A Contextual Combinatorial Semi-Bandit Approach to Network Bottleneck Identification

A Contextual Combinatorial Semi-Bandit Approach to Network Bottleneck Identification

Fazeleh Hoseini
fazeleh.hoseini@gmail.com
Chalmers University of Technology
Göteborg, Sweden

Niklas Åkerblom
niklas.akerblom@volvocars.com
Volvo Car Corporation
Göteborg, Sweden

Morteza Haghir Chehreghani
morteza.chehreghani@chalmers.se
Chalmers University of Technology
Göteborg, Sweden

ABSTRACT

Bottleneck identification is a challenging task in network analysis, especially when the network is not fully specified. To address this task, we develop a unified online learning framework based on combinatorial semi-bandits that performs bottleneck identification alongside learning the specifications of the underlying network. Within this framework, we adapt and investigate several combinatorial semi-bandit methods such as epsilon-greedy, LinUCB, BayesUCB, and Thompson Sampling. Our framework is able to employ contextual information in the form of contextual bandits. We evaluate our framework on the real-world application of road networks and demonstrate its effectiveness in different settings.

CCS CONCEPTS

• **Computing methodologies** → **Sequential decision making**; **Online learning settings**; **Machine learning**.

KEYWORDS

Contextual Bandits, Combinatorial Bandits, Bottleneck Identification, Online Learning

ACM Reference Format:

Fazeleh Hoseini, Niklas Åkerblom, and Morteza Haghir Chehreghani. 2024. A Contextual Combinatorial Semi-Bandit Approach to Network Bottleneck Identification . In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3679867

1 INTRODUCTION

Identifying bottlenecks is a critical task in network analysis, particularly in the study of traffic flow within road networks. In such networks, a bottleneck refers to the road segment with the highest cost along a path from a source node to a destination node. This cost or weight can be defined by various criteria, such as travel time. The goal is to find a path that minimizes the bottleneck among all possible paths connecting the source and destination nodes.

Bottleneck identification can thus be modeled as the task of computing the minimax edge in a given road network to find an edge with the smallest maximum edge over the paths connecting the source node and the destination node. By negating the edge weights, bottleneck identification can also be viewed as the widest path problem or the maximum capacity path problem [12].

Getting from one point of the network to another point via a path with the smallest bottleneck can be considered a sequential decision-making problem. In a classical minimax problem, we assume that the network parameters such as the edge weights are given. However, in some real-world cases, the edge weights are unknown from the beginning and have to be learned while solving the minimax problem. In this paper, we formulate the bottleneck identification task as a contextual combinatorial semi-bandit problem and propose an online learning approach to address it. Then, we evaluate our approach on a real-world application, road networks. We think of road navigation as playing bandit arms where each road segment is paired with a cost. We associate the bottleneck of a path with the most costly edge of the path and aim to find the path with the smallest bottleneck.

The Multi-Armed Bandit (MAB) model is a classic approach to online decision-making. A MAB agent repeatedly plays an arm and receives a stochastic reward drawn from an unknown reward distribution. In each step, the agent chooses an arm to play based on the observed reward of the previously played arms. The goal is to maximize the cumulative reward over a time horizon by making a trade-off between exploration and exploitation. The greedy algorithm is based on pure exploitation. Toward more exploration, we have the ϵ_t-greedy algorithm [5]. As the horizon tends to infinity, the ϵ_t-greedy algorithm guarantees that the actual value of the learnable parameter will be reached.

The Upper Confidence Bound (UCB) [4] algorithm is based on the idea of optimism in the face of uncertainty. It calculates an Upper Confidence Bound for each arm where the value is an overestimate of the learnable parameter with probability close to one. Later, [8] introduced BayesUCB which uses upper quantiles of the posterior distribution to compute the UCB. Thompson Sampling (TS) [17] is a classical approach to the trade-off between exploration and exploitation, assuming the existence of a prior distribution for the learnable parameter. Several studies, e.g., [1, 9, 15], demonstrate the performance of TS in different problems, including TS in combinatorial semi-bandit setting with the objective of finding the shortest path [19] and bottleneck [3] in a network (though these two works do not consider contextual information).

