Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Charging Recommender for Electric Taxis

Martin Vybíralík

Supervisor: Ing. Marek Cuchý
May 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Vybíralík Martin**	Personal ID number:	**474403**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Charging Recommender for Electric Taxis

Bachelor’s thesis title in Czech:

Nástroj pro doporučování nabíjení pro řidiče elektrických taxi

Guidelines:

The use of electric vehicles brings in a number of challenges as is the relatively low capacity of the batteries of current electric vehicles and long charging times which together presents the user with the problem of carefully planning the charging stops. In contrast with typical driver of electric vehicle, the taxi driver does not know the routes he/she will drive much in advance. Therefore, he/she needs to plan the charging in time and place when the lost revenue caused by time spent with charging is expected to be minimal. The objective is to design and implement an algorithm able to recommend such charging options.

1. Survey existing algorithms solving similar problems.
2. Survey and analyse existing available data sets usable for charging recommendations.
3. Define the problem of charging recommendations for electric taxis.
4. Design an algorithm solving the problem.
5. Implement the designed algorithm.
6. Evaluate the implemented algorithm.

Bibliography / sources:

[1] Tseng, C. M., Chau, S. C. K., & Liu, X. (2018). Improving viability of electric taxis by taxi service strategy optimization: A big data study of new york city. IEEE Transactions on Intelligent Transportation Systems, 20(3), 817-829.
[2] Rong, H., Zhou, X., Yang, C., Shafiq, Z., & Liu, A. (2016, October). The rich and the poor: A Markov decision process approach to optimizing taxi driver revenue efficiency. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (pp. 2329-2334). ACM.
[3] Sun, L., Yang, J., & Yang, Z. (2013, June). Optimal charging strategy of plug-in electric taxi. In 2013 10th IEEE International Conference on Control and Automation (ICCA) (pp. 1532-1537). IEEE.

Name and workplace of bachelor’s thesis supervisor:

Ing. Marek Cuchý, Department of Computer Science, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **09.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Marek Cuchý	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

Firstly, I would like to thank my supervisor Ing. Marek Cuchý for his helpful advice and observations during consultations of the thesis. Furthermore, I would like to express my gratitude for the endless patience and support of my girlfriend and family.

Declaration

Abstract

This thesis deals with the optimization of an electric taxi driver's strategy in charging, passenger approaching, or waiting in potentially favorable locations. It focuses primarily on recommending charging actions and taxi driver's decisions concerning the maximization of a driver's potential profit. Firstly I focused on a general introduction to the topic of electric vehicles together with a research of the state of the art in the taxi movement strategy recommending field. Then I introduced two historical taxi trip data sets and defined the recommendation problem as a Markov Decision Problem (MDP). An essential part of the presented solution method is an estimation of parameters such as a passenger pick-up and drop-off probability connected with particular locations on a planning map. Subsequently, I proposed and implemented an algorithm based on dynamic programming generating a policy for an electric taxi driver. Finally, I experimentally showed the performance of my solution working in different environments compared with a base model of an electric taxi driver behavior. Experiments showed that my algorithm outperforms the base model in several fields, such as a total taxi driver's profit, distance to the next passenger, or charging station choice efficiency.

Keywords: electric vehicles, taxi, recommending, planning, charging, dynamic programming, MDP, movement strategy, decision making

Supervisor: Ing. Marek Cuchý

Abstrakt

Tato bakalářská práce se zabývá optimalizací strategie řidiče elektrického taxi ve smyslu plánování nabíjení, přibližování se klientům či čekání na klienty na potenciálně výhodných místech. Cílí především na doporučování místa a času nabíjení, ale také jednotlivých rozhodnutí řidiče se snahou maximalizovat jeho zisk. Nejprve jsem se zaměřil na obecný úvod do tématu elektrických vozidel, společně s průzkumem dostupných zdrojů zabývajících se vytvářením strategií pro pohyb nejen elektrických, ale i standardních (se spalovacími motory) taxi. Poté jsem představil dva rozsáhlé data sety historických cest vozidel taxi. Následně jsem celý problém definoval pomocí frameworku MDP. Důležitou částí řešení je rovněž odhad potřebných parametrů kupříkladu pravděpodobnosti vyzvednutí a vysazení zákazníka na konkrétních místech. Závěrem jsem navrhl a implementoval algorritmus založený na dynamickém programování, který generuje navrhovanou sekvenci kroků, jichž by se měl řidič taxi držet. Experimentálně jsem ukázal jeho výsledky v různých prostředích v porovnání se základním modelem chování řidiče elektrického taxi. Provedené experimenty ukázaly, že má metoda překonává zvolený základní model, jak v aspektu celkového přijmu řidiče, ve vzdálenosti nutné urazit k místu vyzvednutí pasažéra, ale také v efektivitě výběru nabíjecí stanice.

Klíčová slova: elektrická auta, taxi, doporučování, plánování, nabíjení, dynamické programování, MDP, strategie pohybu, rozhodování

Překlad názvu: Nástroj pro doporučování nabíjení pro řidiče elektrických taxi

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Taxis undoubtedly represent an essential part of public transportation all around the world. Compared to trains or busses, they provide considerably higher time and space flexibility for customers. Mainly in large cities such as New York, Peking, Shenzhen, or Hong Kong, taxi or taxi based services still increases in time.

Even though a number of well known yellow cabs cruising streets of New York dropped 25 % since 2010 [18], they were replaced by another comparable way of transport known as ride-hail services. They use online applications to connect passengers and local drivers using their vehicles [27]. Representatives of these services are, for example, Uber or Lyft. According to the New York Mobility Report from 2018, a growth of these services escalated in 2016, counting 92.5 million trips exceeding the decline of yellow cabs [18].

Furthermore, China's ride-hailing market was marked as the fastest-growing compared to the rest of the world with the most significant future expectation of a growth [19].

On the other hand, the dark side of using taxis and generally cars as a primary way of public transport is naturally a production of exhaust gas emitted as a result of a combustion of fuels [22]. In connection with the rising trend of ride-hailing and taxi-like services, ecology became a discussed issue years ago. For example, according to statistics from the United Kingdom, an amount of CO_2 produced by taxi-like services is still rising [15]. To cope with this problem, some cities such as Shenzhen in China have already introduced an effort to reduce emissions produced by public transportation by involving electric vehicles [30].

1.1.1 Electric Vehicles

Electric vehicles represented one of the main courses in a battle against car emissions in the previous decade. Cities mentioned above were introducing programs to support an infrastructure, which is essential for electric transportation, including building a network of charging stations or parking spots. Nevertheless, not only these cities tried to prepare for still likely electric future [6]. For example, London introduced its electric vehicle infrastructure delivery plan [4]. The Government of Canada announced its Zero-Emission Vehicle Infrastructure Program in which they provide funding of electric vehicles infrastructure projects [17].

Even though country governments and many other organizations make a significant effort in this field, there is still a lack of charging stations throughout the world, mainly in areas where electromobility has not expanded. In connection to this, it is also necessary to mention differences in manners of charging electric vehicles and casual refueling at filling stations. I have found out three main circumstances as representatives of these differences. The first example is that multiple electric car users also own their home chargers as its price is around hundreds of dollars [20]. Secondly, a constrained driving range that holds around 250 Km for an average electric vehicle [26]. Last but not least is charging time, which has been depressed as much as possible in recent years. Nevertheless, the best results are still around half of an hour for the fastest chargers [26].

These facts together lead to some assumptions connected with charging manners. The first one is that drivers whose daily driving distance keeps under a driving range of their cars will rarely need to visit charging stations because of their home chargers. A different one is that drivers with a need to charge their vehicles outside of their homes have to spend not little time waiting. The whole issue of charging stations and the efficiency of charging electric cars is still the subject of many studies [31], [32].

1.2 Scope of the thesis

In my thesis, I focus on individual electric taxis or taxi-like services involving the event of charging into planning their daily shift routes to maximize their potential profit. According to found data [14] and a piece of information received from Prague taxi company Liftago, a regular taxi's daily driving range can be around 250 Km for a one-shift taxi or even around 400 Km for a two-shift taxi. Considering increased charging times, a limited number of charging stations, and a constrained driving range, planning of charging for electric taxis becomes a real issue as drivers could lose potentially convenient time by inappropriately planned charging actions.

To support the trend of electrification in public transport, I decided to develop a recommending framework for electric taxis easily extensible for data sets from different cities. A solution will receive a historical taxi trip data set together with available charging stations within a specific region. It will recommend time and place to charge a taxi driver's car concerning a profit loss minimization. It should also assist in planning driver's routes while searching or waiting for passengers or even give him a helping hand in deciding whether to accept passengers. Another self-offering usage is assisting in a pause planning for taxi drivers, which can be connected with a stop for charging a vehicle or waiting in a potentially favorable place concerning a distance to the next passenger.

The recommending system should make recommendations based on the assumption that taxi demand varies according to a location in a city and a part of a day the taxi driver operates. There are some bright candidates for potentially vantage locations in means of picking-up solvent passengers such as an airport, railway station, bus station, etc. It is plausible to identify these places from large taxi trip data sets by estimating a probability of picking up a passenger in individual locations. Such an estimation should be done in several estimation periods, representing a varying time factor of taxi demand. The final recommending algorithm will establish a policy maximizing a taxi driver's potential profit.

To sum up, my thesis's principal goal is to propose, describe, formalize, implement, and evaluate the algorithm corresponding with the previous description. The first part of the thesis is dedicated to a related work with a comparable scope as mine. As mentioned above, an indispensable part of the proposed solution is a study of sizeable historical taxi trip data sets and parameter estimation essential for the recommending algorithm. An introduction of such data sets is followed by a small section pointing out a basic theory of Markov Decision Processes (MPD), which is a fundamental framework used in the subsequently presented problem formalization, algorithm proposal, and implementation. Finally, I concentrated on an experimental performance evaluation comparing my solution with a base electric taxi driver behavior model.

Chapter 2
Related Work

While researching the topic, I have encountered several papers dealing with comparable problems to mine. Many of them consider ordinary taxis instead of electric ones. Nevertheless, they are using methods that enable me to make a planning of charging kind of enrichment of those previously established methods for taxi recommending systems.

There are also papers dealing exclusively with a recommendation of a proper charging station in time to minimize a charging cost, which does not entirely correspond with my definition of the problem as I focus on optimizing a complete taxi driver's profit.

2.1 A Cost-Effective Recommender

This paper represents one of few cases touching the problem of taxi movement recommending systems that are not using the MDP framework [12]. They introduce and define a net profit objective function which can assign potential profit to any *Route*(path) in a presented *Road Segment Network*. The profit of the whole *Route*(consisting of *Road Segments*) is calculated as a sum of potential earnings and potential costs of individual *Road Segments*, which are estimated from historical data. To receive the optimal *Route*, they first introduce a brute force BFS (Breath-First-Search) algorithm, searching for *Routes* maximizing the profit. Then they also propose a new recursive recommendation strategy based on recursive trees, which effectively finds the best *Route* concerning the driver's profit.

