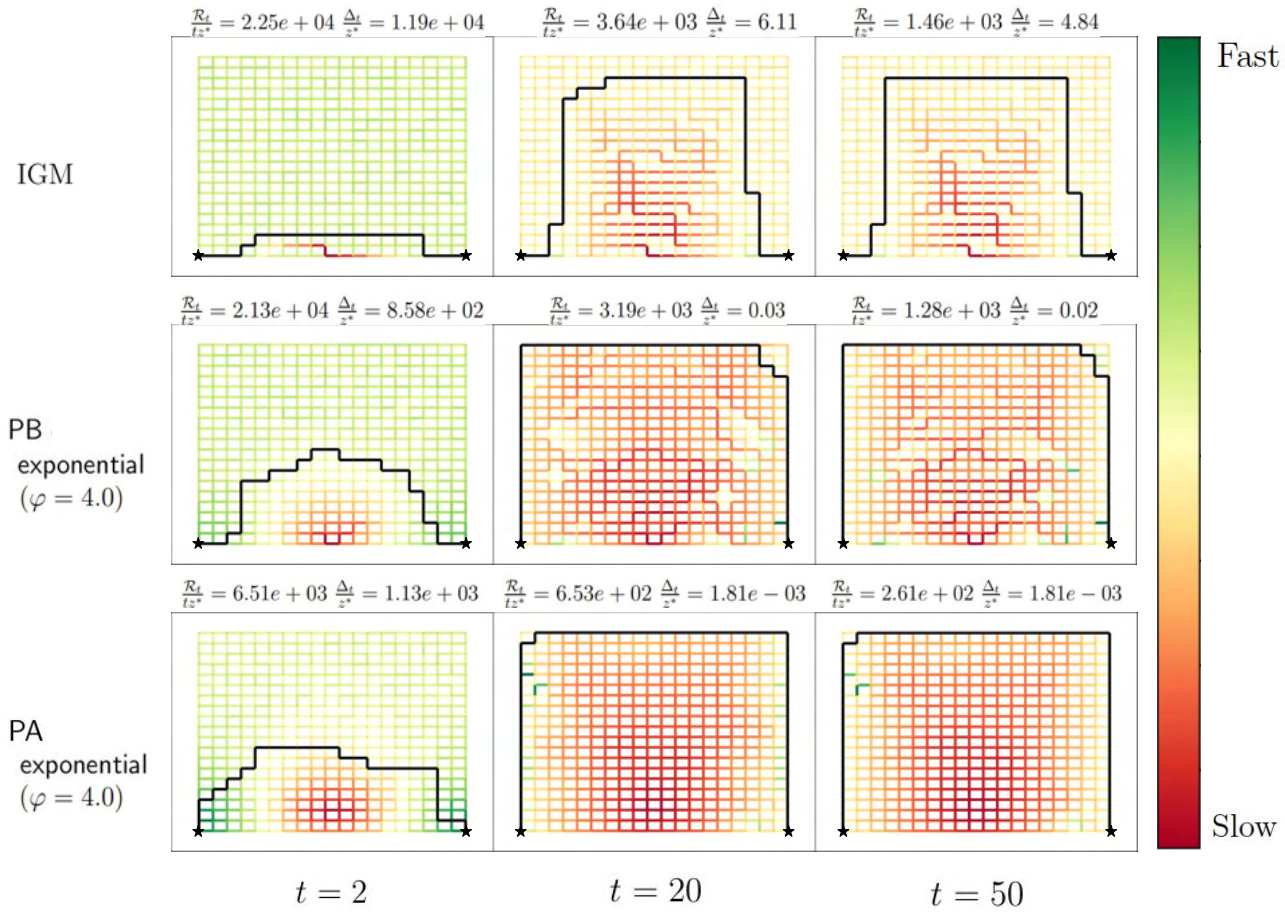


Figure 8: For artificial instance 2, selected path and associated normalized pseudo-regrets for all algorithms at different time periods.