A *Contextual* bandit model is an extension of the standard MAB, in which the agent receives additional information, called context, for each arm, which determines the distribution of the reward. Assuming the expected reward is linear with the arm's context, [4]

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679867

proposed the contextual bandit algorithm LinRel. Later, [10] improved LinRel using a different form of regularization and proposed LinUCB. Studies in [13, 20] proposed a combinatorial MAB based on LinUCB, with application-specific objectives. [2] analyzes TS algorithm with linear reward. Recent work has introduced variance-adaptive algorithms and sampling methods to address challenges in combinatorial semi-bandit more effectively [18, 22].

In this paper, we propose a *Contextual* Combinatorial Semi-Bandit framework to identify bottlenecks in a network. Within this framework, we study ϵ_t-greedy, LinUCB, BayesUCB, and TS under *Contextual* Combinatorial Semi-Bandit conditions. We adapt these algorithms for a minimax path problem. We then demonstrate their performance on a real-world road network.

2 PROBLEM FORMULATION

In this section, we first introduce the problem of identifying bottlenecks in a road network. Then, assuming that the network is not fully specified, we define the probabilistic model of the problem.

2.1 Road Network Formulation

Consider a road network given as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$, where the nodes $\mathcal{V}$ represent intersections, and the edges $\mathcal{E}$ represent road segments. $\mathcal{G}$ is a directed graph, where edge $e = (u, v) \in \mathcal{E}$ is the edge leading from node $u \in \mathcal{V}$ to $v \in \mathcal{V}$. A weight function $w : \mathcal{E} \to \mathbb{R}^+$ maps edge e to its corresponding weight value $w(e)$.

We let the bottleneck of a single path refer to the edge on the path with the highest weight value. Consequently, given an arbitrary pair of source node $v \in \mathcal{V}$ and target node $u \in \mathcal{V}$, the bottleneck of all possible paths connecting v and u, is the smallest bottleneck among all possible paths connecting v to u. In other words, the bottleneck of v and u, denoted by $M(v, u, \mathcal{G})$, is the minimax distance between these two points, obtained by Eq. 1 where $p_{v,u}$ is a path connecting the source v and the target node u and $\mathcal{P}_{v,u}$ is the set of all paths between v and u. Note that to use the minimax distance, the weight function w is required to respect semi-metric constraints.

$$M(v, u, \mathcal{G}) = \min_{p_{v,u} \in \mathcal{P}_{v,u}} \max_{e \in p_{v,u}} w(e) \tag{1}$$

We apply a modification of Dijkstra's algorithm [6] to find the path including the bottleneck edge in the directed setting.

2.2 Probabilistic Model of the Road Network

In our problem setting, the edge weights of the network are uncertain and stochastic. Thus, we adopt a Bayesian approach to model the edge weights and make use of prior knowledge. Given a time horizon T, for each edge $e \in \mathcal{E}$ and time step $t \in [T]$, we assume access to a d-dimensional feature vector $\boldsymbol{c}_{t,e}$ representing contextual information related to the edge at that point in time. We further assume that the expected weight value is linear in $\boldsymbol{c}_{t,e}$, for some unknown parameter $\boldsymbol{\theta}_e^* \in \mathbb{R}^d$.

More specifically, we assume that, at time t, the weight $w_t(e)$, given $\boldsymbol{c}_{t,e}$, for $\forall e \in \mathcal{E}$ is sampled independently from a Gaussian distribution $\mathcal{N}(\boldsymbol{c}_{t,e}^\top \cdot \boldsymbol{\theta}_e^*, \varsigma_e^2)$ where the mean $\boldsymbol{\theta}_e^*$ is itself sampled from a prior distribution $\mathcal{N}(\boldsymbol{\mu}_{0,e}, \Sigma_{0,e})$. For simplicity of explanation, as in other works [2, 15], we assume ς_e^2 is known. With a Gaussian prior and likelihood, the posterior has the same parametric form as the prior, and the update of the parameters of the Gaussian posterior is derived in a closed form.