Such an approach is unfortunately not immediately applicable to my problem representation as it considers only a limited length of a planned path and so makes it impossible to plan charging actions for a future taxi demand situation.

2.2 The Rich and the Poor

As written in an introduction of this paper [5], they are the first to propose a method of a recommendation of a movement for ordinary taxi drivers concerning a more extended time interval than just an immediate next trip.

Their chief innovation prevails in taking a temporal variation of a passenger distribution, and the influence of a passenger destination on the following drivers moves into consideration. To achieve this goal, they model the problem as MDP, where each state is a combination of a current grid (partitioning the entire area into several grids), time, and a driving direction. Available actions are traveling to a subset of neighboring cells and cruising through them. More specifically, they define various possible courses of moves on the grid world, and then they try to map these directions on a real-world map. They also partition planning interval into discrete time slots for which they estimate needed parameters from historical data independently.

They define two possible situations when transferring from one to another state. The first one is picking-up passengers, traveling to some location, and receiving a reward. The second is just moving to a chosen next location. All of these events happen with some probabilities estimated from historical taxi trips. An objective is to maximize the total expected reward. A result algorithm yields a policy based on dynamic programming principles.

The solution presented in the paper contains several key ideas, which inspired me in designing my recommending algorithm – namely, the MDP problem representation, or the dynamic programming approach in receiving the policy. The most significant limitation of the proposed solution I can see is partitioning into one-hour time slots and generating optimal policies only within these periods. When considering charging electric vehicles, it is essential to plan within a more significant time interval. An ideal case would be to prepare recommendations for the whole time which taxi driver intends to operate.

2.3 Optimization of the Revenue

This paper directly extends the previous one. The most significant contribution is in a detailed case study [8] of a New York Taxi trip data set and an application of methods proposed in [5]. It focuses on differences in optimal policies in distinct parts of a day – daytime, nighttime and different parts of a week – weekday, weekend.

2.4 Optimal Charging Strategy

Lihao Sun [10] deals with different issues in his paper. He focuses directly on finding the most suitable charging strategy reducing the charging cost of an individual taxi driver. His primary criterion is the current electricity price. The goal of methods introduced in this paper is to minimize a general price of charging within some time interval by selecting an appropriate charging station.

Planning is done by partitioning time into discrete time intervals. An interval in which charging action is inevitable is computed according to the current state of charge (SOC). Once a charging step is done, remaining operation slots are updated.

Comparing to my subject of research, Lihao Sun focuses merely on minimizing the total cost of charging activity and planning this activity without any respect for other taxi driver's needs. This approach seems to me without any further study or statistical background a bit shortsighted as ignoring taxi driver's incomes play undoubtedly an essential role in planning his future steps and, from my point of view, should be involved in the charging recommender.

2.5 Improving Viability of Electric Taxis

Paper written by Chien-Ming Tseng [13] is, by its scope, the closest to my thesis. I inspired by this paper in several parts of my work and tried to introduce and implement improvements in Chien-Ming Tseng's methods and problem representation.

Even though their main objective is to increase a taxi driver's willingness to use electric vehicles, they propose a recommending framework based on the MDP representation, which is useful for my work.

A state representation consists of the current time, using one-minute time slots, a location determined by the closest OSM node (Open Street Map)[1], and a battery SOC. Actions are generally described as driver's decisions to travel to the next location, which more precisely means a neighboring node in graph generated from OSM data. Another possible choice is to charge for some time.

Formally each action is determined by two nodes (start, destination) and time of charging. The time of charging is zero if no charging is needed. When greater than zero, a taxi should go to the most adjacent charging station before continuing its journey to a destination node. Transition function determines the probability of picking-up passengers when transferring between states. They also point out several already researched problem characteristics such as that searching the next passenger in a drop off place of a previous one leads to higher profits or that predicting traffic conditions leads to more favorable results for the entire system.

Paper uses New York City taxi trip data set from 2013 to estimate probabilities of picking-up passengers in individual nodes. Another critical feature is disabling some actions (picking-up passengers, driving to some location) concerning an insufficient SOC. Unlike Meng Qu [12], who disables a customer's refusal, Chien-Ming Tseng defines three possible scenarios in any state (node, time, SOC).

- Picking-up a passenger and delivering him to his drop-off destination.
- Refusing a passenger due to a low SOC followed by traveling to the nearest charging station.
- Not picking-up a passenger.

[1]`https://www.openstreetmap.org`

In subsequent chapters, I introduce several substantial improvements that put significantly more emphasis on charging electric taxis. The most significant limitation of Chien-Ming Tseng's paper I can see is in choosing the nearest charging station exclusively. Such an approach seems inefficient as the closest charging station is not necessarily the most favorable choice. Furthermore, I also tried to enrich a set of possible actions by ***Staying in a location***, which corresponds a bit more with a real-world situation. On some occasions, it may be convenient not to drive all the time but also to wait in potentially vantage place [1].

Despite the discussed limitations, I was inspired by the paper in the case of Equation 4.1 computing taxi driver's final reward or in the estimation of several parameters.

Chapter 3

Data Sets

As it seems from the related work, creating recommendations for drivers of electric taxis is closely connected with historical taxi trip data sets. They are used for modeling a taxi demand in individual cities, which is a critical factor to cope with.

In this chapter, I introduce two of these historical taxi trip data sets, followed by two corresponding lists of charging stations, which form another indispensable part of the charging recommendation problem.

3.1 Taxi Trip Data Sets

The primary purpose of historical taxi trip data sets is a possibility to mine information about concrete historical taxi trip pick-up and drop-off spots. Concerning this data, it is possible to estimate the probability of picking-up passengers in some particular location and time.

When researching the related work, I met with papers based merely on a single one data set (New York taxi trip data set). This fact gave me an idea to create a solution with an interface for input data, which would be easy to implement and enable simple extensions to cover other cities.

- *pick_up_longitude*
- *pick_up_latitude*
- *drop_of_longitude*
- *drop_of_latitude*
- *trip_distance*
- *start_time*
- *end_time*

It should be possible to parse all of these parameters from any input data set. Concerning a trip distance, there are two possible ways to receive it. Firstly, it can be computed from pick-up and drop-off coordinates. On the other hand, some data sets contain a real recorded distance of a trip. Such a distance contributes much more to modeling the real-world taxi demand than estimating it, for example, from the shortest path[1] between a pick-up and

[1] A* algorithm `https://www.cs.auckland.ac.nz/courses/compsci709s2c/resources/Mike.d/astarNilsson.pdf` computes all the shortest paths in my thesis

drop-off location. The proposed interface enables the use of both of these cases.

3.1.1 Prague

The first is a data set from Prague received from a company Liftago [9]. It contains historical taxi trips recorded in February 2019, counting 42 872 trips.

As observable in Table 3.1, it almost copies the interface for a taxi trip data sets declared above. However, its most significant shortage is a limited size, which is reflected in a bit distorted results of parameter estimation introduced in Chapter 5.

orderID	0d7481a7e17c10bc
requestedPickUpLat	50.080
requestedPickUpLon	14.418
requestedDestinationLat	50.080
requestedDestinationLon	14.445
rideDistance	4801
rideStartedAt_UTC	2019-02-02 16:11:24
rideFinishedAt_UTC	2019-02-02 16:38:32

Table 3.1: Prague taxi trip data set example.

Closer analysis of this data set confirmed the assumption that the highest taxi demand is in the center of the city and becomes sparser moving to the suburbs. Nevertheless, as visible in Figure 3.1, there is also a significantly increased taxi demand near Prague airport. An average distance of one trip is around 8.3 Km.

3.1.2 New York

The second data set contains 25 960 490 taxi trips recorded during 2018 in New York [16], which is leading to much better conditions for both the parameter estimation in Chapter 5 and evaluation in Chapter 8. Actually, I decided to limit the size of the data set to one million randomly chosen taxi trips to avoid unnecessary memory consumption.

Because a trip record described in Table 3.2 does not contain longitudes and latitudes, pick-up and drop-off places of individual trips are parsed from an enclosed look-up table containing distinct zones in the form of approximate address as can be seen in Table 3.3.

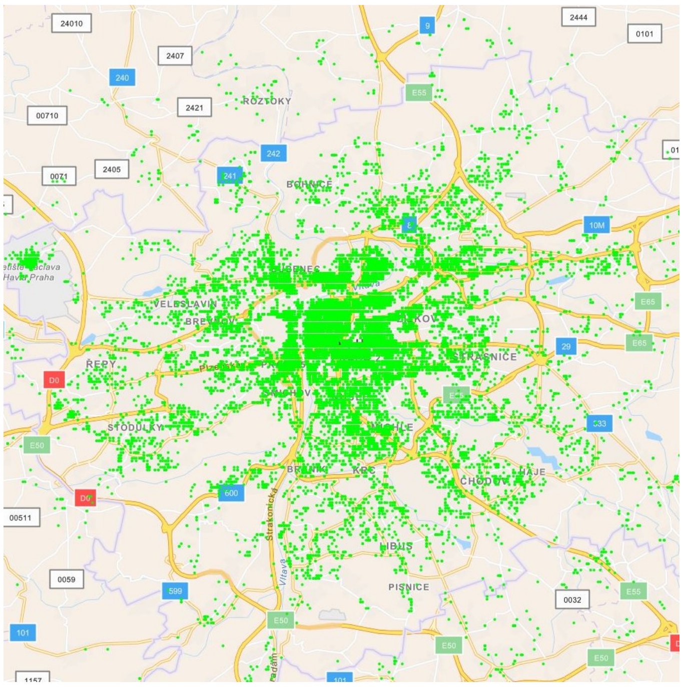


Figure 3.1: Taxi demand in Prague historical taxi trip data set.

VendorID	2
tpep_pickup_datetime	11/04/2018 12:32:24 PM
tpep_dropoff_datetime	11/04/2018 12:47:41 PM
passenger_count	1
trip_distance	1.34
RatecodeID	1
store_and_fwd_flag	N
PULocationID	238
DOLocationID	236
payment_type	2
fare_amount	10
extra	0
mta_tax	0.5
tip_amount	0
tolls_amount	0
improvement_surcharge	0.3
total_amount	10.8

Table 3.2: New York taxi trip data set example.

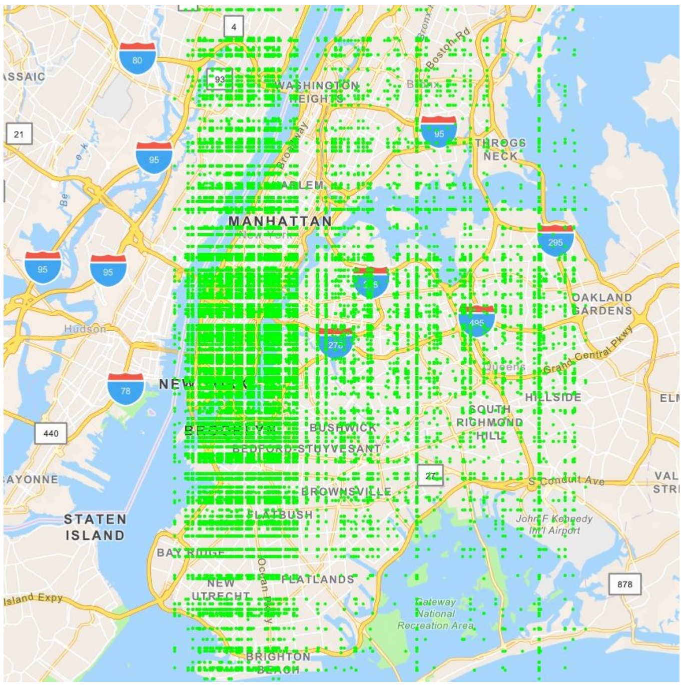


Figure 3.2: Taxi demand in New York historical taxi trip data set.