Kernel	φ	Norm	Pseudo-regret				% of wins			
			$t = 10$		$t = 50$		$t = 10$		$t = 50$	
			PA	**PB**	**PA**	**PB**	**PA**	**PB**	**PA**	**PB**
exponential	0.67	1	0.60	0.90	0.83	0.99	0.99	0.01	1.00	0.00
		2	0.57	0.84	0.80	0.98	0.99	0.01	1.00	0.00
	1.33	1	0.48	0.65	0.78	0.88	0.94	0.06	0.97	0.03
		2	0.43	0.61	0.76	0.80	0.96	0.04	0.74	0.26
	2	1	0.40	0.58	0.76	0.75	0.94	0.06	0.45	0.55
		2	0.39	0.55	0.75	0.69	0.94	0.06	0.21	0.79
	2.67	1	0.38	0.54	0.76	0.67	0.92	0.08	0.11	0.89
		2	0.37	0.50	0.74	0.58	0.87	0.13	0.02	0.98
	3.33	1	**0.36**	0.50	0.76	0.59	0.87	0.13	0.02	0.98
		2	**0.36**	0.44	0.74	0.49	0.75	0.25	0.00	1.00
	4	1	**0.36**	0.46	0.76	0.53	0.80	0.20	0.01	0.99
		2	**0.36**	0.39	0.74	**0.43**	0.61	0.39	0.00	1.00
gaussian	0.67	2	0.67	0.96	**0.84**	1.00	0.96	0.03	1.00	0.00
	1.33	2	**0.55**	0.79	**0.84**	0.97	0.96	0.04	0.99	0.01
	2	2	0.58	0.73	0.88	0.94	0.80	0.19	0.87	0.13
	2.67	2	0.72	0.73	0.93	0.94	0.52	0.47	0.57	0.43
	3.33	2	0.79	0.77	0.97	0.96	0.42	0.56	0.41	0.53
	4	2	0.85	0.82	0.99	0.97	0.45	0.48	0.40	0.47
Independent			16.75		8.46					

Table 6: For artificial instance 3, average (over all episodes) of the time-average total pseudo-regret, and percentage of episodes in which each method achieves the lowest total pseudo-regret among all three methods, at iterations $t = 10$ and $t = 50$. The pseudo-regret reported for IGM is normalized by z^*. The pseudo-regret reported for the spatial methods is normalized by the reported pseudo-regret for the IGM.