Algorithm 1 General Framework

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E}, w), v, u$

1: **for** $t \leftarrow 1, \ldots, T$ **do**
2: **for** $e \in \mathcal{E}$ **do**
3: Observe the edge context $\boldsymbol{c}_{t,e}$
4: Set algorithm-specific weight $\hat{w}_t(e)$ based on $\hat{\boldsymbol{\theta}}_{t,e}$ and $\boldsymbol{c}_{t,e}$
5: **end for**
6: Obtain path $p_{v,u}$ that contains $M(v, u, \mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}_t))$
7: **for** $e \in p_{v,u}$ **do**
8: Observe $w_t(e)$ by traversing edge e
9: Update the parameter estimate $\hat{\boldsymbol{\theta}}_{t,e}$
10: **end for**
11: **end for**

3 THE CONTEXTUAL COMBINATORIAL SEMI-BANDIT FRAMEWORK

We formulate the network bottleneck identification task as a contextual combinatorial semi-bandit problem. We have a set of base arms $\mathcal{A} = \{1, \ldots, K\}$, each of which corresponds to an edge in the graph. Furthermore, we consider a set of super-arms $\mathcal{I} \subseteq 2^{\mathcal{A}}$, corresponding to the set of all paths between a specified source node and target node. In each time step $t \in [T]$, for each base arm $a \in \mathcal{A}$, the environment reveals a contextual feature vector $\boldsymbol{c}_{t,a}$ to the agent. Subsequently, the agent selects a super-arm $\mathcal{S}_t \in \mathcal{I}$. For each base arm $a \in \mathcal{S}_t$, the environment reveals a loss sampled from $\mathcal{N}(\boldsymbol{c}_{t,a}^\top \cdot \boldsymbol{\theta}_a^*, \varsigma_a^2)$ to the agent, i.e., semi-bandit feedback.

The objective of the agent, in each time step t, is to find a super-arm $\mathcal{S}_t \in \mathcal{I}$ minimizing the maximum expected base arm loss, conditioned on the contextual information. This can be formalized as a regret minimization task, over the horizon T, where we define the expected cumulative regret as:

$$\text{Regret}(T) = \mathbb{E}\left[\left(\sum_{t=1}^{T} \max_{a \in \mathcal{S}_t} \boldsymbol{c}_{t,a}^\top \cdot \boldsymbol{\theta}_a^*\right) - \left(\sum_{t=1}^{T} \max_{a \in \mathcal{S}_t^*} \boldsymbol{c}_{t,a}^\top \cdot \boldsymbol{\theta}_a^*\right)\right], \tag{2}$$

where $\mathcal{S}_t^* = \arg\min_{\mathcal{S} \in \mathcal{I}} \max_{a \in \mathcal{S}} \boldsymbol{c}_{t,a}^\top \cdot \boldsymbol{\theta}_a^*$.

3.1 The Framework

We adapt and investigate several contextual combinatorial semi-bandit methods for the bottleneck identification problem. Algorithm 1 describes our general framework for this task. Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a source node $v \in \mathcal{V}$ and a target node $u \in \mathcal{V}$, the agent at each time step $t \in [T]$, for each edge $e \in \mathcal{E}$, observes the context vector $\boldsymbol{c}_{t,e}$. Each agent keeps and updates an internal representation of the graph $\mathcal{G}$, with weights $\hat{w}_t(e)$ assigned using the context $\boldsymbol{c}_{t,e}$ and stochastic weights $w_t(e)$ to induce exploration. Having observed $\boldsymbol{c}_{t,e}$, the agent subsequently updates the internal edge weight $\hat{w}_t(e)$ according to $\boldsymbol{c}_{t,a}$ and algorithm-specific parameter estimate $\hat{\boldsymbol{\theta}}_{t,e}$. Different algorithms have different approaches to update $\hat{w}_t(e)$, which we discuss later in this section. In line 6 the agent finds a path $p_{v,u}$ containing the bottleneck $M(v, u, \mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}_t))$ based on the modification of Dijkstra's algorithm. In lines 7 to 9 the agent

Algorithm 2 Thompson Sampling for bottleneck identification

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E}, w), v, u, \boldsymbol{\mu}_0, \Sigma_0, \varsigma^2$

1: **for** $t \leftarrow 1, \ldots, T$ **do**
2: **for** $e \in \mathcal{E}$ **do**
3: Observe context $\boldsymbol{c}_{t,e}$
4: $\hat{\boldsymbol{\theta}}_{t,e} \leftarrow$ Sample from posterior $\mathcal{N}(\boldsymbol{\mu}_{t-1,e}, \Sigma_{t-1,e})$
5: $\hat{w}_t(e) \leftarrow \boldsymbol{c}_{t,e}^\top \cdot \hat{\boldsymbol{\theta}}_{t,e}$
6: **end for**
7: $p_{v,u} \leftarrow$ Obtain a path that contains $M(v, u, \mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}_t))$
8: **for** $e \in p_{v,u}$ **do**
9: Observe $w_t(e)$ by traversing edge e
10: $\boldsymbol{\mu}_{t,e}, \Sigma_{t,e} \leftarrow$UPDATE-PARAMS$(e, w_t(e), \boldsymbol{\mu}_{t-1,e}, \Sigma_{t-1,e}, \varsigma_e^2)$
11: **end for**
12: **end for**

for each played edge e in the path, observes the stochastic edge weight (arm loss) $w_t(e)$ and then updates the parameters $\hat{\theta}_{t,e}$, and internal weights $\hat{w}_t(e)$ accordingly.