LocationID	2
Borough	Queens
Zone	Jamaica Bay
service_zone	Boro Zone

Table 3.3: New York zone lookup example.

To retrieve desired longitudes and latitudes, I used API[2] performing the conversion from a given address.

Unlike the previous data set, here I am served with much more information such as passenger count, fare amount, tip, and others, which can be used in future development.

A structure of the New York data set significantly differs from the Prague one. While in Prague, the taxi demand smoothly descends from the city center to the suburbs, the majority of historical trips in New York comes from the area of Manhattan and Brooklyn, as shown in Figure 3.2. Also, an average taxi trip distance is around 2.9 Km, which is 5 kilometers less than in Prague. It is probably caused by the fact that using taxis is a much more common way of everyday transport in New York than in Prague.

Such a shape of a taxi demand also profoundly affects the behavior of charging recommender based agents, as can be seen in the evaluation Chapter 8.

[2]`https://opencagedata.com/`

3.2 Charging Station Data Sets

All data connected with charging stations were received from an online OPEN CHARGE MAP API[3]. After writing a simple request specifying the desired area, a JSON data file containing available charging stations and their charging connections (20 KW, 50 KW, ...) is generated. An example can be seen in Appendix A. After receiving such a file, a set of all available charging stations is parsed from it. Each charging station has one or more charging connections characterized by its power in KW.

For an evaluation presented in Chapter 8, I chose 75 random charging stations in the city of Prague and the same in New York. There were two main reasons for such a step. First of all, reduction of a space complexity of the algorithm proposed in Chapter 6 and secondly, the fact that during the study of the data set, I encountered quite a vast amount of charging stations close to each other so they would have a minor effect to final results.

[3] `https://openchargemap.org/site/develop`

Chapter 4

Problem Representation

In this chapter, I introduce several basic building blocks that create a formal definition of the charging recommendation problem. Namely, it is a general environment description of the road network, which is closely connected with the subsequently introduced MDP framework problem representation.

4.1 Environment

Naturally, the first concept to be described is an environment in which a taxi driver will operate in the future recommending algorithm. In this case, the environment signifies a mapping of a real-world road network into data structures usable in planning.

A representation of the road network plays undoubtedly an essential role in any route planning or recommending. This is why I decided to introduce and evaluate more forms of environments in Chapter 6 to determine the most suitable one for the charging recommending problem. Here I list all requirements for a global environment, which serve as a template for following concrete environment implementations.

1. Environment is a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a finite set of nodes $\mathbf{n}$ and $\mathbf{E}$ is a finite set of edges $\mathbf{e}$.

2. Each node $\mathbf{n} \in \mathbf{V}$ is associated with a longitude and latitude, which represents mapping into a real-world location.

3. Each edge $\mathbf{e} \in \mathbf{E}$ represents a connection of two nodes $\mathbf{n} \in \mathbf{V}$ and so it can be considered as a path from node $\mathbf{n}_i$ to a neighbouring node $\mathbf{n}_j$.

4. Each edge $\mathbf{e} \in \mathbf{E}$ is associated with three non negative integers l, d and v, where l signifies a length, d time needed to transfer the edge $\mathbf{e}$ and v is a speed on the edge.

5. $\mathcal{N}[\mathbf{n}_i]$ is a function returning a set of all nodes $\mathbf{n}_j \in \mathbf{V}$ connected with a node $\mathbf{n}_i$ by an edge $\mathbf{e}$ – neighbouring nodes.

6. The set of nodes $\mathbf{V}$ is enriched by the set of charging stations $\mathbf{c} \in \mathbf{C}$ which are fitted into the closest node $\mathbf{n} \in \mathbf{V}$.

4.2 Markov Decision Process (MDP)

As announced in previous sections, I adopted MDP's formalism to describe and define the charging recommendation problem. I combined and adapted approaches used in [13] and [5] to avoid shortcomings mentioned in Chapter 2. Nevertheless, firstly I would like to bring a brief introduction to a concept and notation of the MDP framework introducing the main building blocks used in the thesis.

4.2.1 Definition

Markov Decision Process represents a mathematical framework firstly introduced in the 1950s. Nowadays, it is used as a modeling framework in reinforcement learning but also in computational finance, gambling, graph theory, and many others [7].

Mathematically there are several possible variations and distinct definitions of MDP. The one used in my work is called a finite discrete-time fully observable MDP, defined in [7] as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{T}, \mathcal{R} \rangle$, where:

- $\mathcal{S}$ – A finite set of states.
- $\mathcal{A}$ – A finite set of actions.
- $\mathcal{D}$ – A finite sequence of natural numbers indicating a timestamp.
- $\mathcal{T}$ – A transition function specifying the probability of going to a state s_2 when executing an action a in a state s_1.
- $\mathcal{R}$ – A reward function returning an expected reward for a transition from a state s_1 to a state s_2 executing an action a.

4.2.2 Policy

Another concept essential to define for future use is a policy π. According to [7] it denotes the probability distribution of choosing an action a in a state s and timestamp t. A reward $\mathcal{R}^{\pi}$ is then described as an amount of utility received by executing the policy π. In other words, it is the reward received by performing all actions according to the policy π starting from the current state s and continuing further in time. An optimal MDP solution is a policy π^* maximizing the reward function $\mathcal{R}$ resulting in $\mathcal{R}^{\pi^*}$ [7].

4.3 Problem Definition

In this section, I model the charging recommendation problem representation as a finite discrete-time fully observable MDP. As I intend to provide a driver with recommendations during one complete shift (length of a shift is further discussed in Chapter 8), the state space of the problem is finite with discrete timestamps covering the entire shift. Moreover, a taxi driver has various actions to perform, such as going to a neighboring location, staying in a particular place, or going to a charging station. The goal of the recommending algorithm should be to provide a driver with recommendations of these activities in any possible state. It should also be maximizing his potential profit, which perfectly fits an effort of finding policy π^* described above.

- **$\mathcal{S}$ – A finite set of all possible states**

  Each state is a tuple of:

  - *Location* – An association of a taxi's GPS location with the nearest environment node $\mathbf{n} \in \mathbf{V}$, as defined in Section 4.1.
  - *Time* – A discrete time slot indicating a number of minutes passed after the start of a shift (planning). Represented by a single integer $t, t \in \mathcal{D}$.
  - *SOC* – A discrete state of charge of a battery in %. Represented by a single integer $b, b \in \mathbf{B} = \langle 0, 100 \rangle$.

  Now I can write $s = \langle n, t, b \rangle$ as a full description of one state. Such a state representation corresponds with the one in [13].

- **$\mathcal{A}$ – A finite set of all possible actions a taxi driver can take**

  Each action a is a tuple $\langle \mathbf{n}_s, \mathbf{n}_d, t_a, b_a \rangle$ where:

  - $\mathbf{n}_s$ – A starting node $\mathbf{n}_s \in \mathbf{V}$ (current state node).
  - $\mathbf{n}_d$ – A destination node $\mathbf{n}_d \in \mathbf{V}$ (node in which action results in).
  - t_a – A duration of an action (in minutes).
  - b_a – An amount of consumed, recuperated or charged energy (in %).

  In each state there are several kinds of actions a taxi driver can take:

  - ***Driving to a next location*** – As declared in Section 4.1, each node $\mathbf{n} \in V$ is associated with a not empty set $\mathcal{N}[\mathbf{n}]$ of its neighbours i.e. nodes that are connected by an edge $\mathbf{e} \in \mathbf{E}$.
    Each of these neighbouring nodes $\mathbf{n}_d \in \mathcal{N}[\mathbf{n}]$ represents one feasible action of ***Driving to a next location*** from node $\mathbf{n}$. Such an action takes time t_a equivalent to a transition time d of an edge $\mathbf{e} \in \mathbf{E}$ between these nodes and produces energy consumption b_a estimated in Chapter 5. It can be described as:

    $$\langle \mathbf{n}_s, t, b \rangle \xrightarrow{\langle \mathbf{n}_s, \mathbf{n}_d, t_a, b_a \rangle} \langle \mathbf{n}_d, t + t_a, b + b_a \rangle$$

  - ***Staying in a location*** – In some cases, it can be convenient for a taxi driver to stay in a location with potentially high client concentration. In such a case, starting node $\mathbf{n}_s$ and destination node $\mathbf{n}_d$ are equal. t_a represents a waiting time set to the minimal time unit – one minute and energy consumption b_a equals zero.

$$\langle \mathbf{n}_s, t, b\rangle \xrightarrow{\langle \mathbf{n}_s, \mathbf{n}_s, t_a, 0\rangle} \langle \mathbf{n}_s, t + t_a, b\rangle$$

- ***Going to a charging station*** – In each state (except the one when a driver is already at a charging station) a driver can decide to go to a charging station. This action can be described as driving from a current node $\mathbf{n}_s$ directly (using the shortest path) to a chosen charging station node $\mathbf{c} \in \mathbf{C}$.

  A driver can decide to go to any possible charging station to maximize his future reward. The policy of choosing the best charging station is described in Section 6.3.

  Action time t_a is received by a sum of d parameters of edges $\mathbf{e}$ forming the shortest path between a current node $\mathbf{n}_s$ and a charging station $\mathbf{c}$. Consumption b_a is then a sum of consumption on these edges estimated in Chapter 5.

$$\langle \mathbf{n}_s, t, b\rangle \xrightarrow{\langle \mathbf{n}_s, \mathbf{c}, t_a, b_a\rangle} \langle \mathbf{c}, t + t_a, b + b_a\rangle$$

- ***Charging at a charging station*** – As a driver arrives at a charging station, he can also decide how long he wants to charge his car, representing t_a in this type of an action. There are several charging intervals to choose from, and they are computed according to a charging connection type selected. This is more described in Section 8.3.

  Charging intervals are meant to provide a driver with a choice of charging his car from a current SOC to 100 % by steps of 10 %. A state location $\mathbf{c}$ then remains the same, and b_a represents an amount of charged energy in percents again computed concerning a type of used charging connection and chosen charging interval.