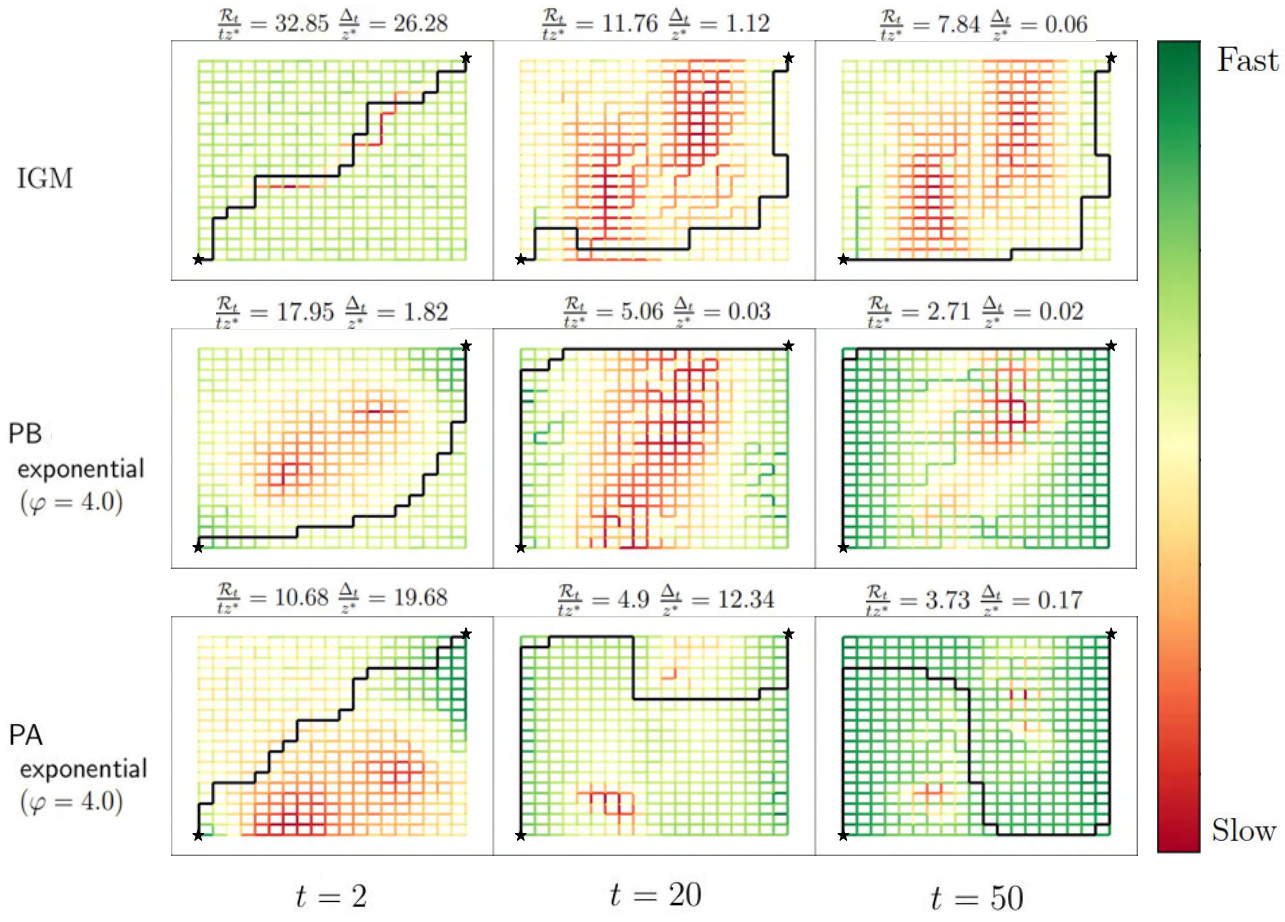


Figure 9: For artificial instance 3, selected path and associated normalized pseudo-regrets for all algorithms at different time periods.