3.2 Adapted Algorithms

We adapt several methods to our contextual combinatorial semi-bandit framework.

- **Thompson Sampling:** In Thompson Sampling, we have a prior distribution $\mathcal{N}(\boldsymbol{\mu}_{0,e}, \Sigma_{0,e})$ for $\boldsymbol{\theta}_e^*$ where $\boldsymbol{\mu}_{0,e}$ is a d-dimensional column vector and $\Sigma_{0,e}^2$ is a scalar matrix $\lambda \mathbb{I}_d$ with a positive constant λ. Algorithm 2 describes the TS agent. In line 4, the agent samples $\hat{\boldsymbol{\theta}}_{t,e}$ from the posterior distribution $\mathcal{N}(\boldsymbol{\mu}_{t-1,e}, \Sigma_{t-1,e})$. Then, it updates $\hat{w}_t(e)$ with the expected weight given the sampled $\hat{\boldsymbol{\theta}}_{t,e}$. The TS agent updates the posterior parameters by UPDATE-PARAMS procedure shown in Algorithm 3.

- **BayesUCB:** We adapt the combinatorial BayesUCB in [3] for contextual setting, where the agent uses the upper quantiles of the posterior distributions over expected arm rewards to determine the level of exploration. The quantile function of a distribution ρ, denoted by $Q(\nu, \rho)$, is defined so that $\mathbb{P}_\rho(X \leq Q(\nu, \rho)) = \nu$. In Algorithm 4, describing BayesUCB, lower quantiles of the expected edge weights given the context is used in line 4. The BayesUCB agent makes use of UPDATE-PARAMS procedure.

- **LinUCB:** The LinUCB agent, described in Algorithm 5, introduces $A_{t,e} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b}_{t,e} \in \mathbb{R}^{d \times 1}$ with initialization of $\boldsymbol{b}_{0,e} = 0_{d \times 1}$ and $A_{0,e} = \mathbb{I}_d$ for each edge $e \in \mathcal{E}$. To find the UCB value for each edge first, it obtains the ridge regression estimate $\hat{\boldsymbol{\theta}}_{t,e} \leftarrow A_{t-1,e}^{-1} \boldsymbol{b}_{t-1,e}$. Then, having the exploration factor α, the agent calculates the current edge weight values. Since our problem is formulated as a loss minimization, we negate the α term. Updating the parameter in LinUCB follows lines 11 and 12.

Algorithm 3 Bayesian posterior parameter update procedure

1: procedure UPDATE-PARAMS$(\boldsymbol{c}_{t,e}, w_t(e), \boldsymbol{\mu}_{t-1,e}, \Sigma_{t-1,e}, \varsigma_e^2)$
2: $\Sigma_{t,e} \leftarrow \left(\Sigma_{t-1,e}^{-1} + \frac{1}{\varsigma_e^2} \boldsymbol{c}_{t,e} \boldsymbol{c}_{t,e}^\top\right)^{-1}$
3: $\boldsymbol{\mu}_{t,e} \leftarrow \Sigma_{t,e} \left(\Sigma_{t-1,e}^{-1} \boldsymbol{\mu}_{t-1,e} + \frac{1}{\varsigma_e^2} w_t(e) \boldsymbol{c}_{t,e}\right)$

Algorithm 4 BayesUCB for bottleneck identification

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E}, w), v, u, \boldsymbol{\mu}_0, \Sigma_0, \varsigma^2$

