CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Online Learning for Energy Efficient Navigation using Contextual Information

Master's thesis in Computer science and engineering

YONCA YUNATCI
AHMET BATIKAN UNAL

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Online Learning for Energy Efficient Navigation using Contextual Information

YONCA YUNATCI
AHMET BATIKAN UNAL

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Online Learning for Energy Efficient Navigation using Contextual Information
YONCA YUNATCI and AHMET BATIKAN UNAL

Supervisor: Niklas Åkerblom, Department of Computer Science and Engineering
Advisor: Niklas Åkerblom, Volvo Cars
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Online Learning for Energy Efficient Navigation using Contextual Information

YONCA YUNATCI
AHMET BATIKAN UNAL
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Accurately predicting the energy consumption of road segments is an important topic in electric vehicles that might alleviate the range concerns if it is addressed properly. We employ a contextual combinatorial multi-armed bandit framework to learn the unknown parameters of an energy consumption model which is necessary for energy-efficient navigation. Four different agents: Thompson Sampling, Disjoint LinUCB, Hybrid LinUCB, and greedy algorithms are implemented to observe their performance. All experiments are conducted on the output of a Luxembourg SUMO traffic simulation. The main finding of this research is that contextual information such as speed and acceleration data contributes to better learning of parameters. Although the contextual combinatorial algorithms seem promising for addressing the energy-efficient shortest path problem, none of the agents achieve zero regret consistently which indicates that further improvements are necessary to obtain the desired results.

Keywords: Contextual combinatorial multi-armed bandit, online learning, electric vehicles, energy consumption prediction, computer science.

Acknowledgements

We would like to thank our supervisor Niklas Åkerblom for his guidance and his valuable feedbacks during this thesis work. We are also thankful to Volvo Cars for giving us the opportunity to work on this project and the team for making us feel welcomed. Lastly, we would like to thank our families and friends who support us unconditionally.

Yonca Yunatci and Ahmet Batikan Unal, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1
Introduction

1.1 Motivation

The development of electric vehicles (EV) has increased very rapidly in the last decade due to growing interest towards both EV and environmental consciousness. However, battery capacities still pose an obstacle for higher demand. Despite the presence of fast chargers, long charging times compared to fuel refilling is another obstacle for electric vehicles to replace conventional vehicles. We can alleviate these concerns by taking advantage of data science to predict energy consumption to construct more robust route planning methods.

Volvo Cars currently has several plug-in hybrid models and aims to transform the conventional combustion engine models into electrified versions. For all these vehicles, they can benefit from having route planning algorithms to calculate the most efficient, thus the most environment-friendly path.

1.2 Problem Description

Our aim is to address Energy-Optimal Shortest Path Problem in electric vehicles, which is choosing the best path with respect to energy consumption while satisfying battery constraints. When the energy consumption associated with each road segment is known in advance, it is fairly easy to address a navigation problem using some existing shortest path algorithm. Even though it is possible to calculate the theoretical energy consumption of a road segment, the actual consumption can vary from theoretic results and thus cause unreliable predictions of remaining energy. Instead, this paper investigates how user consumption data can be utilized to better predict the energy consumption of a road segment with the help of *online learning* methods. The advantage of adopting an online learning method is that we do not have to re-train the model with each newly arrived data, since these methods process the data sequentially.

In order to solve this problem, we aim to map the problem into a multi-arm bandit problem which is an online learning method and involves probabilistic energy consumptions represented by edge weights. These edge weights will later be used in addressing the shortest path problem. Our intended outcome is to minimize the regret that we get while learning the parameters of these distributions. The first question that we try to address throughout the thesis is whether training the

multi-armed bandit by only using observed energy consumption yields a sufficient approximation or whether we can benefit from introducing contextual information, such as speed and acceleration. Secondly, we will evaluate different learning strategies in the multi-armed bandit setting.

1.3 Scope

We simplify the energy-optimal shortest path problem to only concern energy consumption and disregard battery constraints. We focus on navigation in the city of Luxembourg. We intend to see how existing contextual bandit methods in literature can be adapted to address the energy-optimal shortest path problem. We will also assess the degree of benefit from introducing contextual information to the algorithms.

1.4 Thesis Outline

Chapter 2 presents some background information about energy consumption in electric vehicles and multi-armed bandits, collected from several studies in literature. The process of preparing data to be fed into the algorithms is described in chapter 3. The details of algorithms are presented in chapter 4. Chapter 5 contains resulting figures and plots with explanations and discussions. Finally, we present future work and reach a conclusion in chapter 6.

2
Background

2.1 Energy Consumption Estimation of Electric Vehicles

In order to address energy-efficient shortest path problem, a deterministic approach can be to compute the theoretical consumptions of road segments as edge weights. There are several approximation methods of energy consumption in electric vehicles but they either lack precision or require detailed driving cycles (Demir et al., 2014). However, these approximations can still be useful to determine the contexts in our algorithms. The approximation proposed by Basso et al. (2019) involves approximated constant acceleration $\overline{a}$, approximated inclination angle $\overline{\theta}$ over the road segment length d, and initial and final speeds (v_i,v_f). There are also some vehicle specific parameters required for a better approximation, such as rolling resistance coefficient C_r, drag coefficient C_d, frontal surface area of the vehicle A and powertrain efficiency η. Apart from these parameters, air density ρ and gravitational constant g are also important factors that affect energy consumption. Putting all together, the equation 2.1 shows the total electric energy need (Wh) to complete a certain road segment.

$$e = \frac{m\overline{a}d + mgd\sin\overline{\theta} + mgC_r d\cos\overline{\theta} + 0.5C_d A\rho d(v_f^2 + (v_f^2 - v_i^2)/2)}{3600\overline{\eta}} \tag{2.1}$$

We can infer from equation 2.1 that squared speed and acceleration are proportional to the estimated energy consumption.

2.2 Multi-armed Bandits

A probabilistic approach to the energy-optimal shortest path problem can be achieved by a framework called *multi-armed bandits*. This framework enables us to model edge weights as probability distributions of energy consumption which may help us to capture the energy consumption pattern that may be missed by the approximation methods.

The original multi-armed bandit problem is formulated as a gambler trying to get the maximum benefit from a slot machine through experimenting. The slot machine, i.e., bandit is assumed to have several arms with different probability distribution of rewards. The aim is to estimate these distribution parameters and maximize the

expected reward at the end of T steps. Initially, the gambler has no information of which arm yields more reward. While he tries to learn the best benefiting arm, he faces a dilemma of "explore-exploit". Exploiting the best arm according to our best knowledge may not always give the best reward in the long-term since the parameter estimations at that time may be far from reality. If the gambler chooses to explore, he may gain more knowledge about the reward distributions, but it may result in less short-term reward. For those reasons, neither pure exploration (random strategy) nor pure exploitation (greedy strategy) is a valid option for solving the problem effectively. There are several algorithms to address this explore-exploit dilemma which will be discussed in the following section. But let us first introduce some technical terms and discuss the multi-armed bandit problem formally.

The multi-armed bandit problem is described as a game between a *learning agent* and an *environment* in literature (Lattimore & Szepesvári, 2020, ch. 1). The learning agent, also referred to as an *algorithm*, observes the environment to choose an *action* (namely *arm*) in each round. According to the chosen action in each round, the environment reveals a *reward* for that action. The aim of an agent is to adopt a suitable *policy* to maximize the received reward over some time period. The policy is formally defined as a mapping from *history* (a sequence of observed *(action, reward)* pairs) to actions (Lattimore & Szepesvári, 2020, ch. 1). At the end of each round, *instant regret* is calculated by subtracting the expected reward of the played arm from the expected reward of the best arm. *Cumulative regret* is calculated by adding those instant regrets at the end of some time period to evaluate the performance of a learner.

Consider a bandit problem with a set $\mathcal{A}$ of K possible arms to pull. Each arm a has a probability distribution of rewards with mean μ_a (also expected reward). It is important to note that all rewards are assumed to be independent of each other. Let $a_t^* = \text{argmax}_{a \in \mathcal{A}} \mu_a$ be the optimal arm at time t. Every agent/algorithm has a different way to choose the best arm. Let arm a_t be the played arm at time t according to some algorithm. Then, instant regret is calculated as:

$$\mu_{a_t^*} - \mu_{a_t} \tag{2.2}$$

and cumulative regret after T rounds is:

$$\sum_{t=1}^{T} (\mu_{a_t^*} - \mu_{a_t}) \tag{2.3}$$

The overall intention is to estimate the parameters as close as possible to the true values of actions while keeping cumulative regret at the end of T rounds as low as possible.

2.3 Algorithms to Address Explore-Exploit Dilemma

2.3.1 Greedy Algorithms

Since pure exploitation is not an effective solution as we argued in the previous section, ϵ-greedy is one possible method to enforce some exploration (Sutton & Barto, 1998, ch. 2). With small probability ϵ, the algorithm chooses arms randomly which corresponds to exploring the probability distribution of a sub-optimal arm. The natural implication of this statement is that the algorithm exploits the arm with highest expected reward the rest of the time (i.e. with $1 - \epsilon$ probability).

Algorithm 1 ϵ-greedy

Input: ϵ
  for t=1...T **do**
    pick random real number $N \in [0, 1]$
    if $N < \epsilon$ **then**
      select an arm at random $a_t \in \mathcal{A}$
    else
      for all $a \in \mathcal{A}$ **do**
        estimate the mean $\hat{\mu}_a$
      end for
      choose arm $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{\mu}_a$
    end if
    apply a_t and observe reward r_{t,a_t}
    update the distribution of a_t
  end for

Besides its simplicity, another advantage of this algorithm is, as the number of trials goes to infinity, every arm will be explored infinitely many times. Thus, in theory, we would be able to learn the actual value of every action. However, in many real-life problems, the dataset is limited, and we are more interested in learning the optimal action. Thus, wasting time on learning the actual values of the arms that are far from being optimal decreases the efficiency of the algorithm. This is because the ϵ-greedy algorithm explores completely at random.

2.3.2 Thompson Sampling

Thompson sampling (Thompson, 1933) was introduced in 1933 as a method for balancing the explore-exploit dilemma in a smarter way compared to randomly selecting arms. Initially, Thompson sampling assumes that a prior distribution is chosen to represent the prior knowledge of the reward distribution of arms. In each round, the Thompson sampling algorithm samples parameters from the posterior distribution of the reward for each arm and uses these sampled parameters to decide the best choice, i.e., the arm the highest sampled expected reward (Lattimore & Szepesvári, 2020, ch. 7). With each reward that the algorithm gets from the chosen action, the relevant distribution is updated with respect to Bayes rule.

Algorithm 2 Thompson Sampling

for t=1...T **do**
  for all $a \in \mathcal{A}$ **do**
    sample mean $\hat{\mu}_a$ from posterior distribution
  end for
  choose arm $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{\mu}_a$ and observe reward r_{t,a_t}
  update the distribution of a_t
end for

As the arms get selected, their posterior mean is expected to get closer to the true mean and their posterior variance is expected to get smaller. As a consequence, an arm with a low mean and a large variance has a chance of being pulled instead of an arm with a high mean and a small variance. However, it is unlikely that an arm with a low mean and a small variance is selected since it has been sampled enough to determine that is far from being the optimal arm. Therefore, this algorithm allows us to make more reasonable choices in exploration due to the uncertainty involved in the decision process. Consider the following example with 3 options demonstrated in Figure 2.1.