  There is also one extra parameter connected with taking an action of ***Charging at a charging station*** and it is the price p for energy charged. The rate for charging varies by the charging connection type concerning the KW and is also described in Section 8.3.

$$\langle \mathbf{c}, t, b\rangle \xrightarrow{\langle \mathbf{c}, n_s, t_a, b_a\rangle} \langle \mathbf{c}, t + t_a, b + b_a\rangle$$

Picking-up a passenger – This is an action that differs from all the others by its significance. First of all, it is not a member of the set of applicable actions $\mathcal{A}$ as it does not depend on a taxi driver's decision, but it is a purely stochastic phenomenon. In other words, a taxi driver can not decide to take this action on his own, but he must be requested to do so. In the case of recommending algorithm, ***Picking-up a passenger*** only reflects in transition probabilities defined below in this section.

This action is used mainly in the evaluation described in Chapter 8, where taxi drivers receive offers from customers and are able to accept or deny them.

There are two possibilities for picking-up a passenger. The first one is to pick-up a passenger right on the street, which implies a zero distance to travel to a pick-up place and going directly to a customer's desired destination. The second one, and these days a much more frequent case, is picking-up customers after a phone call or an application request. This case is connected with an extra distance, which has to be reached to pick-up a passenger, which also brings additional energy consumption. The recommending algorithm proposed in Chapter 6 is able to cope with both of these cases.

Taxi trip length, duration t_a, and consumption b_a are again computed from the shortest path on the environment graph $\mathbf{G}$ connecting a pick-up node $\mathbf{n}_s \in \mathbf{V}$ and a drop-off node $\mathbf{n}_d \in \mathbf{V}$.

$$\langle \mathbf{n}_s, t, b \rangle \xrightarrow{\langle \mathbf{n}_s, \mathbf{n}_d, t_a, b_a \rangle} \langle \mathbf{n}_d, t + t_a, b + b_a \rangle$$

- $\mathcal{D}$ – **Time steps**

  A finite sequence of natural numbers denoting possible time epochs in which actions can be taken, and states can occur. It can be written as $\mathcal{D} = (t_0, t_0 + 1, ..., t_0 + t_{MAX})$, where t_0 is a start time of a taxi drivers shift and t_{MAX} is a length of a shift – time of a taxi in operation. I am using minutes as a real-time unit. This means that t_0 is a number of minutes from the midnight, i.e., $t_0 = 480$ for a shift starting at 8 AM $(8 \cdot 60)$.

- $\mathcal{T}$ – **Transition function** $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$

  A transition function $\mathcal{T}$ as defined in [7] is a mapping specifying a probability $\mathcal{T}[s_i, a, s_j]$ of going from a state s_i to a state s_j, taking an action a. The transition function $\mathcal{T}$ is given by the probability of picking-up a passenger. In the following text I will denote this probability as $\mathcal{P}[s_i, a, s_j]$ where s_i is a starting state, a is a taken action from $\mathcal{A}$, and s_j is a resulting state. It denotes the probability of picking-up a passenger when taking an action a in state s_i, which should result in a state s_j. A value of this probability is estimated in Section 5.

  I decided to partition all feasible actions from $\mathcal{A}$ into two groups concerning the transition function $\mathcal{T}$ definition:

  1. ***Stochastic group***
     - ***Driving to a next location***
     - ***Staying in a location***

     $$\mathcal{T}[s_i, a, s_j] = 1 - \mathcal{P}[s_i, a, s_j]$$

     The transition probability of successfully finishing these actions equals the probability of not picking-up a passenger. Otherwise, a taxi driver will end up in a state determined by a drop-off node, journey duration, and consumption of a taxi trip.

2. ***Deterministic group***
   - ***Going to a charging station***
   - ***Charging at a charging station***

$$\mathcal{T}[s_i, a, s_j] \;=\; 1$$

   In this case, I assume that there is no chance to change a taxi driver's mind after deciding to go to a charging station or to charge in a charging station and accept a customer's offer.

It is essential to mention, that I will estimate $\mathcal{P}[s_i, a, s_j]$ and $\mathcal{T}[s_i, a, s_j]$ from historical taxi trips (Chapter 5). That is why I assume that recommending a taxi driver's route concerning real pick-up nodes $\mathbf{n}_s$ will optimize both cases of picking-up a passenger as described above in this section. Firstly, it optimizes the probability $\mathcal{P}[s_i, a, s_j]$ of picking-up a passenger in a current node $\mathbf{n}$. But it also minimizes a potential distance to be reached to pick-up a passenger who ordered a trip by a phone or any kind of a taxi app.

- $\mathcal{R}$ – **Reward function** $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$

  The reward function $\mathcal{R}[s_i, a, s_j]$ returns an expected numeric reward for transition from a state s_i to a state s_j, taking an action a. Same as in a case of the transition function $\mathcal{T}$, the reward function $\mathcal{R}$ will also have two forms depending on a taken action $a \in \mathcal{A}$. Before defining them, I will introduce several parameters used in their equations:

  - $\mathcal{L}[s]$ – A function returning a node $\mathbf{n} \in \mathbf{V}$ of a state $s \in \mathcal{S}$.
  - $\mathcal{I}[s]$ – A function returning an estimation interval $\mathbf{T}_i$ introduced in Chapter 5 fitting a timestamp t of a state $s \in \mathcal{S}$.
  - $\mathcal{B}[s]$ – A function returning a SOC b of a state $s \in \mathcal{S}$.
  - $\mathcal{E}[\mathbf{n}_i, \mathbf{n}_j]$ – A function returning an energy consumption in % needed for transfer from a node $\mathbf{n}_i$ to a node $\mathbf{n}_j$.
  - $\mathcal{B}^{min}$ – A minimum SOC set experimentally concerning a taxi trip data set structure.
  - $\mathcal{F}[s, \mathbf{n}]$ – A fare received after picking-up a passenger in a state s and delivering him to a desired location $\mathbf{n}$ introduced in Chapter 5.
  - $\mathcal{G}[s]$ – A function returning a set of all nodes $\mathbf{n}_g \in \mathbf{V}$ that figure as a drop-off node in some historical taxi trip that starts in node $\mathcal{L}[s]$. At the same time, it returns only drop-of locations corresponding to trips taken during an estimation interval $\mathcal{I}[s]$ and satisfying at least one of these conditions:
    - $\mathcal{E}[\mathcal{L}[s], \mathbf{n}_g] + \mathcal{B}^{min} > \mathcal{B}[s]$ – An energy consumption of a trip is exceeding a current SOC.
    - $t_t + t > t_0 + t_{MAX}$ – A trip time t_t together with a current timestamp t of a state s is exceeding a shift end time $t_0 + t_{MAX}$.

In other words, it represents a set of all unreachable taxi trip drop-off destinations.

- $\mathcal{H}[s]$ – A function returning a set of all nodes $\mathbf{n}_h \in \mathbf{V}$ with the same description as $\mathcal{G}[s]$, but satisfying both of these conditions:
    - $\mathcal{E}[\mathcal{L}[s], \mathbf{n}_h] + \mathcal{B}^{min} < \mathcal{B}[s]$ – An energy consumption of a trip not exceeding a current SOC.
    - $t_t + t < t_0 + t_{MAX}$ – A trip time t_t together with a current time stamp t of a state s not exceeding a shift end time $t_0 + t_{MAX}$.

  In other words, it represents a set of all reachable taxi trip drop-off destinations.

Now I can introduce recursive equations of reward functions for both groups of actions (***Stochastic*** and ***Deterministic***). These groups are the same as in the case of the transition function $\mathcal{T}$ defined above. The most crucial reason for the definition of two reward functions is again in the zero probability of picking-up a passenger when taking actions from ***Deterministic group***.

1. ***Stochastic group***

$$\begin{aligned} \mathcal{R}[s_i, a, s_j] \;=\; & \mathcal{R}^*[s_j] \cdot (1 - \mathcal{P}[s_i, a, s_j] + \mathcal{P}[s_i, a, s_j] \cdot \mathcal{PG}[s_j]) \\ & + \mathcal{P}[s_i, a, s_j] \cdot \mathcal{RH}[s_j] \end{aligned} \tag{4.1}$$

- $\mathcal{R}^*[s_j]$ – A reward in a state s_j received after taking an action with the highest potential profit – a recursive part of the reward function.
- $\mathcal{P}[s_i, a, s_j]$ – The probability of picking-up a passenger when taking an action a in a state s_i going to a state s_j – estimated in Chapter 5.
- $\mathcal{P}^d[s, \mathbf{n}_d]$ – The probability of picking-up a passenger commuting to a drop-off node $\mathbf{n}_d$ from a state s – estimated in Chapter 5.
- $\mathcal{PG}[s] = \sum_{\mathbf{n}_g \in \mathcal{G}[s]} \mathcal{P}^d[s, \mathbf{n}_g]$ – The probability of unreachable trip with a passenger i.e., passenger commuting to unreachable destination concerning an insufficient SOC or time till the end of a shift.
- $\mathcal{RH}[s] = \sum_{\mathbf{n}_h \in \mathcal{H}[s]} \mathcal{P}^d[s, \mathbf{n}_h] \cdot (\mathcal{F}[s, \mathbf{n}_h] + \mathcal{R}^*[s_d])$ – A reward for all reachable trips with a passenger from a state s in an estimation interval $\mathcal{I}[s]$. Here s_d signifies a drop-off state of a taxi driver after delivering a passenger.

This equation is inspired by the one in [13] and can be interpreted as that the final reward $\mathcal{R}[s_i, a, s_j]$ is a sum of rewards from two possible situations multiplied by its probabilities:

a. Not picking up a passenger and receiving the reward $\mathcal{R}^*[s_j]$ with the probability equal to a sum of the probability of not picking up a passenger and the probability of picking-up a passenger with an unreachable drop-off destination.

b. Picking-up a passenger and receiving a reward $\mathcal{RH}[s]$.

2. ***Deterministic group***

$$\mathcal{R}[s_i, a, s_j] \;=\; \mathcal{R}^*[s_j] - p \cdot t_a \tag{4.2}$$

Here the final reward $\mathcal{R}[s_i, a, s_j]$ consists only of a result state s_j reward and a subtraction of a charging cost p introduced in Section 8.3. Charging cost is set to 0 when taking an action of ***Going to a charging station***.

4.3.1 π^* – Optimal Policy

As declared in the introduction of Section 4.3, the goal of the recommending algorithm is to provide a taxi driver with an optimal policy π^* proposing an action $a \in \mathcal{A}$ in any reachable state, maximizing his potential profit determined by $\mathcal{R}[s_i, a, s_j]$ equations above.

Recommendations concerning such a policy π^* always depend on the current state s. For example, a taxi driver in state $s = (\mathbf{n}, 500, 10)$ could receive a recommendation to go to a concrete charging station as the SOC (10 %) is quite low. Similarly a taxi driver in state $s = (\mathbf{n}_i, 550, 80)$ could be encouraged to go to a next location $\mathbf{n}_j \in \mathcal{N}[\mathbf{n}_i]$. Both of these recommendations are actions maximizing a taxi driver's future potential profit.

Chapter 5

Parameter Estimation

In this section, I introduce and estimate several parameters essential to define the above described MDP representation of the charging recommendation problem. Historical taxi trip data sets introduced in Section 3.1 are used for this purpose.