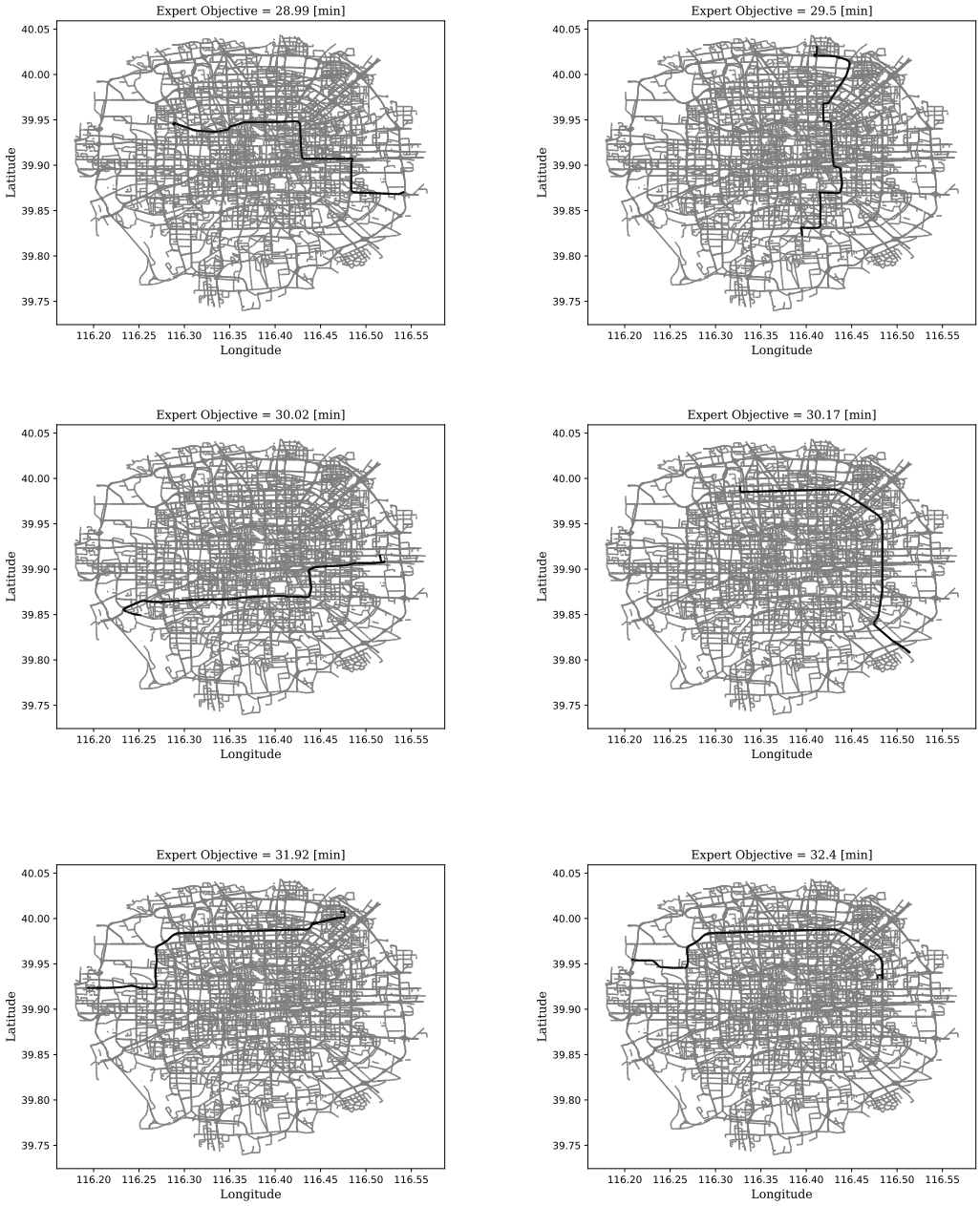


Figure 10: Expert's solution for multiple O-D pairs in the case study.