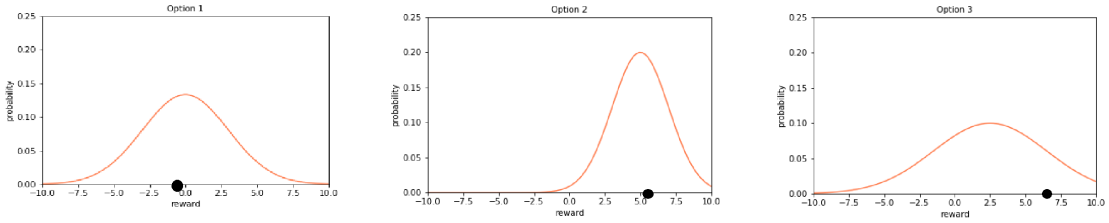


Figure 2.1: An example of the posterior distributions of the expected rewards of 3 arms after T steps with sampled means

In the current setting, the posterior mean of option 2 is higher than any other options, but option 3 has a higher sampled expected reward caused by its large uncertainty, in other words, large variance. This means that option 3 also has a high probability of being the optimal arm but we do not know whether it has a higher true mean reward than option 2 because we did not sample option 3 enough.

2.3.3 Upper Confidence Bound Algorithms (UCB)

Upper Confidence Bound algorithms propose a way to select the action by assessing their potential to be the optimal arm (Sutton & Barto, 1998, ch. 2). The decision process involves both the expected reward and a confidence level associated with each arm. The action is selected based on the following formula:

$$\underset{a \in \mathcal{A}_t}{\text{argmax}}\, \hat{\mu}_a + \sqrt{\frac{\alpha \ln t}{n_{t,a}}} \tag{2.4}$$

where the first part of the formula ($\hat{\mu}_a$) is the current estimation of the mean reward for arm a and the second part of the formula corresponds to a measure of uncertainty.

In the calculation of uncertainty, t is time, $n_{t,a}$ is the count for how many times arm a has been pulled until time t and α is the parameter to control the degree of exploration (ch. 1 Nicol, 2014; Sutton & Barto, 1998, ch. 2). The overall algorithm is presented in algorithm 3.

Algorithm 3 UCB

Input: α
  $n_a \leftarrow 0$
  for t=1...T **do**
    for all $a \in \mathcal{A}$ **do**
      estimate the mean $\hat{\mu}_a$
    end for
    choose arm $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{\mu}_a + \sqrt{\frac{\alpha \ln t}{n_{t,a}}}$ and observe reward r_{t,a_t}
    $n_{a_t} \leftarrow n_{a_t} + 1$
    update the distribution of a_t
  end for

Pulling an arm less often results in higher uncertainty. Therefore, the algorithm tends to favor the less explored arms and enforces optimism in the decision process. Compared with the ϵ-greedy algorithm, introducing confidence intervals enables us to make more sensible selection of arms for exploration (Li et al., 2010). Again, consider the same example but this time with the UCB approach in Figure 2.2:

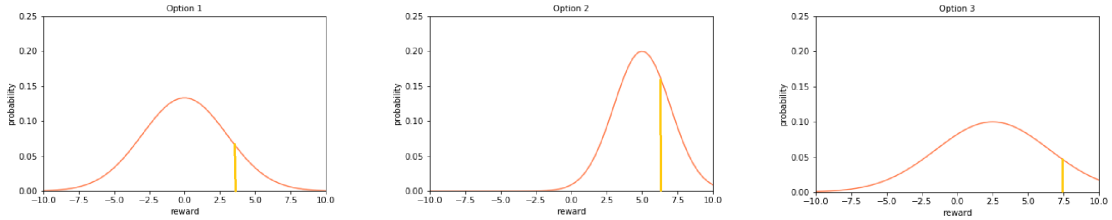


Figure 2.2: An example of the posterior distributions of the expected rewards of 3 arms with the upper confidence bounds after T steps

The yellow lines are calculated by adding the estimated mean (indicated with dashed line) to the uncertainty term as described in formula 2.4 above. The values corresponding to these yellow lines are used for the decision process. Again, the algorithm favors option 3 over option 2 because of the algorithm's optimism towards not-sufficiently-explored arms.

2.4 Combinatorial Semi-Bandits

In the original multi-armed bandit setting, only one arm is selected and pulled by the agent. However, it is not suitable for some applications that require pulling more than one arm at the same time. Recently, combinatorial multi-armed bandits are actively being studied by scientists to be able to adapt the multi-armed bandit framework to different applications with different requirements. One example for

such applications is presented by Qin et al. (2014) suggesting that many real life recommendation systems actually yield a set of recommendations rather than just one. Such a requirement corresponds to pulling a set of arms in each round in the multi-armed bandit setting. A combinatorial multi-armed bandit setting enables an agent to pull more than one arm at once, which is called a *super arm*. The arms in the played super arm are called *base arms*. The agent can either observe only the reward for the super arm or observe individual rewards for each base arm in the super arm. If the agent receives individual rewards for each base arm, then such a setting is called a *semi* bandit (Wang & Chen, 2018).

2.5 Contextual Bandits

The multi-arm bandit problem can be extended to consider additional information while deciding on the action. Several real-world problems such as the user-oriented movie recommendation problem can benefit from additional information such as users' age, gender, previously watched movies, etc., to be able to provide more relevant recommendations (Lattimore & Szepesvári, 2020, ch. 5).

Additional information, namely context, is associated with each arm and observed in each iteration before picking an action that maximizes the expected reward. Again, the aim is to learn a good policy that yields less cumulative regret at the end of T rounds; but here, the policy is defined as a mapping from contexts to actions. Suppose the contextual information is represented by a d-dimensional feature vector $x_{t,a}$ for arm a at time t and θ_a^* is a d-dimensional coefficient vector of arm a (Deshmukh et al., 2017). Then, the expected reward is calculated as:

$$E(r_{t,a}|x_{t,a}) = x_{t,a}^T \theta_a^* \tag{2.5}$$

To indicate each step separately for solving a contextual bandit problem, the algorithm for the contextual bandits is presented below (Deshmukh et al., 2017):

Algorithm 4 General Algorithm for Contextual Bandits

```
for t=1...T do
  for all a ∈ A do
    estimate the parameters θ̂_a
    observe context x_{t,a}
  end for
  choose arm a_t = argmax_{a∈A} x_{t,a}^T θ_a^* and observe reward r_{t,a_t}
  update the distribution of a_t
end for
```

2.5.1 Contextual Thompson Sampling

Like in the context-free setting, the Thompson sampling algorithm assumes a prior distribution of rewards initially. In each round, the agent samples a set of parameters $\hat{\theta}_a$ from posterior distribution. Since the agent is able to observe the context x_a,

it can calculate the estimated expected reward for each arm with these sampled parameters and the observed context vector according to formula 2.5. Then, it selects the best arm that maximizes the expected reward. Algorithm 5 demonstrates this procedure.

Algorithm 5 Contextual Thompson Sampling

```
for t=1...T do
  for all a ∈ A do
    sample parameters θ̂_a from posterior distribution
    observe context x_{t,a}
  end for
  choose arm a_t = argmax_{a∈A} x_{t,a}^T θ̂_a and observe reward r_{t,a_t}
  update the distribution of a_t
end for
```

2.5.2 Contextual UCB

The linear UCB, *LinUCB*, method was proposed by Li et. al. to calculate confidence bounds efficiently in the presence of contextual information when the reward function is linear (Li et al., 2010). Two variants of LinUCB are introduced in the paper: the first one, *disjoint* LinUCB, assumes that arms do not share information, i.e., do not share any parameters and the second one, *hybrid* LinUCB, assumes another set of parameters that is common for all arms in addition to arm-specific ones.

For disjoint LinUCB, the expected reward is calculated in the same way as it is presented in the formula 2.5. The coefficient vector θ_a^* is estimated with ridge regression according to the following formula:

$$\hat{\theta}_a = (D_a^T D_a + I_d)^{-1} D_a^T r_a \tag{2.6}$$

where D_a is the matrix for representing previously observed contexts and r_a is the reward associated with arm a. After the parameters for all arms are estimated, the arm that maximizes the following formula is chosen:

$$a_t = \underset{a \in \mathcal{A}_t}{\operatorname{argmax}} (x_{t,a}^T \hat{\theta}_a + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}) \tag{2.7}$$

where A_a is defined as $D_a^T D_a + I_d$ and α is a variable to control the degree of exploration. The algorithm allows to introduce a new set of arms in any iteration; thus, the set of arms at time t is denoted by a_t. The clear flow of algorithm is presented in the algorithm 6:

Algorithm 6 Disjoint LinUCB

Input: α
  for t=1...T **do**
    $x_{t,a} \leftarrow$ observe features of all arms $a \in \mathcal{A}_t$
    for all $a \in \mathcal{A}_t$ **do**
      if a is new **then**
        $A_a \leftarrow I_d$
        $b_a \leftarrow 0_{d\times 1}$
      end if
      $\hat{\theta_a} \leftarrow A_a^{-1} b_a$
      $p_{t,a} \leftarrow \hat{\theta}_a^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$
    end for
    choose arm $a_t = \text{argmax}_{a\in\mathcal{A}_t} p_{t,a}$ and observe reward r_{t,a_t}
    $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$
    $b_{a_t} \leftarrow b_{a_t} + r_{t,a_t} x_{t,a_t}$
  end for

For hybrid LinUCB algorithm, let β^* be the coefficient vector shared by all arms and $z_{t,a}$ be the k-dimensional feature vector of the shared context. Then, the expected reward is calculated by:

$$E(r_{t,a}|x_{t,a}) = z_{t,a}^T \beta^* + x_{t,a}^T \theta_a^* \tag{2.8}$$

Since we do not assume independence among arms in the hybrid LinUCB algorithm, we need a different formula to calculate the upper confidence bound than how it is presented for the disjoint LinUCB (see equation 2.7). The coefficient vectors are estimated by the following formulas:

$$\hat{\theta}_a = A_a^{-1}(b_a - B_a\hat{\beta}) \tag{2.9}$$

$$\hat{\beta} = A_0^{-1} b_0 \tag{2.10}$$

where the calculations of A_a, A_0, B_a, B_0, and b_a are presented clearly in algorithm 7. Then the arm that maximizes below formula is selected to observe the reward.

$$a_t = \underset{a\in A_t}{\text{argmax}}\, z_{t,a}^T \beta^* + x_{t,a}^T \theta_a^* + \alpha\sqrt{s_{t,a}} \tag{2.11}$$

where $s_{t,a}$ is $(z_{t,a}^T A_0^{-1} z_{t,a} - 2z_{t,a}^T A_0^{-1} B_a^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} B_a A_0^{-1} B_a^T A_a^{-1} x_{t,a})$.

Algorithm 7 Hybrid LinUCB

Input: α

$A_0 \leftarrow I_k$
$b_0 \leftarrow 0_k$
for t=1...T **do**
  $z_{t,a}, x_{t,a} \leftarrow$ observe features of all arms $a \in \mathcal{A}_t$
  $\hat{\beta} = A_0^{-1} b_0$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $A_a \leftarrow I_d$
      $B_a \leftarrow 0_{d \times k}$
      $b_a \leftarrow 0_{d \times 1}$
    end if
    $\hat{\theta_a} \leftarrow A_a^{-1}(b_a - B_a \hat{\beta})$
    $s_{t,a} \leftarrow z_{t,a}^T A_0^{-1} z_{t,a} - 2 z_{t,a}^T A_0^{-1} B_a^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} x_{t,a} + x_{t,a}^T A_a^{-1} B_a A_0^{-1} B_a^T A_a^{-1} x_{t,a}$
    $p_{t,a} \leftarrow z_{t,a}^T \hat{\beta} + x_{t,a}^T \hat{\theta_a} + \alpha \sqrt{s_{t,a}}$
  end for
  choose arm $a_t = \text{argmax}_{a \in \mathcal{A}_t} p_{t,a}$ and observe reward r_{t,a_t}
  $A_0 \leftarrow A_0 + B_{a_t}^T A_{a_t}^{-1} B_{a_t}$
  $b_0 \leftarrow b_0 + B_{a_t}^T A_{a_t}^{-1} b_{a_t}$
  $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$
  $B_{a_t} \leftarrow B_{a_t} + x_{t,a_t} z_{t,a_t}^T$
  $b_{a_t} \leftarrow b_{a_t} + r_{t,a_t} x_{t,a_t}$
  $A_0 \leftarrow A_0 + z_{t,a_t} z_{t,a_t}^T - B_{a_t}^T A_{a_t}^{-1} B_{a_t}$
  $b_0 \leftarrow b_0 + r_{t,a_t} z_{t,a_t} - B_{a_t}^T A_{a_t}^{-1} b_{a_t}$
end for

3
Data

3.1 SUMO

Simulation of Urban Mobility, abbreviated as SUMO (Lopez et al., 2018), is a tool for simulating traffic on a road network given some scenarios as input. The microscopic nature of SUMO is one of the important features which enables users to individually model the vehicle parameters and the routes for every vehicle. Codecá et al. (2017) have benefited from this microscopic nature and the open-source nature of the tool to create their own realistic scenario of a regular day in Luxembourg. The LuST (Luxembourg SUMO traffic) scenario network consists of 2365 nodes and 5959 edges where the edges belong to any of 3 different road types: highways, arterial roads, and residential roads marked with blue, red, and black respectively in Figure 3.1.