Unlike [13], where they estimate the majority of parameters depending on a current time t I decided to built the whole idea of the parameter estimation and the recommending algorithm proposal on splitting the sequence of time steps $\mathcal{D}$ into q larger estimation intervals $\mathbf{T}_1, ..., \mathbf{T}_q$. These intervals are of a length r (corresponding with an **ESTIMATION_INTERVAL** in Table 8.1), where:

$$\begin{aligned} \mathbf{T}_i &= \langle t_{i_0}, t_{i_0} + 1, ..., t_{i_0} + r) \\ q &= \frac{t_0 + t_{MAX}}{r} \end{aligned}$$

Here t_{i_0} is a starting time of an interval $\mathbf{T_i}$ and t_0 is a starting time of a taxi driver's shift.

I assume that the transition function $\mathcal{T}$ and values of the reward function $\mathcal{R}$ during individual estimation intervals $\mathbf{T}_i$ are the same. On the other hand, they differ when transitioning from $\mathbf{T}_i$ to $\mathbf{T}_j$. Thanks to this, I can now split historical taxi trips from Section 3.1 into groups $\mathbf{G}_1, ..., \mathbf{G}_q$. Each group $\mathbf{G}_i$ corresponds with its estimation interval $\mathbf{T}_i$ during which trips in it were taken. Then I can estimate all needed parameters separately for each group.

To illustrate such an intention I present an example introducing a simple model situation:

Example: Taxi driver starts his shift at 8:00 AM and wants to drive for three hours (180 minutes). I split his shift into 6 intervals $\mathbf{T_1}, ..., \mathbf{T_6}$, 30 minutes long, where $\mathbf{T_1} = \langle 0, 30), \mathbf{T_2} = \langle 30, 60), ..., \mathbf{T_6} = \langle 150, 180 \rangle$. I get all historical taxi trips and sort them according to a time of a trip into groups $\mathbf{G_1}, ..., \mathbf{G_6}$, where in $\mathbf{G_1}$ there are trips done from 8:00 to 8:30, in $\mathbf{G_2}$ trips from 8:30 to 9:00 e.t.c.. Now I estimate all needed parameters for all of these groups separately.

5.1 $\mathcal{P}[s_i, a, s_j]$ – Pick-up Probability

As described above in Section 4.3, the pick-up probability $\mathcal{P}[s_i, a, s_j]$ is used to define the transition function $\mathcal{T}[s_i, a, s_j]$ of the ***Stochastic group*** (***Driving to a next location***, ***Staying in a location***) of actions and it also figures in Equation 4.1 of the reward function $\mathcal{R}[s_i, a, s_j]$. This makes it one of the key parameters of the whole charging recommendation problem.

It is essential to remind that $\mathcal{P}[s_i, a, s_j]$ is closely connected with the definition of estimation intervals $\mathbf{T}_i$ above. When talking about the real value of $\mathcal{P}[s_i, a, s_j]$, it automatically means the probability of picking up a passenger during an estimation interval $\mathbf{T}_i$. This interval can be received by using previously introduced function $\mathcal{I}[s_j]$ that returns it concerning a timestamp t of a state s_j. The main reason for such an emphasis is the difference in these probabilities between individual estimation intervals $\mathbf{T}_i$.

In papers [5] and [8], this kind of a probability is estimated as a number of pick-ups within a concrete cell of a grid (i.e., larger area) divided by a total number of vacant taxi visits of the cell. This method yields a total number of taxi visits of a particular cell as another parameter to be modeled. To do so, they use historical taxi trip data sets again. They try to estimate a taxi driver's movement i.e., visited locations, from all historical trips, by computing approximate paths between pick-up and drop-off locations. From these paths, it is then possible to parse numbers of visits in individual cells.

In [13], the pick-up probability is estimated as a total pick-up number in one node within some period of time divided by a sum of the pick-up number and a total number of drop-offs of passengers within a surrounding area. The number of drop-offs represents competitive vacant taxi drivers, who decrease the probability of picking up a passenger.

Both of these approaches model the probability of $\mathcal{P}[s_i, a, s_j]$. They estimate it as a ratio between the number of successful pick-ups and a somehow estimated number of taxi visits of an area. From my point of view, this attitude puts two locations of completely different parameters on the same level. For example, let me introduce two locations $\mathbf{A}$ and $\mathbf{B}$, where, in case of location $\mathbf{A}$ a number of pick-ups in some period equals to 1000 and a number of drop-offs equals to 500. In the case of location $\mathbf{B}$, these numbers are 10 and 5. The above-described estimation approach would assign the same pick-up probabilities to both of these places, which does not correspond to reality.

That is the main reason I decided to skip the idea of modeling competition between individual taxi drivers. Instead of this, I estimate the probability $\mathcal{P}[s_i, a, s_j]$ between states s_i and s_j involving just a number of pick-ups in an environment node $\mathcal{L}[s_j] \in \mathbf{V}$ corresponding to a state s_j, a number of pick-ups in its neighbors $\mathcal{N}[\mathcal{L}[s_j]]$ and a total number of pick-ups in all the other nodes $\mathbf{n} \in \mathbf{V}$. The inclusion of neighbors should express the pick-up potential of a larger area than just one node, as in the case of [13].

I define:

- $N_{\mathbf{T}_i}[\mathbf{n}]$ – A number of historical pick-ups in a node $\mathbf{n} \in \mathbf{V}$ during an estimation interval $\mathbf{T}_i$.

- $N_{\mathbf{T}_i}[\mathcal{N}[\mathbf{n}]]$ – A number of historical pick-ups in a set of neighbours $\mathcal{N}[\mathbf{n}]$ of a node $\mathbf{n}$ during an estimation interval $\mathbf{T}_i$.
- $N_{\mathbf{T}_i}[\mathbf{V}]$ – A number of historical pick-ups in a set of all nodes $\mathbf{n} \in \mathbf{V}$ during an estimation interval $\mathbf{T}_i$.

Now I can define $\mathcal{P}(s_i, a, s_j)$ as:

$$\mathcal{P}(s_i, a, s_j) = \frac{N_{\mathcal{I}[s_j]}[\mathcal{L}[s_j]] + N_{\mathcal{I}[s_j]}[\mathcal{N}[\mathcal{L}[s_j]]]}{N_{\mathcal{I}[s_j]}[\mathbf{V}]}.$$

5.2 $\mathcal{P}^d[s, \mathbf{n}]$ – Drop-off Probability

Another parameter already used in Section 4.3 is the drop-off probability signifying the probability of a passenger commuting from a starting node $\mathcal{L}[s]$ to a drop-off node $\mathbf{n}_d$ during an estimation interval $\mathcal{I}[s]$. It could be also described as the probability that a taxi driver in a state $s = \langle \mathbf{n}_s, t, b_s \rangle$, who picked-up a passenger will travel to a node $\mathbf{n}_d$ during an estimation interval $\mathcal{I}[s]$.

I estimate this parameter same as in [13]:

$$\mathcal{P}^d[s, \mathbf{n}] = \frac{N^d_{\mathcal{I}[s]}[\mathcal{L}[s], \mathbf{n}]}{N^d_{\mathcal{I}[s]}[\mathcal{L}[s], \mathbf{V}]},$$

where $N^d_{\mathcal{I}[s]}[\mathcal{L}[s], \mathbf{n}]$ is a number of trips from a node $\mathcal{L}[s]$ to a node $\mathbf{n}$ in an estimation interval $\mathcal{I}[s]$ and $N^d_{\mathcal{I}[s]}[\mathcal{L}[s], \mathbf{V}]$ is a number of all trips from a node $\mathcal{L}[s]$ to any other node $\mathbf{n} \in \mathbf{V}$.

5.3 $\mathcal{E}[\mathbf{n}_i, \mathbf{n}_j]$ – Energy Consumption

An estimation of the function $\mathcal{E}[\mathbf{n}_i, \mathbf{n}_j]$ can be easily simplified into an estimation of an energy consumption b of an edge $\mathbf{e} \in \mathbf{E}$ connecting two neighbouring nodes $\mathbf{n}_k$ and $\mathbf{n}_l$, where $\mathbf{n}_l \in \mathcal{N}[\mathbf{n}_k]$ as introduced in Section 4.1. Function $\mathcal{E}[\mathbf{n}_i, \mathbf{n}_j]$ for any pair of nodes $\mathbf{n}_i, \mathbf{n}_j \in \mathbf{V}$ is then computed as a sum of individual consumption b_m of edges $\mathbf{e}_m$ on the shortest path between these nodes.

Many models are capturing electric vehicle consumption, which represents one whole field of study. Some studies are focused on introducing new models aiming to predict future vehicle consumption concerning speed, acceleration, roadway grade, and other parameters [3], [29]. Even though there are many possibilities, I decided to include a straightforward and naive linear model working with a traveled distance and estimated maximal vehicle driving range:

$$\mathcal{E}[\mathbf{n}_k, \mathbf{n}_l] = b_a = (D[\mathbf{n}_k, \mathbf{n}_l]/D_{MAX}) \cdot 100$$

Here $\mathbf{n}_l \in \mathcal{N}[\mathbf{n}_k]$, b_a is energy consumption of ***Driving to a next location*** from a node $\mathbf{n}_k$ to a node $\mathbf{n}_l$. $D[\mathbf{n}_k, \mathbf{n}_l]$ returns a distance between nodes $\mathbf{n}_k$ and $\mathbf{n}_l$, here it corresponds with a length parameter l of an edge $\mathbf{e}$ connecting these two nodes. D_{MAX} is maximal driving range of an electric taxi. Its value is set in Table 8.1. 100 is used to switch to % of SOC.

5.4 $\mathcal{F}[s, \mathrm{n}]$ – Taxi Fare

I decided to introduce a simple taxi fares model generating the reward of a taxi driver. It consists of a pick-up fee f and then a standard rate r for a mileage.

$$\mathcal{F}[\mathcal{L}[s], \mathbf{n}] \;=\; D[\mathcal{L}[s], \mathbf{n}] \cdot r + f$$

Concrete values of f and r are used in evaluation Chapter 8.

Chapter 6

Solution Proposal

After performing a detailed definition of the charging recommendation problem as MDP and introducing all essential parameters in previous sections, it remains to present a solving method. Such a method receives a concrete implementation of a general environment defined in Section 4.1 which is closely connected with the Finite Discrete-Time Fully Observable Markov Decision Process introduced in Section 4.3. Another essential inputs are parameters estimated in Chapter 5. The recommending algorithm returns the policy π^* providing a taxi driver with step by step recommendations of actions $a \in \mathcal{A}$ maximizing his potential profit.

6.1 Environment Implementations

First of all, I propose three concrete implementations of the general environment representation defined in Section 4.1.

6.1.1 OSM Environment

First implementation of an environment employed also in [13] uses a graph $\mathbf{G}$ that originates from Open Street Map[1] (OSM) data[2]. This kind of an environment somehow copies a structure of the road network, presenting an ideal case to simulate real-world conditions. OSM data fulfill all requirements listed in Section 4.1 as it is possible to parse the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ together with edge parameters l, d, v essential for the algorithm described in the subsequent Section 6.2.