1: **for** $t \leftarrow 1, \ldots, T$ **do**
2: **for** $e \in \mathcal{E}$ **do**
3: Observe context $\boldsymbol{c}_{t,e}$
4: $\hat{w}_t(e) \leftarrow \boldsymbol{c}_{t,e}^\top \boldsymbol{\mu}_{t-1,e} - Q(1 - \frac{1}{t}, \mathcal{N}(0,1)) \sqrt{\boldsymbol{c}_{t,e}^\top \Sigma_{t-1,e} \boldsymbol{c}_{t,e}}$
5: **end for**
6: $p_{v,u} \leftarrow$ Obtain a path that contains $M(v, u, \mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}_t))$
7: **for** $e \in p_{v,u}$ **do**
8: Observe $w_t(e)$ by traversing edge e
9: $\boldsymbol{\mu}_{t,e}, \Sigma_{t,e} \leftarrow$UPDATE-PARAMS$(e, w_t(e), \boldsymbol{\mu}_{t-1,e}, \Sigma_{t-1,e}, \varsigma_e^2)$
10: **end for**
11: **end for**

- ϵ_t**-greedy:** We adapt the ϵ_t-greedy in [5] to our problem setting, with $\epsilon_t = t^{-1/2}$. The Greedy agent estimates $\hat{\boldsymbol{\theta}}_{t,e}$ as the mode of the posterior distribution $\mathcal{N}(\boldsymbol{\mu}_{t,e}, \Sigma_{t,e})$, and then updates the edge by $\boldsymbol{c}_{t,e}^\top \hat{\boldsymbol{\theta}}_{t,e}$. Then, with probability $1 - \epsilon_t$, the agent applies the modification of Dijkstra's algorithm, and with probability ϵ_t it chooses an arbitrary path containing a uniformly random edge. The update of the parameters follows Algorithm 3.

An expanded version, including the extension of NeuralUCB [21] to our problem and additional experiments, is available in [7].

4 EXPERIMENTS

Here, we evaluate our framework in a real-world road network dataset. We use the simulation framework in [16], adapt it to road networks and extend it to the contextual combinatorial setting.

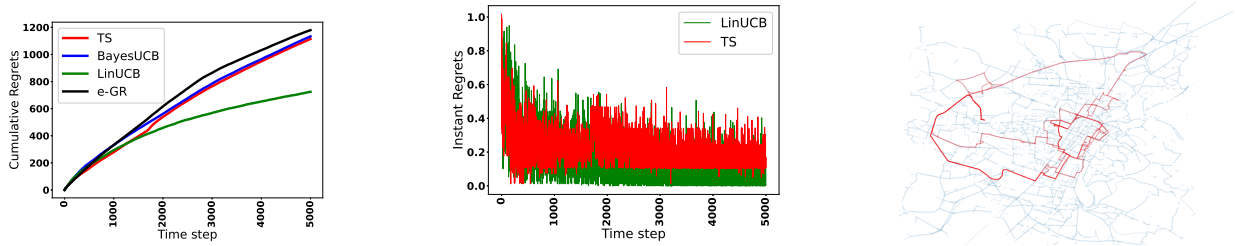
4.1 Road Network Dataset

To evaluate our method, we used data from the Turin SUMO Traffic (TuST) Scenario [11, 14]. TuST describes the traffic of Turin during one day. It contains 14,226 intersections and 18,741 road segments. For each segment, information is provided such as the IDs of two

Algorithm 5 LinUCB for bottleneck identification

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E}, w), v, u, d, \alpha, \varsigma_e^2$