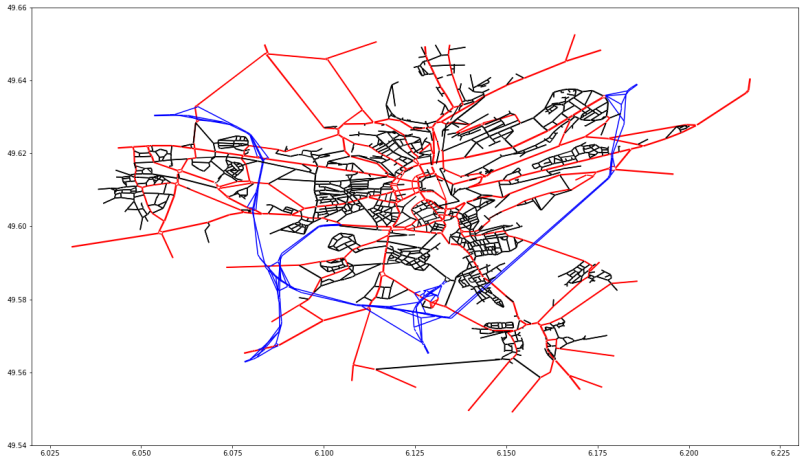


Figure 3.1: LuST scenario road network

To make this simulation realistic, two main rush-hour periods are defined by the authors of the scenario. The vehicle density reaches its peaks around 8.15 and again at 18.30 with approximately 5000 vehicles. There is also a smaller rush hour period in the middle of the day whose peak is centered around 13.00 with 3500 vehicles. Within the working hours, the lowest vehicle density occurs at 11.00 and 15.30 with approximately 1500 vehicles. In general, the lowest traffic demand is at night between 21.00-06.00. We can say that the traffic demand is modelled almost symmetrically around 13.00 as it can be observed in the following figure:

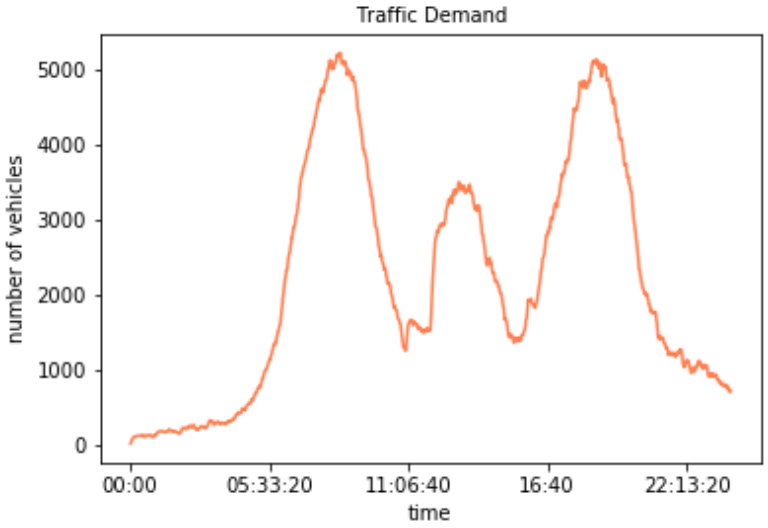


Figure 3.2: Traffic Demand

Although the original LuST scenario does not contain any electric vehicles, SUMO supports modelling electric vehicles since version 0.24.0 with energy consumption models implemented by Kurczveil et al. (2014). It is possible to modify some vehicle parameters that have an impact on energy consumption like *vehicle mass*, *front surface area*, *air drag coefficient*, *recuperation efficiency* etc. In addition to these vehicle-based parameters, it is also possible to set maximum speed, acceleration, and deceleration for each vehicle type to simulate different driving habits. We modified the vehicle parameters file to add 6 types of electric vehicles equivalent to standard passenger cars. The parameters we used are mainly the default values proposed by Kurczveil et al. (2014) with some modifications presented in the paper by Genikomsakis & Mitrentsis (2017). The common parameters for all 6 vehicle types are shown in Table 3.1.

Table 3.1: Electric vehicle parameters in SUMO

$maximumBatteryCapacity$	50000
$maximumPower$	80000
$vehicleMass$	2000
$frontSurfaceArea$	2.19
$airDragCoefficient$	0.29
$internalMomentOfInertia$	0.01
$radialDragCoefficient$	0.5
$rollDragCoefficient$	0.008
$constantPowerIntake$	300
$propulsionEfficiency$	0.9
$recuperationEfficiency$	0.95
$stoppingTreshold$	0.1

To introduce different driving patterns, we modified *maximum speed*, *acceleration*, and *deceleration* data differently for each type of vehicle. It is also important to note that all vehicles do not appear equally often in the simulation. The difference can be better observed in the following table:

Table 3.2: Driver-based parameters

	maximum speed	acceleration	deceleration	frequency
$Vehicle1$	70	2.6	4.5	40%
$Vehicle2$	90	4.5	4.5	20%
$Vehicle3$	50	2.0	4.0	20%
$Vehicle4$	70	2.7	4.5	10%
$Vehicle5$	30	2.4	4.5	5%
$Vehicle6$	30	2.3	4.5	5%

To generate realistic energy consumptions, we also need elevation data for each edge. To achieve that, we used USGS/NASA SRTM elevation data post-processed by Reuter et al. (2007) in *tif* format. We used a built-in function of SUMO for combining elevation data and road network: *netconvert*.

We have run the simulation for 24 simulation-hours to obtain the results of a regular day in Luxembourg. The output is an *xml* file of approximately 60GB. For each timestep, there is a list of vehicles currently active on some roads. The list also includes *energy consumed*, *speed*, *acceleration*, and *road id* fields that are necessary to form the reward and contextual data. An example of the output file is demonstrated in Table 3.3:

Table 3.3: Selected samples from simulation output, where timestep is 24.0

vehicle id	road id	energy consumed	speed	acceleration
0DEtoFR.0	-31622#1	-85.16	28.84	-1.14
0DEtoLU.0	-31622#0	205.11	33.06	2.03
randUni16882:1	-7046_5	287.36	7.08	2.07
randUni20481:1	—32964#8	24.12	13.48	0.54

Since a vehicle can stay on a specific road for more than one second, we need to aggregate those samples. We generated a total of 4 fields by aggregating all records of a specific vehicle on a specific road segment: *(energy consumption, speed, acceleration, and deceleration)*. To generate the *energy consumption* field, we summed all samples. This field can either be positive or negative related to whether the vehicle consumed or regenerated energy on that road segment. To obtain the value of the *speed* field, we took the average of all records. *Acceleration* and *deceleration* fields were obtained by summing the positive and negative acceleration data separately where the sum of negative acceleration corresponds to the deceleration field in the output file. The reason for obtaining both acceleration and deceleration data is that we wanted to observe the effect of both fields. Another reason is that negative and positive acceleration data can cancel each other which might result in misleading

the model. After the processing step, we get a *json* file where the road ids are the field names, which contains a list of *(energy consumption, speed, acceleration, and deceleration)* tuples. An example data of some roads is given in the Figure 3.3.


```
"-30528#50": [[23.4, 13.512, 0.4575, -0.45714]],
"--31340#1": [[7.03, 10.695, 0.765, -0.39]],
"-31712#1": [[28.01, 13.8375, 0.62, -0.18667], [-19.96, 15.9, 0.26, -0.83667], [24.88, 11.406, 0.39, -0.53]],
"-32964#7": [[50.27, 12.38286, 1.18, -1.34]],
"-31622#8": [[74.24, 38.97182, 1.202, -1.10833], [-126.24, 34.22538, 1.102, -2.26875], [50.62, 38.38727, 0.26, -0.24429],
"-30612": [[33.7, 18.07733, 0.775, -1.07571], [27.65, 13.68, 0.495, -0.71889]],
"-30528#10": [[11.93, 13.29875, 0.54, -0.485]],
```


Figure 3.3: A small part of *json* output file containing data of some roads after the first 600 seconds, (key, value): (road id, list of tuples)

Since 60GB of data does not fit into our computer's memory, aggregating approximately 60GB of data is not a straightforward process. We continuously iterated over the *xml* file and created the output file, but this process is also very time consuming. Therefore, we processed only half of the data, i.e. until approximately 12.00 in simulation time. No significant information loss is expected due to this decision because the traffic demand is almost symmetrical around 13.00.

After these processing steps, we found out that some road segments have no samples. This may be due to the sampling rate, which is one sample per second. We believe that travelling with high speed on very short road segments caused this issue. The road segments with at least one sample are demonstrated in Figure 3.4

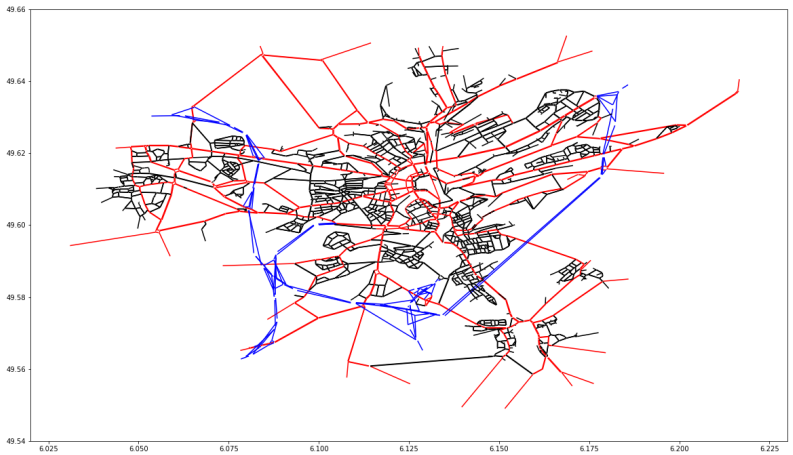


Figure 3.4: The road segments of Luxembourg map with at least one sample after the preprocessing steps

4
Methods

4.1 Contextual Combinatorial Semi-Bandit

In order to address the energy efficient navigation problem, a contextual combinatorial semi-bandit setting is adopted. When the map of a city is represented by a graph where junctions correspond to nodes and road segments correspond to edges, pulling one arm in a standard multi-armed bandit setting (selecting one road segment) is not enough to form a whole path. For this reason, a combinatorial multi-armed bandit setting is employed in the application where each path corresponds to a super arm with road segments as base arms. The road segments should receive individual rewards since energy consumption of a road segment can be assumed to be independent of the chosen path. To embed this structure, it is appropriate to adopt a semi-bandit setting as well.