To improve a performance of the algorithm, I decided to add extra edges $\mathbf{e} \in \mathbf{E}$ connecting every single node $\mathbf{n} \in \mathbf{V}$ with every charging station $\mathbf{c} \in \mathbf{C}$. These connections are created by computing the shortest path between each environment node $\mathbf{n} \in \mathbf{V}$ and each charging station $\mathbf{c} \in \mathbf{C}$. Lengths l and times d are computed as a sum of these parameters on all edges forming the shortest path. Speed parameter v is computed as an arithmetical average of speeds weighted by distances of these edges.

[1] `https://www.openstreetmap.org`

[2] All OSM data were received from `https://www.openstreetmap.org/export`

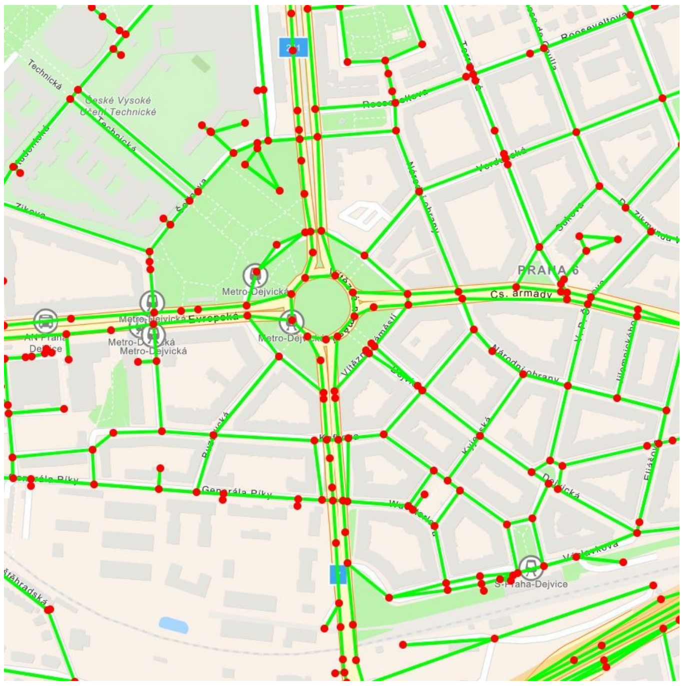


Figure 6.1: OSM environment graph example.

Unfortunately, planning on such an environment leads to enormous sizes of a state-space as a number of nodes and edges covering, for example, the whole city of Prague, is in tens of thousands. Also, time t to transfer edges $\mathbf{e} \in \mathbf{E}$ could be in seconds. These are the main reasons I decided not to use such an environment implementation as it would be impossible to evaluate a larger state space on my machine. Nevertheless, I use it extensively to define other environments properly.

6.1.2 Grid World Environment

A grid world environment is a classical simplification also used in [5]. More precisely, it partitions a map into rectangle cells. These cells represent environment nodes $\mathbf{n} \in \mathbf{V}$ in graph $\mathbf{G}$ defined in Section 4.1. Edges $\mathbf{e} \in \mathbf{E}$ are created between neighbouring cells in all eight directions as showed in Figure 6.2.

The width and height of one cell are essential parameters of the grid world environment. The smaller values of these parameters are the more detailed representation of the real-world road network originates. On the other hand, a higher resolution also leads to a higher number of environment nodes. To keep the state space bearable, I set both of these parameters experimentally to 2500 m, which preserves numbers of environment nodes in hundreds.

Then I partition a covered area into individual cells and fit every node from the OSM environment representation introduced in the previous subsection into its corresponding cell. I compute a center of each cell by averaging all longitudes and latitudes of OSM nodes in it.

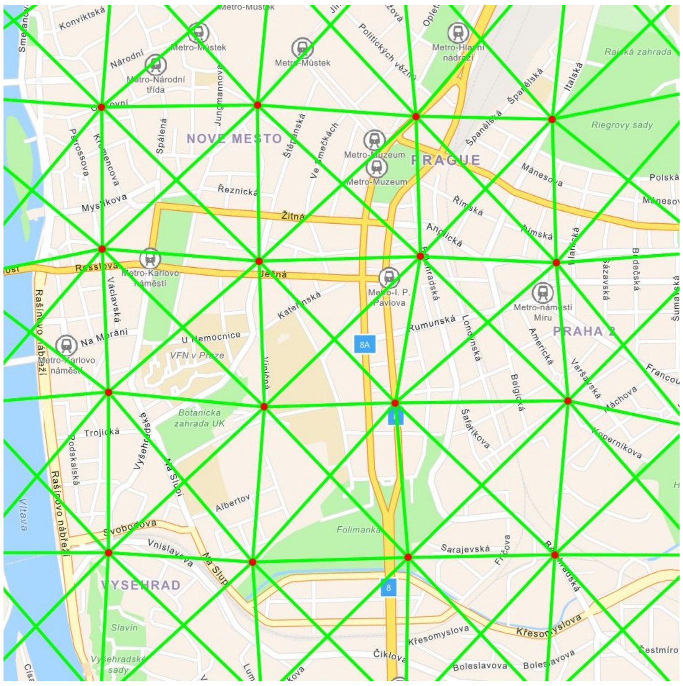


Figure 6.2: Grid World environment graph example.

Finally, I fit the computed center to the closest OSM environment graph node and compute the shortest paths between center points of neighboring cells again using the OSM environment graph. Computed paths represent edges between neighboring cells in all eight directions together with their lengths l, traverse times d, and speeds v. I calculate them in the same way as in the case of edges between nodes and charging stations in previous Subsection 6.1.1. Calculated cell center points are precise representations of all environment nodes $\mathbf{n} \in \mathbf{V}$. I also add edges connecting charging stations $\mathbf{c} \in \mathbf{C}$ with all these cell center points $\mathbf{n} \in \mathbf{V}$ received again by computing the shortest paths.

6.1.3 K-Means Clustering Environment

The last presented environment originates from a K-Means clustering of taxi trip pick-up points from historical taxi trip data sets described in Chapter 3. The K-Means algorithm produces clusters containing all given pick-up points together with finally placed centroids. These centroids represent the set of environment nodes $\mathbf{n} \in \mathbf{V}$ from general environment representation defined in Section 4.1.

Generating such an environment representation could be interpreted as an effort to partition points on a map to clusters corresponding to taxi driver's points of interest as the K-Means algorithm optimizes a variance of a distance between centroids and real data set pick-up points.

K-Means [24] is a well known and simple clustering algorithm used in a wide variety of fields such as computer vision, image segmentation, feature learning, and many others. It receives a set of data points $\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$, where L is a size of the data set. A result of the algorithm is a set of chosen centroids $\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^{K}$ and a partitioning $\{\mathcal{T}_k\}_{k=1}^{K}$ of original data points to clusters corresponding to individual centroids. K signifies a number of clusters and corresponds with the K in the name of the clustering method. The algorithm can be described in few simple steps as [11]:

1. Initializing centroids $\{\mathbf{c}_k\}_{k=1}^{K}$ randomly into taxi trip pick-up points $\{\mathbf{x}_l\}_{l=1}^{L}$. Eventually using K-means++ initialization [23] as in my case.

2. Assigning each pick-up point of the data set $\mathcal{T}$ to its closest centroid $\mathbf{c}_k$ and so creating the current cluster partitioning $\mathcal{T}_k$. Euclidean distance $d = ||\mathbf{x} - \mathbf{c}_k||^2$ is originally used to compare the distance between data set points and centroids. Nevertheless, I replaced it by using a Haversine formula[3] as discussed below.

3. Replacing all centroids $\{\mathbf{c}_k\}_{k=1}^{K}$ into a mean of data assigned to it i.e. $\mathbf{c}_k = \frac{1}{|\mathcal{T}_k|} \sum_{\mathbf{x} \in \mathcal{T}_k} \mathbf{x}$ or re-initialization of centroids in a case of an empty cluster $\mathbf{c}_k$.

4. The algorithm terminates in a case of no change in the cluster partitioning between two steps. Otherwise, going back to the step 2.

Following the general definition of the environment from the Section 4.1, the set of finally placed centroids $\{\mathbf{c}_k\}_{k=1}^{K}$ represent the set of environment nodes $\mathbf{n} \in \mathbf{V}$ as announced in the introduction of this subsection. Edges $\mathbf{e} \in \mathbf{E}$ leading from each node $\mathbf{n} \in \mathbf{V}$ are constructed for each neighbouring cell in a resulting Voronoi diagram [28]. A final environment structure is observable in Figure 6.3. Lengths l, times d, and speeds v of edges $\mathbf{e} \in E$ are retrieved by computing the shortest paths between OSM environment nodes corresponding to K-means centroids. It is done again by summing lengths, times, and computing a weighted average of speeds on individual edges of the shortest path. Finally, edges connecting centroids and charging stations are added.

There are several disadvantages of using the K-Means algorithm for spatial data to be discussed. First of all, it is designed to minimize the variance of a data distance to centroids, which leads to a decreased robustness to noise and outliers [25]. Furthermore, Geoff Boeing discourages using the K-Means algorithm to cluster spatial data represented by longitude and latitude in [2]. The reason is a significant distortion of longitude and latitude data far from the equator caused by the curved earth and again the variance optimization. To cope with such a shortcoming, I substituted a basic euclidean distance computation by involving a Haversine formula that considers the curved earth.

Despite some limitations, I decided to use the K-Means algorithm for its simplicity with the possibility of future development and incorporation of more suitable clustering algorithms such as a K-Medoids [25] or a DBSCAN used in [2].

[3] https://www.geeksforgeeks.org/program-distance-two-points-earth/

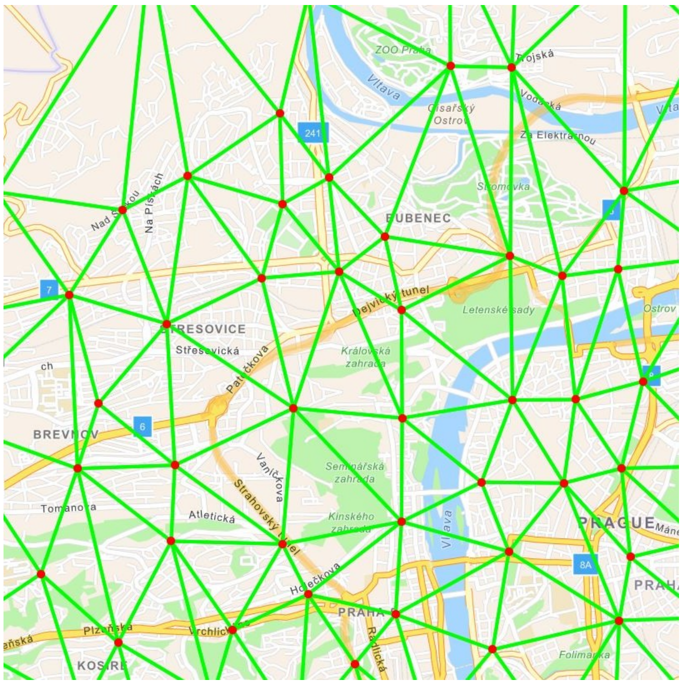


Figure 6.3: K-Means environment graph example.