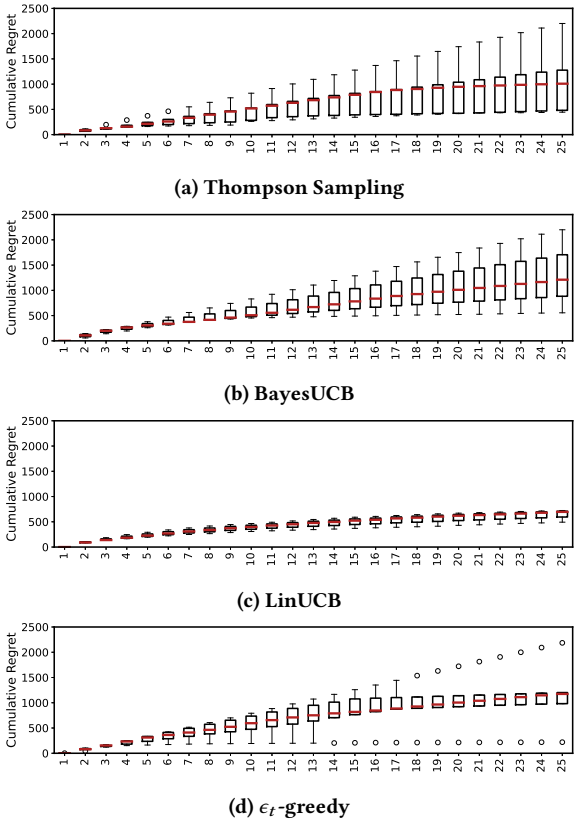
1: For all $e \in \mathcal{E}$ initialize $A_{0,e} \leftarrow \mathbb{I}_d$, and $\boldsymbol{b}_{0,e} \leftarrow 0_{d \times 1}$
2: **for** $t \leftarrow 1, \ldots, T$ **do**
3: **for** $e \in \mathcal{E}$ **do**
4: Observe context $\boldsymbol{c}_{t,e}$
5: $\hat{\boldsymbol{\theta}}_{t,e} \leftarrow A_{t-1,e}^{-1} \boldsymbol{b}_{t-1,e}$
6: $\hat{w}_t(e) \leftarrow \hat{\boldsymbol{\theta}}_{t,e}^\top \boldsymbol{c}_{t,e} - \alpha \sqrt{\boldsymbol{c}_{t,e}^\top A_{t-1,e}^{-1} \boldsymbol{c}_{t,e}}$
7: **end for**
8: $p_{v,u} \leftarrow$ Obtain a path that contains $M(v, u, \mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}_t(e))$
9: **for** $e \in p_{v,u}$ **do**
10: Observe $w_t(e)$ by traversing edge e
11: $A_{t,e} \leftarrow A_{t-1,e} + \frac{1}{\varsigma_e^2} \boldsymbol{c}_{t,e} \boldsymbol{c}_{t,e}^\top$
12: $\boldsymbol{b}_{t,e} \leftarrow \boldsymbol{b}_{t-1,e} + \frac{1}{\varsigma_e^2} w_t(e) \boldsymbol{c}_{t,e}$
13: **end for**
14: **end for**



(a) Cumulative regrets (b) Instant regrets of LinUCB and TS (c) Explored paths by LinUCB

Figure 1: Average of the cumulative regrets of TS, BayesUCB, LinUCB and ϵ_t-greedy (e-GR) over 10 runs (a), instant regrets of the LinUCB and TS agents (b), and the road segments that the LinUCB agents explored in Turin road network (c).

intersections connected by the road segment, the length of the intersection, the speed limit, and the mean and variance of the speed of vehicles passing the segment at different hours of the day. Since the simulation data for 24 hours are extensive, we use speed information of 5 hours as the segments' contextual information. Then, $d = 5$, and for each $e \in \mathcal{E}$, $c_{t,e}^{(i)} \sim \mathcal{N}(m_{i,e}, s_{i,e}^2)$ for $i = 1, \ldots, 5$, where $m_{i,e}$ and $s_{i,e}^2$ are the mean and variance of the speed at edge e in the i-th hour of a day.



(a) Thompson Sampling

(b) BayesUCB

(c) LinUCB

(d) ϵ_t-greedy

Figure 2: The boxplot of the cumulative regrets for the mentioned agents over 10 runs. The x-axes show the horizon. We plotted a box for every 200 time steps.

4.2 Experimental Results

For the agents that make use of the prior, for each edge $e \in \mathcal{E}$ with edge length l_e, we initialize $\boldsymbol{\mu}_{0,e}$ with a $d-$dimensional all-ones vector multiplied by λ_e where $\lambda_e = l_e \cdot 10^{-2}$ and $\Sigma_{0,e}$ is $\lambda_e \cdot 5 \cdot \mathbb{I}_d$. We set $\varsigma_e = 1$. We fix the horizon to $T = 5000$. For LinUCB we set the exploration factor $\alpha = 4$. We repeat all experiments for 10 runs.

Figure 1a shows the average cumulative regrets of the agents over 10 runs. The LinUCB agent has the lowest cumulative regret, which stabilises faster compared to the others. Then, the BayesUCB and TS agents perform almost equally, and the ϵ_t-greedy agent has the highest cumulative regret. Figure 1b illustrates the average instant regrets of the LinUCB and TS agents, which decrease as we progress in time. The LinUCB achieves the lowest regrets compared to TS. Figure 1c shows Turin road network, where the explored paths by the LinUCB agent are colored red, and the opacity illustrates the amount of exploration of each of the road segments.