Lastly, a contextual bandit setting is chosen, even though it may not be necessary, it may achieve better predictions of energy consumptions. Average speed, acceleration, and deceleration data are included as the contextual information. Since keeping contextual information constant at every iteration would be equivalent to have a context-free bandit, different contexts are needed for every iteration. To achieve this diversity at every iteration, the samples are selected from the data presented in Figure 3.2 for each edge in the network graph. The first field of sampled data (energy consumption) is used in the reward calculation and the rest is for constructing the context vector $x_{t,a}$. Two sets of context vectors are used to observe the effect of utilizing acceleration and deceleration data separately. The context vectors are formulized as following:

$$\begin{bmatrix} (speed)^2 \\ acceleration + deceleration \end{bmatrix} \qquad \begin{bmatrix} (speed)^2 \\ acceleration \\ deceleration \end{bmatrix} \tag{4.1}$$

For the evaluation of algorithms, the true parameters of all road segments should be known to be able to find the optimal path and to calculate the regret. Shortest path algorithms like Dijkstra's algorithm (Dijkstra, 1958) can reveal the optimal path according to these true parameters treated as edge weights. Then, the optimal reward is calculated by summing the true mean rewards of the base arms in the optimal super arm found by Dijkstra's algorithm. Likewise, summing the true mean rewards of the base arms in the played super arm at each iteration would reveal the mean reward of playing that super arm. Subtracting the reward of the played super

arm from the reward of the optimal path yields the instant regret. One approach to approximate the true parameters can be to calculate the mean energy consumption from history. Since it is averaged over the whole dataset related to a specific edge, this approach can be seen as an *offline* evaluation technique. Another evaluation method that uses expected reward conditioned on context as true parameters can be adopted. To approximate the expected reward conditioned on context, it is fairly reasonable to use the actual historic energy consumption associated with the sampled context. The major difference from the previous approach is that at each iteration, the optimal path is recalculated with new edge weights. The reason for choosing such an evaluation method is the fact that the optimal path may change when different contexts are sampled. Additionally, this method does not require a static dataset; thus, it may be seen as an *online* evaluation technique.

The base code that is used was initially created by Russo et al. (2017) and later modified by Åkerblom et al. (2020) to support more functionalities. This code is modified to use the road network graph of Luxembourg and contextual information obtained from SUMO. Additionally, disjoint LinUCB and hybrid LinUCB algorithms have also been implemented.

4.1.1 Thompson Sampling for Contextual Combinatorial Semi-Bandit

A Gaussian prior distribution and a Gaussian likelihood function are chosen since the Gaussian family is self-conjugate, meaning that the posterior distribution will also be Gaussian in the Bayesian setting. Thus, it is fairly easy to compute the parameters of the posterior distribution. The cumulative regret can depend on the selection of prior parameters in Thompson sampling; thus, they should be selected carefully. The prior parameters can be selected proportionally to the length of individual edges or according to some energy calculation formula. It is also possible to choose a completely uninformative prior. The reason for choosing an uninformative prior is to be able to fairly compare the Thompson sampling algorithm with UCB-based algorithms. Additionally, it can be interesting to observe the effect of choosing different priors on cumulative regret. When d fields are selected as contextual information, the prior mean $\mu_{0,e}$ is in the form of a $d \times 1$ column matrix and the prior covariance $\Sigma_{0,e}$ is a $d \times d$ matrix. The first set of priors includes the same constant mean and covariance matrix for every edge. The prior mean is defined as a $d \times 1$ column-vector of zeros and the covariance matrix as $d \times d$ identity matrix. The second set of priors is chosen by taking edge lengths into account. Let $\phi_e = [\textit{length of } e] \cdot 0.01$ and $\sigma_e^2 = \phi_e \cdot 5$ for edge e. Then, the prior mean is a $d \times 1$ column-vector of ones multiplied with ϕ_e and the prior covariance matrix is $\sigma_e^2 \cdot I$ for edge e where I is a $d \times d$ identity matrix.

The agent samples parameters $\hat{\theta}_e$ from the posterior distribution for each edge e and uses the sampled contextual information $x_{t,e}$ to calculate the expected reward. The calculation of expected reward is given in the equation 2.4. To be able to select the best super arm, Dijkstra's algorithm is used to find the shortest path between

a source node and a destination node. The calculated expected rewards are given as edge weights to the algorithm. The base arms in the super arm proposed by Dijkstra's algorithm are pulled to observe the reward. The reward is given for each individual base arm as the negative sum of the energy consumption field of the sampled data. Then, *only* the distributions of base arms in the played super arm are updated according to the following formula where $\mu_{t,e}$ represents the posterior mean vector and $\Sigma_{t,e}$ represents the posterior covariance matrix:

$$\begin{aligned} \Sigma_{t,e} &\leftarrow (\Sigma_{t-1,e}^{-1} + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T x_{t,e})^{-1} \\ \mu_{t,e} &\leftarrow \Sigma_{t,e}(\Sigma_{t-1,e}^{-1}\mu_{t-1,e} + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T r_{t,e}) \end{aligned} \tag{4.2}$$

where $\tilde{\sigma_e}$ is the standard deviation of the samples of edge e. It is assumed that the standard deviation of the energy consumption of all edges is known in advance to ease the calculations. The overall algorithm is presented in algorithm 8:

Algorithm 8 Contextual Combinatorial Thompson Sampling

Input: μ_0, Σ_0, $G = (V, E)$
  for t=1...T **do**
    for all $e \in E$ **do**
      $\hat{\theta}_e \leftarrow$ sample parameters from posterior distribution $\mathcal{N}(\mu_{t-1,e}, \Sigma_{t-1,e})$
      $x_{t,e} \leftarrow$ observe the context of edge e
      Update edge weight of e in G with $x_{t,e}^T \hat{\theta}_e$
    end for
    $S_t \leftarrow$ run Dijkstra's algorithm on G to find the shortest path
    for all $e \in S_t$ **do**
      $r_{t,e} \leftarrow$ observe the reward of played edge e
      $\Sigma_{t,e} \leftarrow (\Sigma_{t-1,e}^{-1} + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T x_{t,e})^{-1}$
      $\mu_{t,e} \leftarrow \Sigma_{t,e}(\Sigma_{t-1,e}^{-1}\mu_{t-1,e} + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T r_{t,e})$
    end for
  end for

4.1.2 Disjoint LinUCB for Contextual Combinatorial Semi-Bandit

The pseudocode presented in algorithm 6 is used for implementing the disjoint LinUCB algorithm with minor changes both in the calculation of upper confidence bound $p_{t,e}$ and updating the parameters A and b. In the general algorithm, it is reasonable to add the confidence width term to the mean, since the optimistic behavior would be to think that it is possible to receive a higher reward from playing that arm. However, in this application, this approach would result in a pessimistic behavior, i.e. thinking that the edge may result in a higher cost (energy consumption). Therefore, the confidence width should be subtracted from the mean to adopt an optimistic behavior. For clarity, the new calculation of upper confidence bound is presented in equation 4.3.

$$p_{t,e} \leftarrow \hat{\theta}_e^T x_{t,e} - \alpha\sqrt{x_{t,e}^T A_e^{-1} x_{t,e}} \tag{4.3}$$

For updating the parameters, A and b, standard deviation of edges $\tilde{\sigma_e}$ is also added to the calculation like in the Thompson Sampling algorithm. Equation 4.4 shows the new update algorithms.

$$\begin{aligned} A_e &\leftarrow A_e + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T x_{t,e} \\ b_e &\leftarrow b_e + \frac{1}{\tilde{\sigma_e}^2} x_{t,e}^T r_{t,e} \end{aligned} \tag{4.4}$$

Since UCB-based algorithms do not have any priors to set, the only input is α which controls the degree of exploration. Setting α to 0 is expected to result in less exploration. In fact, it would be equivalent to have a greedy algorithm where the prior mean is a zero vector and the prior covariance is an identity matrix. The reason is that $A^{-1}b$ can be interpreted as prior mean and the inverse of A, A^{-1}, as prior covariance matrix in the Bayesian setting. These facts are also utilized to perform sanity checks.

However, the difference is that at least in the standardized LinUCB, A and b are initialized with constants, in contrast to the Thompson Sampling where the prior parameters are treated as hyperparameters and can be adjusted. Thus, some experiments are run with changing the only adjustable parameter of LinUCB, α, to assess how the degree of exploration contributes to learning process. The chosen α values are: 2, 1, 0.5, 0.1. The overall algorithm is presented in algorithm 9.

Algorithm 9 Combinatorial Disjoint LinUCB

Input: α, $G = (V, E)$
  $d \leftarrow$ dimension of context vectors
  for all $e \in E$ **do**
    $A_e \leftarrow I_{d \times d}$
    $b_e \leftarrow 0_{d \times 1}$
  end for
  for t=1...T **do**
    for all $e \in E$ **do**
      $x_{t,e} \leftarrow$ observe the context of edge e
      $\hat{\theta}_e \leftarrow A_e^{-1} b_e$
      $p_{t,e} \leftarrow \hat{\theta}_e^T x_{t,e} - \alpha\sqrt{x_{t,e}^T A_e^{-1} x_{t,e}}$
      Update edge weight of e in G with $p_{t,e}$
    end for
    $S_t \leftarrow$ run Dijkstra's algorithm on G to find the shortest path
    for all $e \in S_t$ **do**
      $r_{t,e} \leftarrow$ observe the reward of played edge e
      $A_e \leftarrow A_e + \frac{1}{\tilde{\sigma_e}^2} x_{t,e} x_{t,e}^T$
      $b_e \leftarrow b_e + \frac{1}{\tilde{\sigma_e}^2} r_{t,e} x_{t,e}$
    end for
  end for

4.1.3 Hybrid LinUCB for Contextual Combinatorial Semi-Bandit

Similar to disjoint LinUCB, the implementation of the hybrid LinUCB also has small differences from the original implementation presented in algorithm 7. The modification of subtracting the confidence width from the mean is also valid in this case. Likewise, the sample standard deviation is used in updating the parameters. The equations will not be presented individually to avoid repetition, but can be observed in algorithm 10. The same set of α introduced for disjoint LinUCB is also used for hybrid LinUCB experiments.

In addition to the edge-specific contexts used in disjoint LinUCB and Thompson Sampling, hybrid LinUCB utilizes population data as a common feature represented as z. The idea is that the edges with similar population density are expected to have similar energy consumptions, since electric vehicles are considered to be more efficient in urban driving compared to e.g. highways (Wu et al., 2015).