6.2 Backward Induction Algorithm

Because I have defined the charging recommendation problem in Section 4.3 as a finite horizon problem [7], the set of timestamps $\mathcal{D}$ is limited. Thanks to this I can now introduce a simple and optimal form of a Value iteration algorithm also used in [5]. It is based on dynamic programming returning the desired policy π^*.

The principal idea is in starting computing the maximal reward from the end of a shift, i.e., from time $t_0 + t_{MAX}$ (line 5 in Algorithm 1) following backward and choosing actions $a \in \mathcal{A}$ (line 9) with the most significant potential reward (line 7) for all states $s \in \mathcal{S}$ till the beginning of a shift (states in time t_0). This method is also called backward induction [21].

A time complexity of described algorithm is linear in each dimension of states $s \in \mathcal{S}$ and number of feasible actions i.e. $O(|\mathbf{V}| \times |\mathbf{B}| \times |\mathcal{D}| \times |\mathcal{A}|)$. On the other hand, the space complexity is linear in each dimension of states $s \in \mathcal{S}$ leading to $O(|\mathbf{V}| \times |\mathbf{B}| \times \mathcal{D})$.

In both cases, the complexity is affected mainly by a number of environment nodes $|\mathbf{V}|$ as all the other components still stay the same and are in hundreds. Namely, a cardinality of possible SOC $|\mathbf{B}| = 100$, a number of possible time stamps is determined by a length of a taxi driver's shift. In the evaluation section I use 12 hours as the upper bound i.e. $|\mathcal{D}| = 60 \cdot 12 = 720$. There are 4 feasible actions. A cardinality of environment nodes set $|\mathbf{V}|$ can rise to hundreds of thousands in a case of the OSM environment (Section 6.1.1), presenting the main reason to introduce other types of environments with a lower number of environment nodes to keep the space complexity of the algorithm bearable.

Algorithm 1: Finite Horizon Value Iteration

```
   Input: $\mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{T}, \mathcal{R}, \mathcal{P}, \mathcal{P}^d, \mathcal{E}, \mathcal{F}, p, r, f$
   Output: $\pi^*$
1  $\pi^*[\mathcal{S}] \leftarrow NULL$;
2  $\mathcal{R}^*[\mathcal{S}] \leftarrow 0$;
3  for $t \leftarrow t_0 + t_{MAX}$ to 0 do
4      for $s_d \in \mathcal{S}$ with time $t$ do
5          for $s_s \in \mathcal{S}$ with $s_d$ reachable from it do
6              for $a$ feasible in $s_s$ to reach $s_d$ do
7                  if $\mathcal{R}^*[s_s] < \mathcal{R}[s_s, a, s_d]$ then
8                      $\mathcal{R}^*[s_s] \leftarrow \mathcal{R}[s_s, a, s_d]$;
9                      $\pi^*[s_s] \leftarrow a$;
10                 end
11             end
12         end
13     end
14 end
15 Return $\pi^*$;
```

6.3 Charging Recommendations

The whole idea of the proposed solution works with an assumption that data from historical taxi trip data sets and estimated parameters express the time and space context of a taxi demand. Generating a policy π^*, should not only lead a taxi driver through potentially profitable locations, but it should also fulfill the primary purpose of my thesis which is optimal planning of charging stops.

Such a policy proposes charging stops concerning multiple constraints such as suitable time of a charging stop, a desirable place with the most significant future expected profit, or a charging station connection type (Section 3.2) influencing the price and length of charging. Moreover, thanks to the generality of the above-declared problem definition, it is easy to introduce new factors affecting charging recommendations such as charging connection availability prediction or electricity price time variation. As the proposed solution takes into account all available parameters at once, it always decides to choose the one with the highest expected profit according to the given and estimated parameters.

Chapter 7
Implementation

In this section, I briefly describe an implementation of the previously defined charging recommendation problem, together with the solving algorithm. Everything is implemented in Java 11, and Maven manages dependencies.

The algorithm itself is then used by an agent in simulation introduced in subsequent Chapter 8. Generated policy π^* is a base model of an agent's behavior, ruling his step by step decisions. All evaluation results and graphs were received by short scripts written in MATLAB.

7.1 Domain Implementation

The most significant part of code is formed by so-called domain implementation, which corresponds to all sections in Chapters 4 and 5. It is all situated in a package `domain` in the root source directory of a project.

7.1.1 Environment

I will start with description of an `environment` package containing an implementation of all three environments introduced in Section 6.1. First of all it contains generic abstract classes of `Environment`, `EnvironmentGraph`, `EnvironmentEdge` and `EnvironmentNode` prescribing all necessary methods to be implemented by concrete environment classes.

As indicated in the description of individual environments in Section 6.1, each of them is closely connected with graph originating from OSM data. For this purpose I used `Graph`, `RoadNode` and `RoadEdge` implementations defined in `jones.felk.cvut` repository[1] in packages `basestrucures` and `multimodal-strucures`. They contain all necessary features for my environment definition, together with tools to serialize the final graph object to `.fst`[2] file.

As discussed in Subsection 6.1.1, the OSM environment copies the structure of original `Graph` parsed from OSM data. Contrarily, `GridWorldEnvironment` and `KMeansEnvironment` are responsible for its correct creation based on received `Graph` and static parameters set in class `Utils`.

[1] http://jones.felk.cvut.cz/artifactory/repo/

[2] https://github.com/RuedigerMoeller/fast-serialization

All `Environment` classes create a wrapper for an environment representation implemented in `EnvironmentGraph` classes. As soon as an instance of an `Environment` is created, an abstract method to set properties is called.

Grid World Environment

Besides the OSM `Graph`, the Grid World Environment is parameterized by the height and width of the one cell in kilometers discussed in Subsection 6.1.2. Exact values are also specified in evaluation Chapter 8. Per these values, numbers of rows and columns are computed, individual cells are connected with their bounding longitudes and latitudes, and OSM `RoadNode`s are fitted into corresponding cells. After such a procedure all `GridWorldEnvironmentNode` and `GridWorldEnvironmentEdge` instances are created in means of Subsection 6.1.2.

K-Means Environment

The core of K-Means Environment implementation is in the K-Means algorithm itself. Thank to its simplicity, I decided to use a modified implementation published in [3] together with the K-Means++ initialization inspired by a Python implementation in [4]. In contrast with the Grid World Environment, the size of the K-Means Environment is parameterized by a single integer K. This means a number of K-Means clusters that are created during the algorithm. Its value is again specified in the evaluation Section 8.

After the clustering process finishes, `KMeansEnvironmentNode` instances are created based on result clusters and connected by `KMeansEnvironmentEdge` instances as Voronoi diagram borders computed by an algorithm placed in [5].

7.1.2 MDP Representation

A `TaxiRecommenderDomain` is a main class responsible for the entire domain initialization placed in a root of the `domain` package. After its creation, it loads and initializes all essential data from the OSM `Graph`, a historical taxi trip data set, to estimated parameters, a charging station data set, or transitions between individual environment nodes. All the other nested packages contain an implementation of individual domain's parts.

Actions and States

Both of them are placed in identically named packages and correspond with structures defined in Section 4.3. The most essential feature to point out is an abstract class `TaxiActionType` prescribing an abstract method `createConnections` to be implemented by all action types $a \in \mathcal{A}$ to create connections between a given state s and all the other states reachable by some

[3] https://www.baeldung.com/java-k-means-clustering-algorithm
[4] https://www.geeksforgeeks.org/ml-k-means-algorithm/
[5] https://github.com/aschlosser/voronoi-java

action. States $s \in \mathcal{S}$ are implemented in a class `TaxiState` corresponding with its formal definition in Section 4.3. Discussing the concept of reachability, there are introduced several conditions such as *Not running out of battery* or *Not operating after the end of a shift* defined in a class `ActionUtils`. These conditions are used in individual implementations of a `TaxiActionType` to monitor reachability together with a non zero value of the transition function $\mathcal{T}[s_i, a, s_j]$ between two states set in the `TaxiRecommenderDomain` class.

Parameter Estimation

Parameter estimation is delegated by a class `ParameterEstimator` consisting of three main parts. Namely a `PassengerPickUpEstimator` computing the pick-up probability $\mathcal{P}[s_i, a, s_j]$, a `PassengerDestinationEstimator` taking responsibility for estimates of the passenger drop-off probability $\mathcal{P}^d[s, \mathbf{n}]$, and an `EnergyConsumptionEstimator` providing static methods estimating energy consumption according to a distance of a given path. All the other parameters are defined either in a `Utils` class as in a case of the taxi fare $\mathcal{F}[s, \mathbf{n}]$ or in a `charging` package in a `ChargingRateType` enum. There is also an interface `DataSetReader` to be implemented for simple adding of any other historical taxi trip data set.

Charging

A package named the same (`charging`) contains all necessary methods to parse charging stations data sets as introduced in Section 3.2. Parsing JSON files is done by `JSONParser` from [6].

Data Serialization

Due to a usage of several time-consuming operations such as the K-Means clustering or the shortest paths computing, I use `fast-serialization` wherever needed. It serializes objects into `data/programdata` directory placed in a root directory of the project.

7.2 Solution Implementation

Thanks to the simplicity of the solution proposed in Section 6, an implementation is captured in class `ChragingRecommender` placed in `problemsolving` package. It is also fair to mention a `TaxiRewardFunction` class implementing the reward function $\mathcal{R}[s_i, a, s_j]$ as it plays an essential role in the Algorithm 1.

The final solution consists of three simple steps:

1. Generation of all reachable states.

[6] `https://github.com/fangyidong/json-simple`

2. Creating all connections between individual reachable states (reachability by taking action $a \in \mathcal{A}$).
3. Computing the policy π^* by the proposed Algorithm 1

After performing this procedure, parameters of the maximally profitable action and the expected reward in each reachable state $s \in \mathcal{S}$ are already set. In other words, the policy π^* is computed.

7.3 Simulation Implementation

For the purpose of the evaluation presented in subsequent Chapter 8, I implemented a simple `Simulation` of a taxi driver's shift in a package `evaluation`. The whole simulation works with the OSM `Graph` as a representation of the real-world road network. This fact helps to receive results corresponding as much as possible with the reality. Still, it also means that using some kind of an environment as defined in Section 4.1 brings the requirement to fit nodes from OSM `Graph` to `EnvironmentNode` $\mathbf{n} \in V$. Fitting is done by an assignment of each OSM node from the `Graph` to the closest `EnvironmentNode`.

The simulation is prepared for two kinds of agents (based on my charging recommending algorithm and base model agent) more specified in Section 8.1.1.

On the very beginning, a short initialization based on static parameters set in a `Utils` class is necessary. It contains loading of all needed data, initialization of both agents (policy π^* computation), and choosing agent to start. A starting position is always chosen randomly from all available nodes in the OSM `Graph`. A starting time stamp and a SOC are defined in the mentioned `Utils` class. These three properties together create a `SimulationState`. After such a process, it is possible to start the simulation with minute steps going through an entire shift.