Figure 2 shows the boxplot of the cumulative regrets where the y-axes show the cumulative regrets and x-axes show the horizon. We plotted a box for every 200 time steps. So, the labels of x-axes times 200 is the actual t. The red lines show the median and the boxes mark the upper and lower quantiles, with outliers shown as points. LinUCB has the smallest quantile boxes, suggesting that LinUCB behaves almost the same in different runs. It also has the lowest median. ϵ_t-greedy has relatively small quantile boxes with outlier points that show occasional random behavior of this agent. BayesUCB and TS behave almost the same regarding the sizes of the quantile boxes. The BayesUCB and TS agents compete closely with each other, and the ϵ_t-greedy agent follows other agents.

5 CONCLUSION

We developed a unified contextual combinatorial bandit framework for network bottleneck identification which takes into account the available side information. We used a minimax concept to find the path with the bottleneck edge. We adapted Thompson Sampling, BayesUCB, LinUCB and ϵ_t-greedy to our framework. We evaluated its performance on a real-world road network application, and demonstrated that LinUCB outperforms the other agents.

6 ACKNOWLEDGMENTS

This work is partially funded by the Strategic Vehicle Research and Innovation Programme (FFI) of Sweden, through the project EENE (reference number: 2018-01937).

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2012. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*. JMLR Workshop and Conference Proceedings, PMLR, Edinburgh, Scotland, 39–1.

[2] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*. PMLR, PMLR, Atlanta, Georgia, USA, 127–135.

[3] Niklas Åkerblom, Fazeleh Sadat Hoseini, and Morteza Haghir Chehreghani. 2023. Online learning of network bottlenecks via minimax paths. *Machine Learning* 112, 1 (2023), 131–150. https://doi.org/10.1007/s10994-022-06270-0

[4] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[5] Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*. PMLR, PMLR, Atlanta, Georgia, USA, 151–159.

[6] Edsger W Dijkstra et al. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[7] Fazeleh Hoseini, Niklas Åkerblom, and Morteza Haghir Chehreghani. 2024. A contextual combinatorial semi-bandit approach to network bottleneck identification. *arXiv preprint arXiv:2206.08144* (2024).

[8] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. 2012. On Bayesian upper confidence bounds for bandit problems. In *Artificial intelligence and statistics*. PMLR, PMLR, La Palma, Canary Islands, 592–600.

[9] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. 2012. Thompson sampling: An asymptotically optimal finite-time analysis. In *International conference on algorithmic learning theory*. Springer, Springer-Verlag, Berlin, Heidelberg, 199–213.

[10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. Association for Computing Machinery, New York, NY, USA, 661–670.

[11] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO, In The 21st IEEE International Conference on Intelligent Transportation Systems. *IEEE Intelligent Transportation Systems Conference (ITSC)*. https://elib.dlr.de/124092/

[12] Maurice Pollack. 1960. The maximum capacity through a network. *Operations Research* 8, 5 (1960), 733–736.

[13] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. 2014. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 461–469.

[14] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. 2021. Vehicular Traffic Simulation in the City of Turin from Raw Data. *IEEE Transactions on Mobile Computing* (2021), 1–12. https://doi.org/10.1109/TMC.2021.3075985

[15] Daniel Russo and Benjamin Van Roy. 2014. Learning to optimize via posterior sampling. *Mathematics of Operations Research* 39, 4 (2014), 1221–1243.

[16] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning* 11, 1 (2018), 1–96.

[17] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3-4 (1933), 285–294.

[18] Ruo-Chun Tzeng, Po-An Wang, Alexandre Proutiere, and Chi-Jen Lu. 2024. Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits. *Advances in Neural Information Processing Systems* 36 (2024).

[19] Siwei Wang and Wei Chen. 2018. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*. PMLR, PMLR, 5114–5122.

[20] Chenxi Zhang, Pinyi Ren, and Qinghe Du. 2017. A contextual multi-armed bandit approach to caching in wireless small cell network. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 1–6.

[21] Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural Contextual Bandits with UCB-based Exploration. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*. PMLR, 11492–11502.

[22] Julien Zhou, Pierre Gaillard, Thibaud Rahier, Houssam Zenati, and Julyan Arbel. 2024. Covariance-Adaptive Least-Squares Algorithm for Stochastic Combinatorial Semi-Bandits. *arXiv preprint arXiv:2402.15171* (2024).