Algorithm 10 Combinatorial Hybrid LinUCB

Input: α, $G = (V, E)$
  $d \leftarrow$ dimension of edge-specific context vectors
  $k \leftarrow$ dimension of shared context vectors
  $A_0 \leftarrow I_{k\times k}$
  $b_0 \leftarrow 0_{k\times 1}$
  for all $e \in E$ **do**
    $A_a \leftarrow I_{d\times d}$
    $B_a \leftarrow 0_{d\times k}$
    $b_a \leftarrow 0_{d\times 1}$
  end for
  for t=1...T **do**
    $\hat{\beta} = A_0^{-1} b_0$
    for all $e \in E$ **do**
      $z_{t,e}, x_{t,e} \leftarrow$ observe features of edge e
      $\hat{\theta_e} \leftarrow A_e^{-1}(b_e - B_e\hat{\beta})$
      $s_{t,e} \leftarrow z_{t,e}^T A_0^{-1} z_{t,e} - 2z_{t,e}^T A_0^{-1} B_e^T A_e^{-1} x_{t,e} + x_{t,e}^T A_e^{-1} x_{t,e} + x_{t,e}^T A_e^{-1} B_e A_0^{-1} B_e^T A_e^{-1} x_{t,e}$

      $p_{t,e} \leftarrow z_{t,e}^T \hat{\beta} + x_{t,e}^T \hat{\theta_a} - \alpha\sqrt{s_{t,e}}$
      Update edge weight of e in G with $p_{t,e}$
    end for
    $S_t \leftarrow$ run Dijkstra's algorithm on G to find the shortest path
    for all $e \in S_t$ **do**
      $r_{t,e} \leftarrow$ observe the reward of played edge e
      $A_0 \leftarrow A_0 + B_e^T A_e^{-1} B_e$
      $b_0 \leftarrow b_0 + B_e^T A_e^{-1} b_e$
      $A_e \leftarrow A_e + \frac{1}{\tilde{\sigma_e}^2} x_{t,e} x_{t,e}^T$
      $B_e \leftarrow B_e + \frac{1}{\tilde{\sigma_e}^2} x_{t,e} z_{t,e}^T$
      $b_e \leftarrow b_e + \frac{1}{\tilde{\sigma_e}^2} r_{t,e} x_{t,e}$
      $A_0 \leftarrow A_0 + \frac{1}{\tilde{\sigma_e}^2} z_{t,e} z_{t,e}^T - B_e^T A_e^{-1} B_e$
      $b_0 \leftarrow b_0 + \frac{1}{\tilde{\sigma_e}^2} r_{t,e} z_{t,e} - B_e^T A_e^{-1} b_e$
    end for
  end for

4.1.4 Greedy Algorithm for Contextual Combinatorial Semi-Bandit

The greedy algorithm is expected to explore less when compared to other algorithms; thus, is expected to result in higher regret. Because of this characteristic, the greedy algorithm is implemented to constitute a baseline for comparing different algorithms. Like the Thompson Sampling algorithm, the priors are set at the beginning. The same set of priors introduced for Thompson Sampling is also used for the greedy algorithm. The algorithm is implemented based on the ϵ-greedy algorithm presented in algorithm 1 with ϵ set to 0. The overall algorithm is demonstrated in algorithm 11.

Algorithm 11 Contextual Combinatorial Greedy Algorithm

Input: μ_0, Σ_0, $G = (V, E)$
  for t=1...T **do**
    for all $e \in E$ **do**
      $\hat{\theta}_e \leftarrow$ estimate the mean of posterior distribution $\mathcal{N}(\mu_{t-1,e}, \Sigma_{t-1,e})$
      $x_{t,e} \leftarrow$ observe the context of edge e
      Update edge weight of e in G with $x_{t,e}^T \hat{\theta}_e$
    end for
    $S_t \leftarrow$ run Dijkstra's algorithm on G to find the shortest path
    for all $e \in S_t$ **do**
      $r_{t,e} \leftarrow$ observe the reward of played edge e
      $\Sigma_{t,e} \leftarrow (\Sigma_{t-1,e}^{-1} + \frac{1}{\bar{\sigma_e}^2} x_{t,e}^T x_{t,e})^{-1}$
      $\mu_{t,e} \leftarrow \Sigma_{t,e}(\Sigma_{t-1,e}^{-1} \mu_{t-1,e} + \frac{1}{\bar{\sigma_e}^2} x_{t,e}^T r_{t,e})$
    end for
  end for

4.2 Context-free Combinatorial Semi-Bandits

These bandits will serve as baselines for assessing the degree of benefit from introducing contexts. Replacing context vectors with a constant 1 (1×1 matrix) in contextual algorithms, all contextual algorithms listed in the previous section can be converted to context-free versions except for hybrid LinUCB. Hybrid LinUCB is a contextual algorithm inherently and setting both arm-specific and shared contexts to 1 causes more harm than good since it raises the issue of multicollinearity. Likewise, removing the shared context would result in the same algorithm as disjoint LinUCB. For this reason, no experiment on context-free hybrid LinUCB is conducted.

The parameters that can tuned are prior mean and variance for Thompson Sampling and α for disjoint LinUCB. The one-dimensional versions of previously introduced sets of parameters for the contextual algorithms will also be used in the context-free setting in order to compare the algorithms fairly.

5

Results and Discussion

In this chapter, the results of different scenarios are presented with comparisons. For each agent in a scenario, the regret is calculated by averaging over 3 experiments to decrease the effect of randomness. Figure 5.1 shows the start and end points of all scenarios, unless otherwise stated.

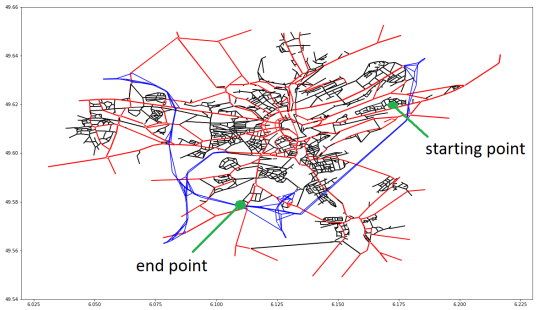


Figure 5.1: Start and end points of main scenario

Likewise, the default values/methods used in the experiments are as follows unless otherwise stated:

- semi-informative prior mean and covariance/variance
- α=0.1
- context vector: $\begin{bmatrix}(speed)^2 & acceleration & deceleration\end{bmatrix}^T$
- regret calculation: historic mean

5.1 Prior Selection for Thompson Sampling and Greedy Algorithm

In contrast to the context-free setting, prior means do not represent mean energy consumption here. Rather, they represent the coefficients to be multiplied with the sampled context. Therefore, coefficients (prior means) are expected to be much smaller than the energy consumption in contextual setting. For this reason, the priors are chosen to be small numbers initially and they can be observed in Table 5.1.

Table 5.1: Two different set of priors

semi-informative priors	uninformative priors
$\mu_{0,e} = \begin{bmatrix} length(e) \cdot 0.01 & length(e) \cdot 0.01 & length(e) \cdot 0.01 \end{bmatrix}^T$	$\mu_{0,e} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
$\Sigma_{0,e} = \begin{bmatrix} length(e) \cdot 0.05 & 0 & 0 \\ 0 & length(e) \cdot 0.05 & 0 \\ 0 & 0 & length(e) \cdot 0.05 \end{bmatrix}$	$\Sigma_{0,e} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The first set of priors, semi-informative priors, is unique to each edge with respect to its length. 2882 road segments out of 5959 are shorter than 100 meters and only 87 road segments have length of more than 1000 meters. Thus, for most of the road segments, dividing the edge length to 100 would result in small numbers, i.e. they start with relatively small prior mean while taking advantage of edge lengths. The second set of priors, uninformative priors, are chosen to ease the comparison between UCB-based algorithms with Thompson Sampling. Something which is important to be careful about is to make sure that Thompson Sampling explores enough with these relatively small prior variances.
The Figures 5.2, 5.3, 5.4, and 5.5 are the resulting plots where Thompson Sampling and greedy agents are initiated with semi-informative priors.

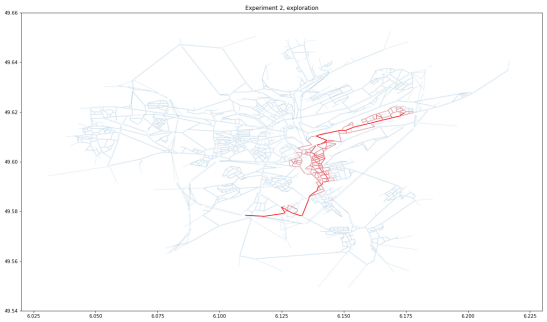


Figure 5.2: Exploration of Thompson Sampling with semi-informative priors

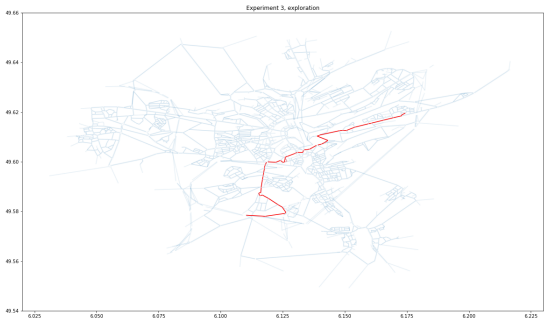


Figure 5.3: Exploration of the greedy algorithm with semi-informative priors

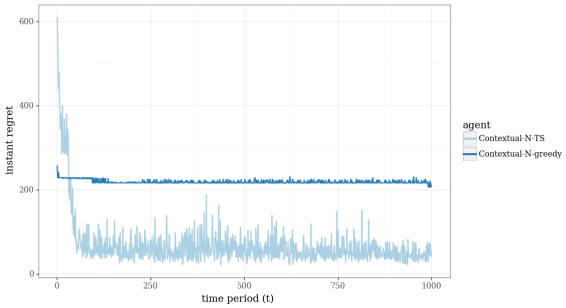


Figure 5.4: Instant regret plot of two agents with semi-informative priors over 1000 iterations

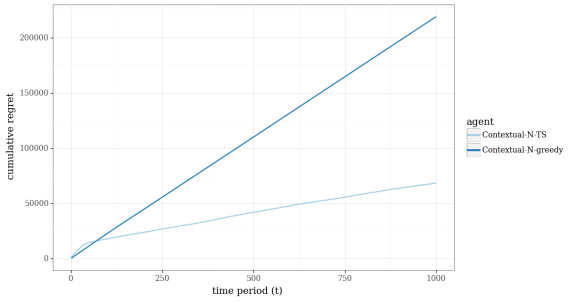


Figure 5.5: Cumulative regret plot of two agents with semi-informative priors over 1000 iterations

In Figures 5.2 and 5.3, it can be observed that Thompson Sampling explores more than the greedy algorithm. This exploration enables Thompson Sampling to learn better predictions, which can be seen in the instant regret plot. While the instant regret of Thompson Sampling decreases over time, the instant regret of the greedy algorithm stays the same. The cumulative regret plot (Figure 5.5) clearly indicates the difference between the two agents.

The Figures 5.6, 5.7, 5.8 and 5.9 are the resulting plots where Thompson Sampling and greedy agents are initiated with uninformative priors.

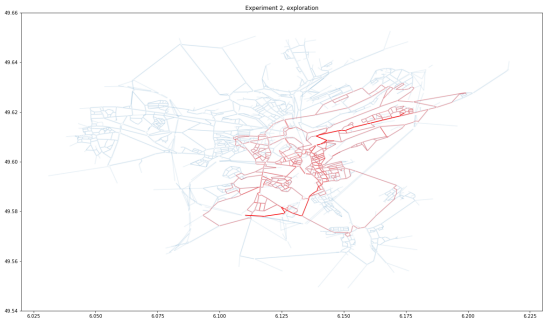


Figure 5.6: Exploration of Thompson Sampling with uninformative priors

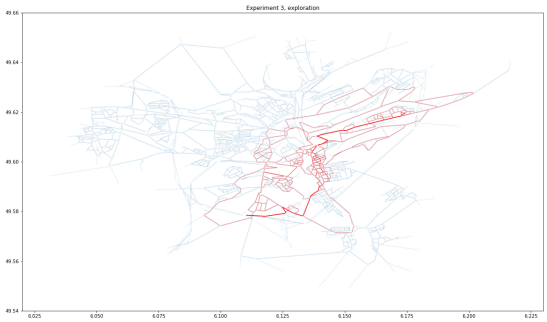


Figure 5.7: Exploration of the greedy algorithm with uninformative priors

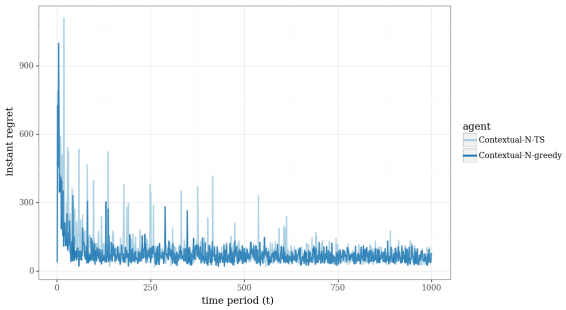


Figure 5.8: Instant regret plot of two agents with uninformative priors over 1000 iterations

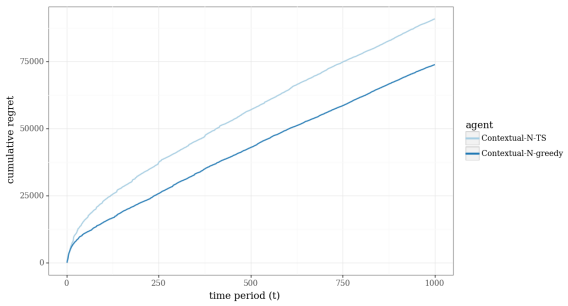


Figure 5.9: Cumulative regret plot of two agents with uninformative priors over 1000 iterations

In this case, both Thompson Sampling and the greedy algorithm explore more than in the previous case. For the greedy algorithm, an improvement may be noticed which shows the need for exploration to some degree. However, that does not necessarily lead to a better cumulative regret. For Thompson Sampling, a worse cumulative regret is observed when it is initiated with uninformative priors.