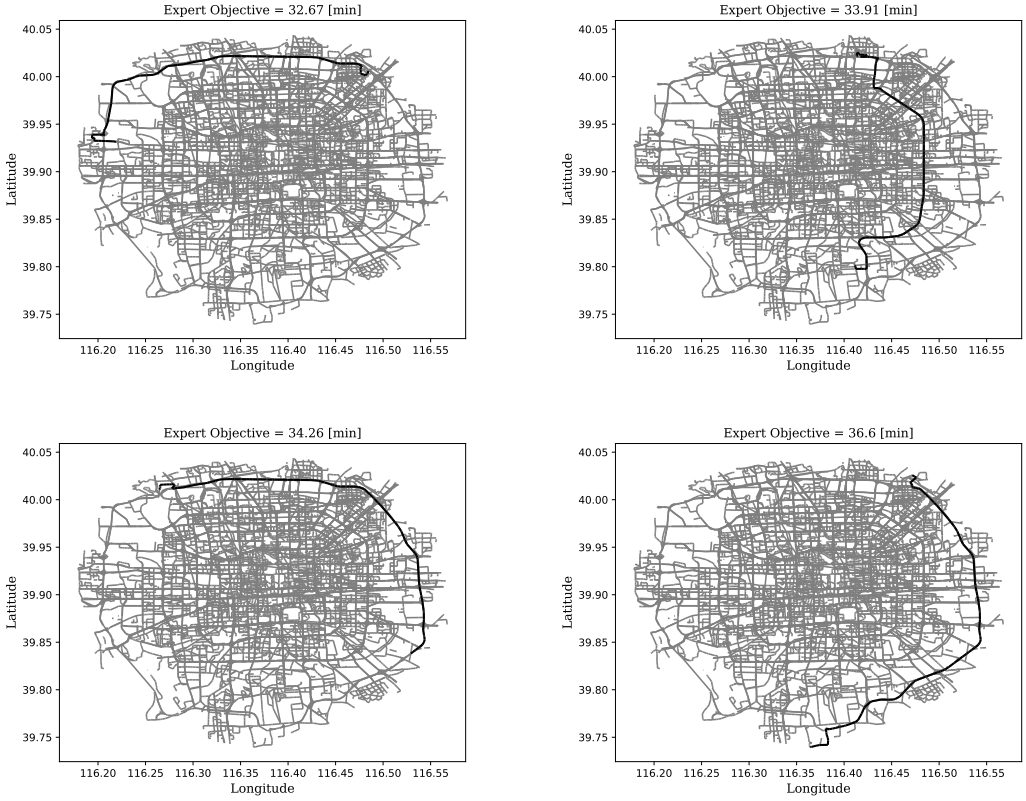


Figure 11: Expert's solution for multiple O-D pairs in the case study (continued).

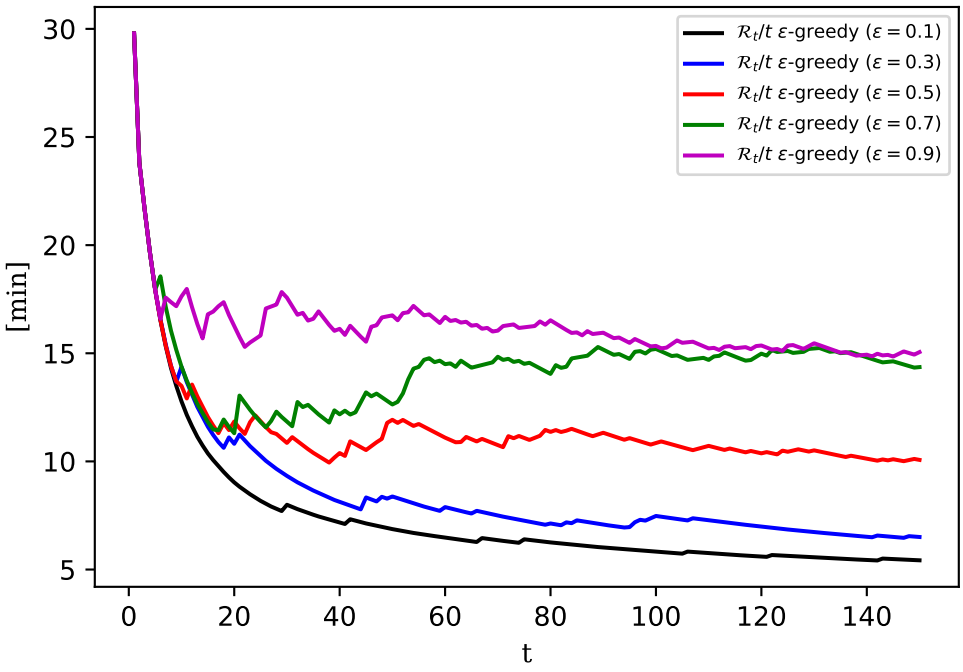


Figure 12: For ε-greedy , average pseudo-regrets obtained from 100 episodes of the Beijing case study with multiple origin-destination pairs (10 episodes from each of the 10 instances shown in Figures 10 and 11), for multiple values of ε. The best performance is attained for $\varepsilon = 0.1$.