The above cumulative regret plots are combined into one plot in Figure 5.10 to visually ease the comparison. The best agent is seemingly Thompson Sampling with semi-informative priors. Even though the greedy agent with semi-informative priors results in the worst regret by far, it behaves no differently than a greedy agent is expected to behave. Thus, we choose not to take this into consideration while

deciding on the priors. For these reasons, the majority of the rest of the work is conducted with semi-informative priors.

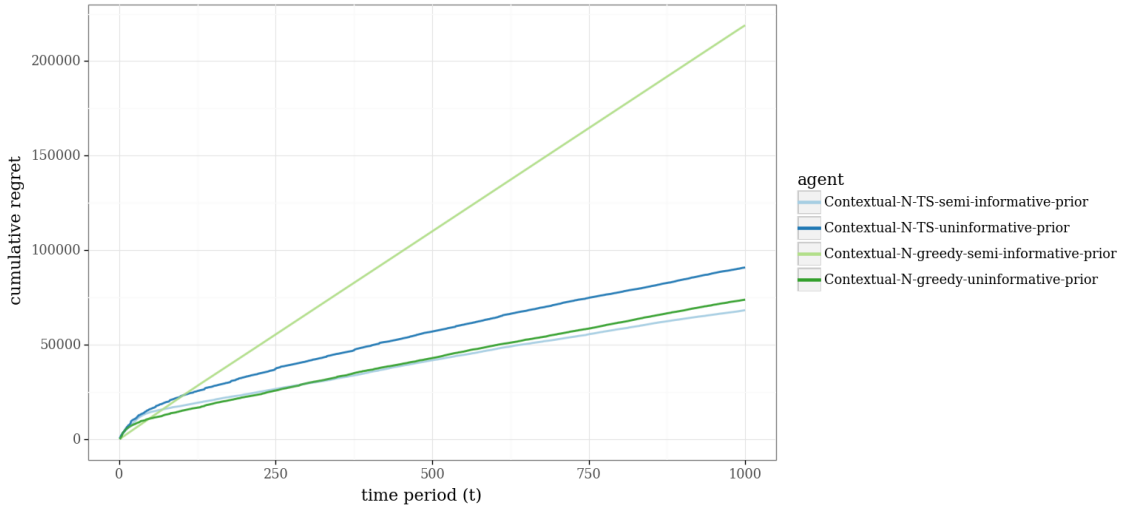


Figure 5.10: The cumulative regrets of Thompson Sampling and greedy agents initiated with both semi-informative and uninformative priors

As a bonus, to experimentally show that larger prior mean leads to worse cumulative regret, an experiment is conducted where semi-informative prior mean is decupled, but the prior covariance stays the same. To be clear, the formulation of *bonus* priors are as following:

$$\mu_{0,e} = \begin{bmatrix} length(e) \cdot 0.1 & length(e) \cdot 0.1 & length(e) \cdot 0.1 \end{bmatrix}^T$$

$$\Sigma_{0,e} = \begin{bmatrix} length(e) \cdot 0.05 & 0 & 0 \\ 0 & length(e) \cdot 0.05 & 0 \\ 0 & 0 & length(e) \cdot 0.05 \end{bmatrix}$$

The Figures 5.11, 5.12, 5.13, and 5.14 are the resulting plots where Thompson Sampling and greedy agents are initiated with these *bonus* priors.

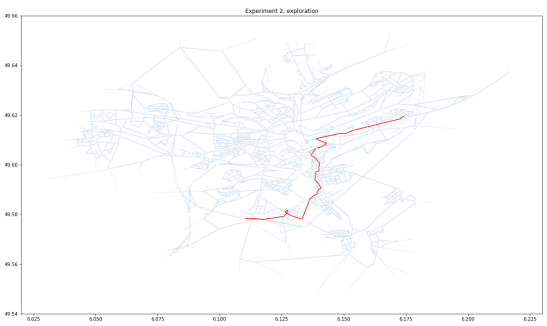


Figure 5.11: Exploration of Thompson Sampling with bonus priors

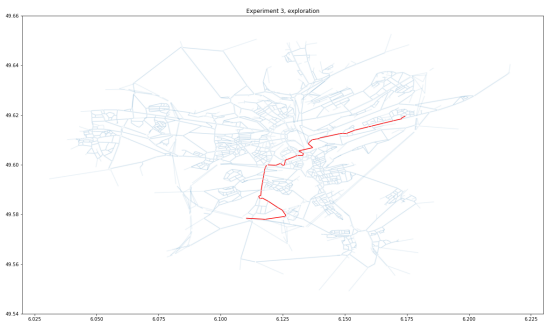


Figure 5.12: Exploration of Greedy Algorithm with bonus priors

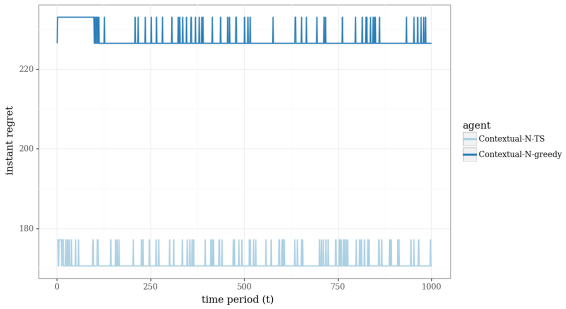


Figure 5.13: Instant regret plot of two agents with bonus priors over 1000 iterations

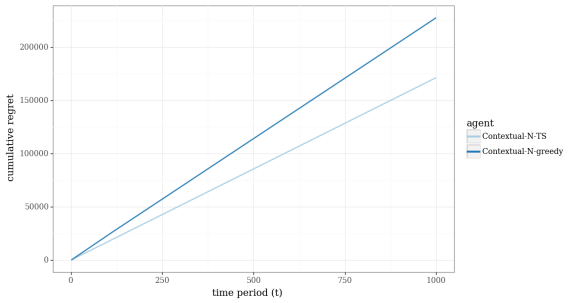


Figure 5.14: Cumulative regret plot of two agents with bonus priors over 1000 iterations

Looking at the exploration plots of semi-informative case and this case (Figures 5.2, 5.3, 5.11, and 5.12), it can be claimed that the larger the prior mean, the less it explores when the prior covariance is fixed. When an agent chooses a path at the first iteration and receives the reward, it updates so that the selected path is believed to result in substantially better regret at the next iteration. This behavior causes the agents to be pessimistic.

5.2 Hyperparameter Tuning for Disjoint and Hybrid LinUCB

The experiments in this section are conducted using 4 different α values: 2, 1, 0.5, and 0.1

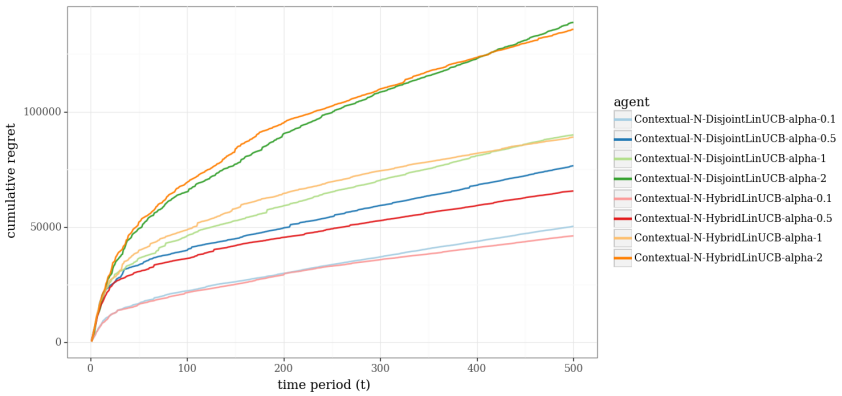


Figure 5.15: Cumulative regret of disjoint and hybrid LinUCB agents with several different α values over 500 iterations

From Figure 5.15, it can be observed that the smaller α values result in lower cumulative regret in both agents. However, before reaching a conclusion from this plot, we need to be sure that the agents with $\alpha = 0.1$ also explore sufficiently. Table 5.2 shows the amount of exploration by each agent.

Table 5.2: Explored edges with different α values where agents are disjoint and hybrid LinUCB

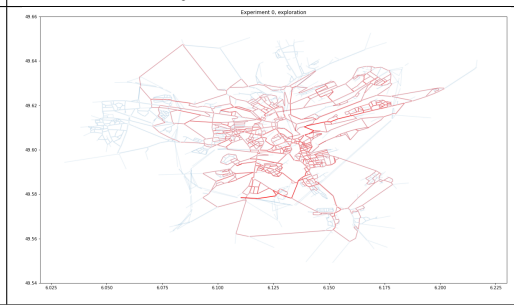
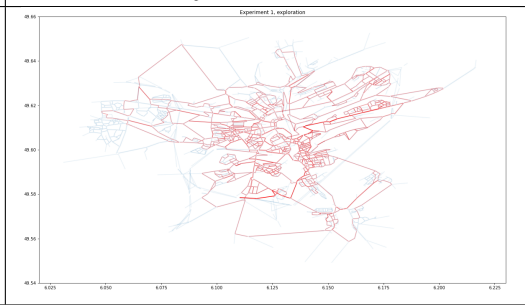
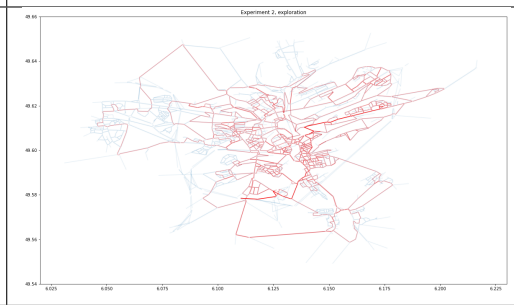
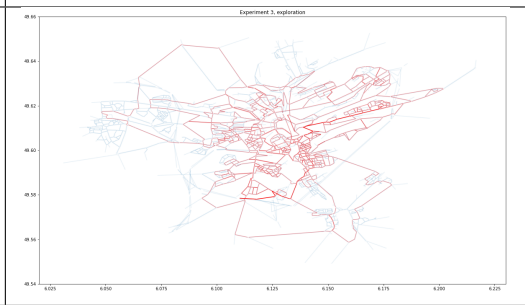
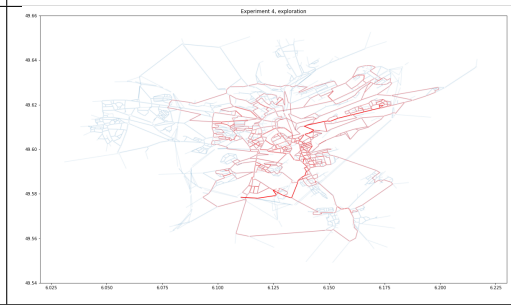
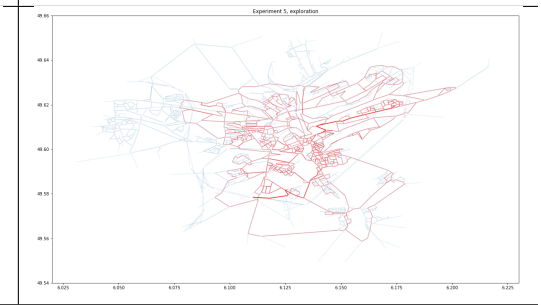
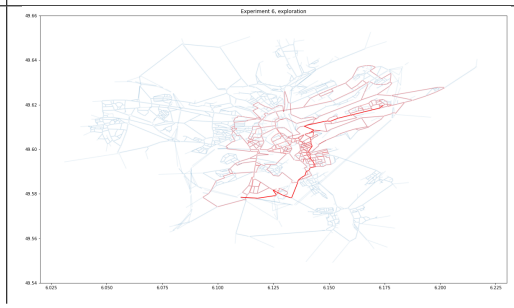
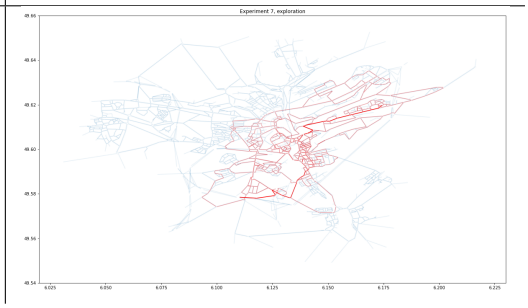
	Hybrid LinUCB	Disjoint LinUCB
$\alpha = 2$		
$\alpha = 1$		
$\alpha = 0.5$		
$\alpha = 0.1$		

Table 5.2 shows the effect of changing α on controlling the degree of exploration for both agents. While the degree of exploration does not change much between two agents for the same α value, decreasing α to 0.1 reduces the unnecessary exploration for both agents. Nevertheless, it should not be inferred that decreasing α further will always yield a better result. In the following sections, it is shown that the greedy approach, where $\alpha = 0$, has the worst regret.

5.3 Choice of Context

In order to determine whether 2×1 or 3×1 context vector yields a lower cumulative regret, the experiments with 1000 iterations are run and the results are shown in Figure 5.16.

- 2 contexts: squared speed, acceleration + deceleration
- 3 contexts: squared speed, acceleration, deceleration

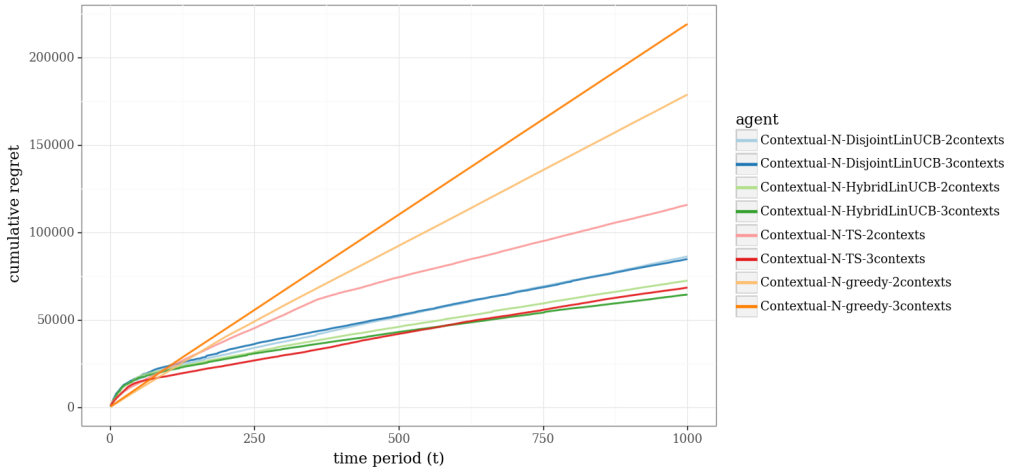


Figure 5.16: Cumulative regret of greedy, Thompson Sampling, disjoint LinUCB and hybrid LinUCB agents with two different sets of contextual information

Except for greedy, all agents yield lower cumulative regret when the context involves squared speed, acceleration, and deceleration data. This supports the hypothesis made in the Data section: Disregarding the positive and negative acceleration and adding them up to reach an average acceleration would mislead the model. Here, at least some information from the driving cycle is retained.

5.4 Contextual vs Context-free Algorithms

The default parameters mentioned at the beginning of Results and Discussion chapter are used for contextual algorithms, whereas the context-free priors are the one dimensional versions of contextual priors. The cumulative regrets of contextual and context-free algorithms are merged into one plot in Figure 5.17. The instant regret can be observed separately in Figures 5.18 and 5.19.

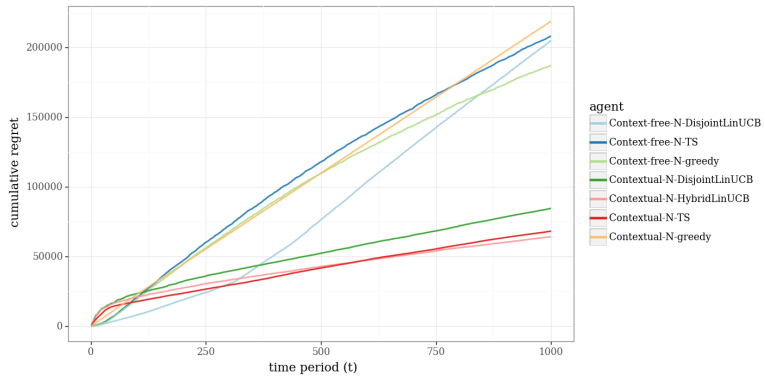


Figure 5.17: Cumulative regret of contextual and context-free algorithms

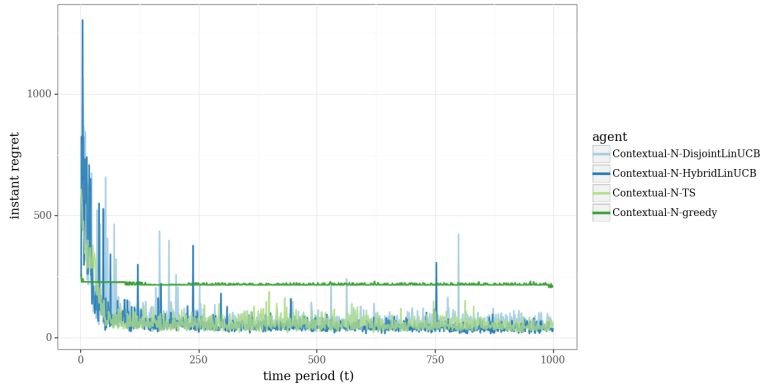


Figure 5.18: Instant regret of contextual algorithms

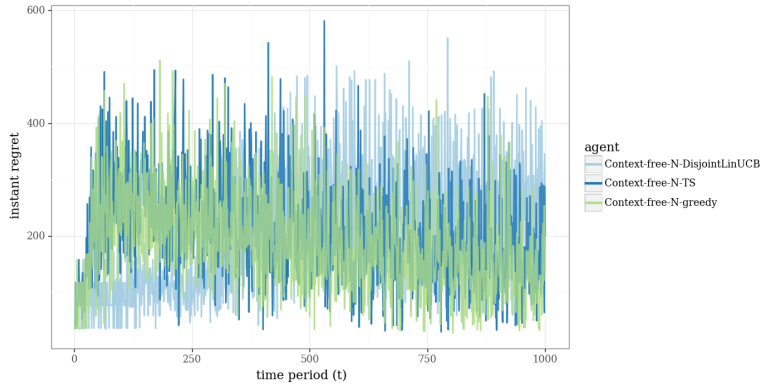


Figure 5.19: Instant regret of context-free algorithms

From the cumulative regret plot, it is clear that context-free algorithms perform worse than contextual ones except for the context-free greedy algorithm. This can also be observed in the instant regret plots. The instant regrets of contextual agents tend to decrease over time, indicating that the agents are learning the parameters of the energy consumption distributions, whereas the instant regrets of context-free agents do not decrease. Thus, it can be concluded that contextual algorithms are superior to the context-free ones here.

5.5 Evaluation of Different Regret Calculations

As mentioned in the Methods chapter, two different regret calculations are used to investigate a suitable way to evaluate contextual algorithms. First approach uses sample mean consumption as edge weight, referred to here as *sample mean regret*. The second approach uses the energy consumption of the sampled context as edge weight, referred to as *conditional regret* in this section. The instant and cumulative regrets of all 4 agents with different regret calculation methods are demonstrated in Figure 5.20 and 5.21.

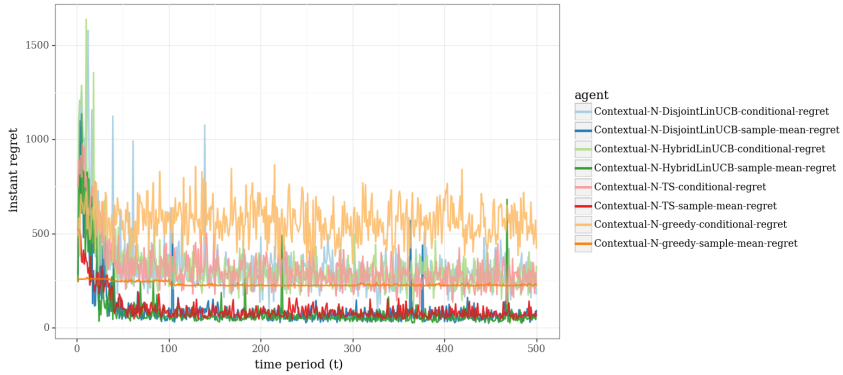


Figure 5.20: Instant regret of all 4 agents with different regret calculation methods

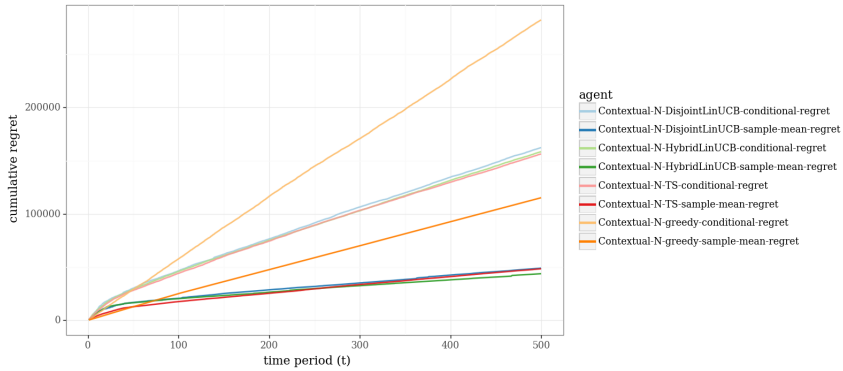


Figure 5.21: Cumulative regret of all 4 agents with different regret calculation methods

When the regret calculation methods are compared for each agent individually, it is obvious that the conditional regret method results in worse cumulative regret. Although it is true that mean energy consumption should change with different contexts, some irregularities in the data are observed such that similar contexts do not always yield similar energy consumptions. Thus, conditional regret calculation suffers from outliers in this case.

5.6 Different Paths

Experiments on other scenarios are also conducted where starting and end points are changed. The purpose here is to learn whether a certain agent performs better under certain circumstances. The first path is presented in Figure 5.1, referred to here as *east-west* path. The second path starts from north of the city and ends in the southern part (Figure 5.22), referred to as *north-south* path. The last one is selected to be a short path where both source and destination points are located in the city center (Figure 5.23), referred to as *short* path.

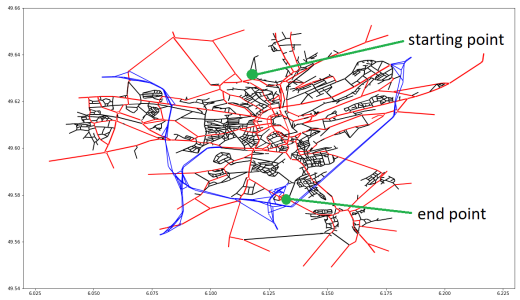


Figure 5.22: Starting and end points of north-south path

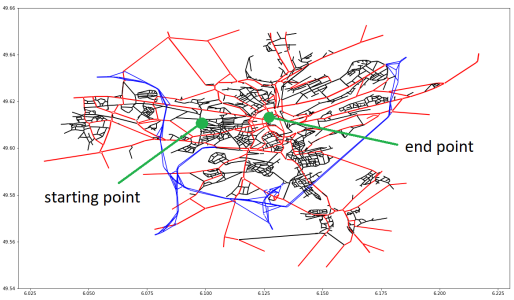


Figure 5.23: Starting and end points of short path

Since the length of paths are different from each other, it would not be wise to compare the cumulative regrets of these different paths. Nevertheless, it can be checked that whether one agent performs better compared to the other agents regardless of the source and destination points. Figure 5.24, 5.26, and 5.28 are the cumulative regret plots of different paths, whereas Figures 5.25, 5.27, and 5.29 are the instant regret plots.

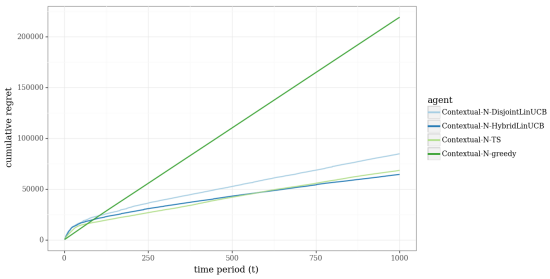


Figure 5.24: Cumulative regret of all 4 agents on east-west path

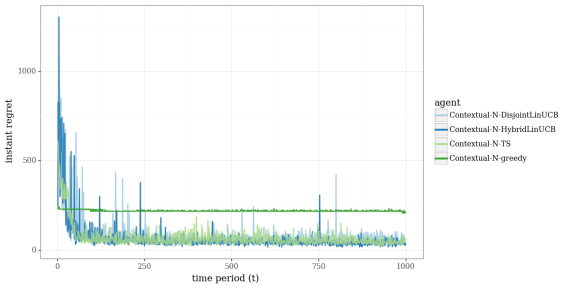


Figure 5.25: Instant regret of all 4 agents on east-west path

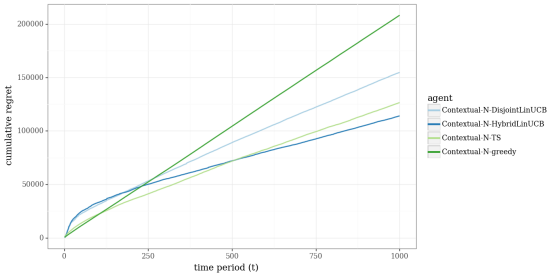


Figure 5.26: Cumulative regret of all 4 agents on north-south path

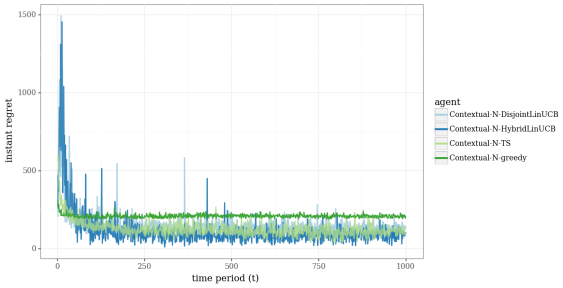


Figure 5.27: Instant regret of all 4 agents on north-south path

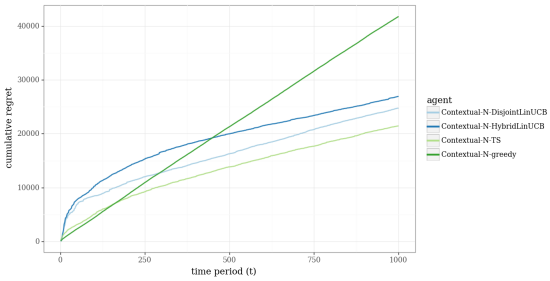


Figure 5.28: Cumulative regret of all 4 agents on short path

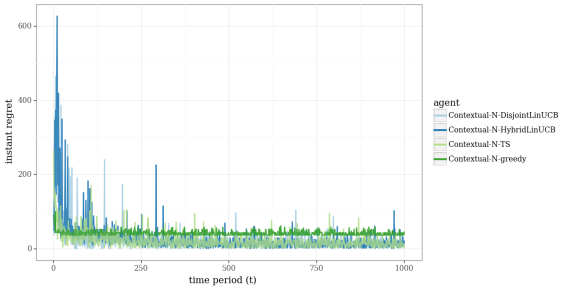


Figure 5.29: Instant regret of all 4 agents on short path

It is clear that the greedy agent gives worse regret regardless of path, but the best agent depends on the selection of path. For longer routes, hybrid LinUCB performs slightly better than other agents, whereas for the short path, Thompson Sampling has the lowest cumulative regret. Looking at the cumulative regrets of UCB-based algorithms, one thing to notice is that hybrid LinUCB seems to take advantage of population data when it is compared with disjoint LinUCB. The exception here is the short path, where the edges are located in the city center and expected to have similar population data. In this case, we avoid claiming that hybrid LinUCB is worse than disjoint LinUCB, since the inclination of cumulative regret plots suggests that they are likely to converge at some point in the future. However, it is also obvious that using population data for navigation in the city center offers no significant contribution.

Another thing to infer from the plots is that short paths do not need much optimism. The optimism of UCB-based algorithms is determined by tuning the α value and this implies that UCB-based algorithms need α tuning for different paths. This situation makes UCB-based algorithms ineffective for navigation purposes. Thus, Thompson Sampling can be preferable considering its performance even for longer routes.

The point of performing all these experiments is to obtain zero regret consistently such that cumulative regret converge to some value. With longer routes, none of the agents have reached zero regret at any point of time. With the short route, all

agents except for greedy agent have reached zero regret at some point but this time not consistently. All these findings suggest that the data contains non-linear energy consumption in contexts, since the predictions of agents do not converge to the mean energy consumption. One reason for not having a linear energy consumption in contexts can be the auxiliary energy consumption such as air conditioning. Another cause can be the sampling rate of the simulation. Sampling in every 1 second might not be sufficient to obtain close results to the real world.

6
Conclusion

6.1 Conclusion

In this thesis, we studied the contextual combinatorial multi-armed bandits in order to address the energy-efficient shortest path problem. Four different agents are implemented to learn the parameters of energy consumption models of each edge in a road network graph: Thompson Sampling, Hybrid LinUCB, Disjoint LinUCB and greedy. The experiments are conducted on the map of Luxembourg city with simulated energy consumption data. To determine whether contextual information helps agents to learn better parameters of the model, we use three kinds of context extracted from simulated data: squared speed, acceleration, and deceleration.

After tuning priors for Thompson Sampling and greedy agents and exploration coefficient for hybrid and disjoint LinUCB, we found that contextual algorithms result in lower cumulative regret in general. Overall, hybrid LinUCB performs best on long routes whereas Thompson Sampling is the best agent when the route is relatively shorter. On the downside, neither of the agents has achieved zero instant regret consistently. This situation can be caused by several things. First, our simulated data might not be linear in contexts. In this case, one can either increase the sampling rate of the simulation and preprocess all the output again to check whether the resulting energy consumptions are linear in the chosen contexts or implement other agents that do not require linearity in rewards. Second, computing mean energy consumptions of edges and using them in the regret calculation might not be suitable for evaluating contextual algorithms. In this case, some other regret calculation methods can be employed. In summary, contextual combinatorial multi-armed bandits seem promising for addressing the energy-efficient shortest path problem if the specified issues are resolved.

6.2 Future Work

In order to enhance the learning, we could change the sampling method. Currently, we are sampling i.i.d. from history and the sampled contexts of consecutive road segments might be very different from each other. However, it is more rational to think that consecutive edges might have similar contexts. One way to achieve this dependence could be to define time intervals such that rush hours are captured. Then, only sampling from the data created at the selected time interval for each edge might lead to similar contexts. Also, for the regret calculation, the mean en-

ergy consumption of the sampled time interval can be used as edge weight for finding the optimal path. This way, we might suffer less from outliers. Another advantage of adopting this approach is the opportunity to supply time interval information to the hybrid LinUCB agent as another common feature.

Increasing the sampling rate of the simulation and regenerating the energy consumptions might help us to achieve the desired linearity in rewards. However, this would significantly increase the time complexity of preprocessing of the simulation output. If it is not possible to reach the desired linearity of the data, we can adopt some other learning algorithms that do not require the linearity assumption.

Lastly, if a better solution is obtained by applying the above suggestions, then it is also advised to conduct the similar experiments with high precision real data to assess the applicability to real life scenarios.

References

Basso, R., Kulcsár, B., Egardt, B., Lindroth, P., & Sánchez-Díaz, I. (2019, 04). Energy consumption estimation integrated into the electric vehicle routing problem. *Transportation Research Part D: Transport and Environment*, *69*, 141-167. doi: 10.1016/j.trd.2019.01.006

Codecá, L., Frank, R., Faye, S., & Engel, T. (2017). Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, *9*(2), 52–63.

Demir, E., Bektaş, T., & Laporte, G. (2014, 09). A review of recent research on green road freight transportation. *European Journal of Operational Research*, *237*, 775–793. doi: 10.1016/j.ejor.2013.12.033

Deshmukh, A., Dogan, U., & Scott, C. (2017, 05). Multi-task learning for contextual bandits. *Advances in neural information processing systems*, 4848-4856.

Dijkstra, E. (1958, 11). A note on two problems in connexion with graphs. *Numb. Math.*, *1*.

Genikomsakis, K., & Mitrentsis, G. (2017, 01). A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. *Transportation Research Part D: Transport and Environment*, *50*. doi: 10.1016/j.trd.2016.10.014

Kurczveil, T., Álvarez López, P., & Schnieder, E. (2014, 11). Implementation of an energy model and a charging infrastructure in sumo. In (p. 33-43). doi: 10.1007/978-3-662-45079-6_3

Lattimore, T., & Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

Li, L., Chu, W., Langford, J., & Schapire, R. (2010, 02). A contextual-bandit approach to personalized news article recommendation. *Computing Research Repository - CORR*. doi: 10.1145/1772690.1772758

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., ... Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st ieee international conference on intelligent transportation systems.* IEEE. Retrieved from `<https://elib.dlr.de/124092/`

Nicol, O. (2014). *Data-driven evaluation of contextual bandit algorithms and applications to dynamic recommendation* (Unpublished doctoral dissertation).

Qin, L., Chen, S., & Zhu, X. (2014, 04). Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 siam international conference on data mining* (p. 461-469). doi: 10.1137/1.9781611973440.53

Reuter, H., Nelson, A., & Jarvis, A. (2007, 10). An evaluation of void-filling interpolation methods for srtm data. *International Journal of Geographical Information Science*, *21*, 983-1008. doi: 10.1080/13658810601169899

Russo, D., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2017). A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038*.

Sutton, R., & Barto, A. (1998, 02). Reinforcement learning: An introduction. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, *9*, 1054. doi: 10.1109/TNN.1998.712192

Thompson, W. (1933, 01). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*, 285-. doi: 10.1093/biomet/25.3-4.285

Wang, S., & Chen, W. (2018, 03). Thompson sampling for combinatorial semi-bandits. *arXiv preprint arXiv:1803.04623*.

Wu, X., Freese, D., Cabrera, A., & Kitch, W. (2015, 01). Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, *34*. doi: 10.1016/j.trd.2014.10.007

Åkerblom, N., Chen, Y., & Chehreghani, M. (2020, 03). An online learning framework for energy-efficient navigation of electric vehicles. *arXiv preprint arXiv:2003.01416*.