7.3.1 Actions

Both agents are allowed to take actions as defined in Section 4.3. The only difference is that in the case of ***Driving to a next location***, it is not necessary to travel into a neighboring node in the `Graph`, but an agent is free to choose from all available nodes. After such a choice, action a properties like duration t_a, consumption b_a or a transferred distance are computed in means of Chapter 5. In each simulation step, the current agent's position is computed from them. After finishing each action, an agent is asked to choose another one.

7.3.2 Taxi Trip Offers

During the whole simulation, an agent receives offers from customers and decides whether to do or do not accept them. I assume that a taxi driver is not to change his mind after deciding on taking actions of ***Going to a charging station*** and ***Charging at a charging station*** as also announced in the definition of the transition function $\mathcal{T}$ in Section 4.3. Therefore taxi trip offers are generated exclusively in a case of taking actions:

1. ***Going to a next location***.
2. ***Staying in a location***

These trips are chosen randomly from the historical taxi trip data set concerning the current estimation interval, as presented in Chapter 5.

Of course, this way of trip generation is a considerable simplification and a way to avoid implementing a complex simulation model. On the other hand, it still preserves assumptions for time and space varying taxi demand used in my definition of charging recommending problems as they originate from real data, and a start time of each trip is considered.

Chapter 8
Evaluation

In this chapter, I present a comparison of two taxi driver agents. A behavior of the first one is based on the proposed recommending algorithm. The second one serves as a base model behavior presented in Subsection 8.1.2.

There are several configurations with different input parameters compared. These parameters are different taxi trip data set – Prague, New York, and a different environment representation of the charging recommender agent – `GridWorldEnvironment`, `KMeansEnvironment`. I have also compared a performance of both agents under a different starting SOC.

All data needed for comparisons were received by simulating 1000 taxi driver's shifts for all kinds of configurations and saving parameters introduced in Section 8.2 for each shift. As my thesis's primary goal was to propose an algorithm that recommends charging stops for a taxi driver concerning his profit, *rewardPerShift* parameter appears to be the primary choice to focus. Nevertheless, it is closely connected with many others, and it needs to be interpreted together with them.

Because of the fact, that comparison is made by using means of received data. I decided to discard 5 % of the best and the worst simulated shifts in ways of taxi driver's reward, to avoid their harmful effect on a result mean. Significant outliers were produced mainly by the base model agent introduced in Section 8.1.2 so omitting them does not considerably affect results of the charging recommender based agent.

All comparisons were made on a machine with four cores of Intel Core i7-8565U CPU @ 1.80-4.60GHz and 16 GB of RAM. Running time of one experiment was around 60 minutes consuming approximately 13 GB of RAM. The calculation of the policy π^* itself took approximately 5 minutes.

Only the most critical figures capturing evaluation results are listed in subsequent sections. The complete overview is to be found in Appendix B.

8.1 Taxi Driver Models

8.1.1 Charging Recommender Agent (CRA)

As announced above, the first of two examined taxi driver agents makes his step by step decisions based on the policy π^* produced by the algorithm described in Section 6. Each decision is made by fitting the current `SimulationState` into agents `TaxiState` and performing an action corresponding to the policy π^*.

Moreover, this agent decides whether to do or do not accept an offer to trip comparing expected rewards in the current state with a reward in a potential ending state of an offered trip.

8.1.2 Base Model Agent (BMA)

The behavior of the BMA to be compared with the CRA can be described in several points:

- Chooses "the most lucrative place" in means of a number of historical passenger pick-ups and always returns back to this place while waiting for a passenger request.
- Accepts all passenger requests. The only restriction is a SOC, which is maintained above defined level.
- Taking ***Going to a charging station action*** after a SOC drop under a defined level.
- Choosing always the closest charging station.
- Choosing the fastest charging connection in a chosen charging station.
- Charging to full SOC.

The low level of SOC is set individually for each data set. I decided to use such a model as it shows base signs of rational human behavior such as the selection of a "favorite" place, for example, near an airport and keeps moving around it.

8.2 Metrics

I decided to observe several metrics that are recorded and saved in class `SimulationStatistics` for each simulated taxi driver's shift. They are used to model and evaluate the performance of individual agents.

- ***rewardPerShift*** – A sum of all trip fares together with payments for charging during a shift.
- ***distanceTransferred*** – A total sum of distances transferred during a shift.
- ***numOfTripsDone*** – A number of trips done during a shift.
- ***rewardFromTrips*** – A sum of all trip fares during a shift.
- ***costOfCharging*** – A sum of payments for charging during a shift.

- ***totalEnergyCharged*** – An amount of energy charged in % during a shift.
- ***totalEnergyConsumed*** – An amount of energy consumed in % during a shift.
- ***distanceToReachPassenger*** – A sum of distances transferred by reaching passengers during a shift.
- ***distanceWithPassenger*** – A sum of distances transferred when travelling with passengers during a shift.
- ***timeSpentCharging*** – A total amount of time spent charging during a shift.

After receiving such parameters from all simulated shifts I process them and generate results in the form of graphs in MATLAB.

8.3 Static Parameters

As the problem of electric taxi shift simulating is quite complex, there are several static parameters needed to be set. All of them are presented in Table 8.1 as rough approximations of real-world data.

SHIFT_START_TIME	8 AM
SHIFT_LENGTH	12 hours
TAXI_FARE_FOR_KM	30 Kč
TAXI_START_JOURNEY_FEE	30 Kč
ELECTRIC_VEHICLE_DRIVING_RANGE	300 Km
BATTERY_CAPACITY	40 KW
ESTIMATION_INTERVAL	30 min

Table 8.1: Simulation static parameters.

According to charging station data introduced in Section 3.2, I also decided to present three different types of charging and three corresponding charging rates p based on a kind of a charring connection.

- ***Speed charging*** – 60+ KW with rate $p = 15$ Kč/min
- ***Standard charging*** – 20–60 KW with rate $p = 10$ Kč/min
- ***Slow charging*** – 0–20 KW with rate $p = 5$ Kč/min

Undoubtedly this is a simplification of the real-world situation. It is introduced mainly because of inconsistencies in charging station data and missing information about the real cost in a large number of cases.

8.4 Prague Data Set Evaluation

For the purpose of Prague taxi trip data set evaluation, I used the OSM `Graph` containing 54 821 nodes and 115 833 edges. They cover an area of approximately 30 × 22 Km. Both agents could choose from 75 charging stations with all types of charging connections, as presented in Section 8.3. The low level of BMA's SOC was set to 15 % as an examined area is quite large, and distances between remote locations and the closest charging stations are long. A taxi trip offer is generated in each minute simulation step with the probability 0.05.

8.4.1 BMA vs. CRA (Grid World)

For a comparison of BMA and CRA using the Grid World representation of the environment, I set the height and width of one cell to 2500 m. Such values produce the Grid World with 12 columns and 9 rows leading to 108 `GridWorldEnvironmentNode`s.

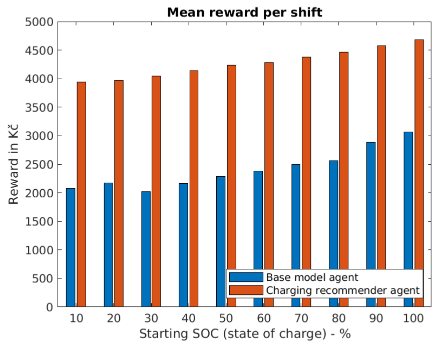
Reward

As shown in Figure 8.1a, CRA adopting a behavior lead by the policy π^* generated by the above-proposed algorithm overwhelmingly outperforms BMA in means of a total received reward per one shift. It is essential to state that real values of rewards can significantly differ in real-world situations due to approximate parameter values used in the simulation. Nevertheless, both agents were served with the same conditions, so pointing out the difference between them is a fair statement.

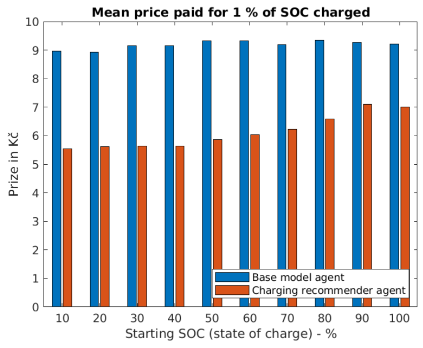
According to comparisons of remaining parameters, there are several indicators of such a result. First of all, as observable in Figure 8.1b, the price paid for charging of 1 % of SOC by CRA is significantly lower than the one paid by BMA. This can be interpreted as CRA successfully chooses stations and mainly charging connections with a more favorable ratio between a price and an amount of charged energy.

Moreover, there are several other connected indications of such an outperformance in the reward criteria. First of all, BMA transfers considerably more kilometers per shift that naturally leads to higher energy consumption, a higher amount of energy charged, and the most fundamentally, a longer time spent by charging. Due to these results, a mean number of trips per shift of BMA is about 15 % lower, making approximately 2-3 trips per shift.

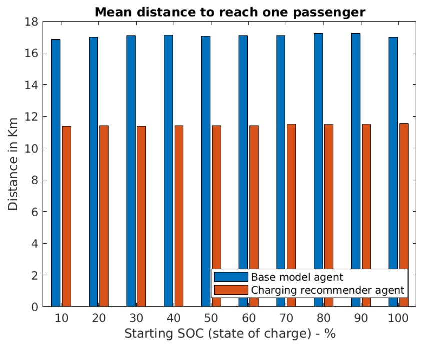
Two main factors cause the longer transferred distance of BMA. Probably the more significant one is BMA's returning to his "most lucrative place" that presents a lot of traveled kilometers. On the other hand, the second factor could be presented as a benefit of CRA, which successfully minimizes a distance to his next passenger making around 6 Km, as shown in Figure 8.1c.



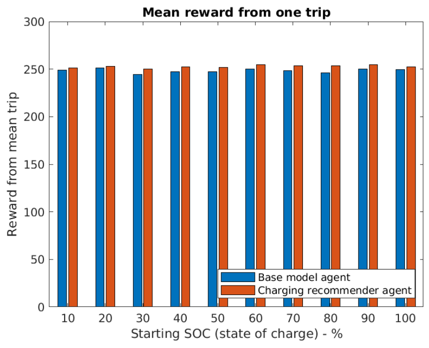
(a) : Mean reward per shift.



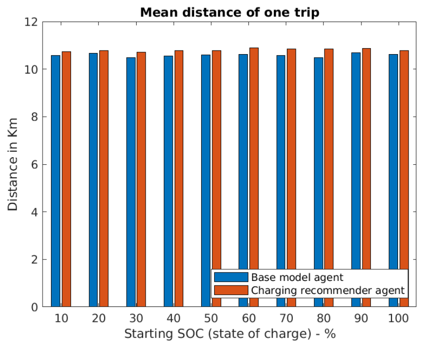
(b) : Mean Price paid for 1 % of SOC charged.



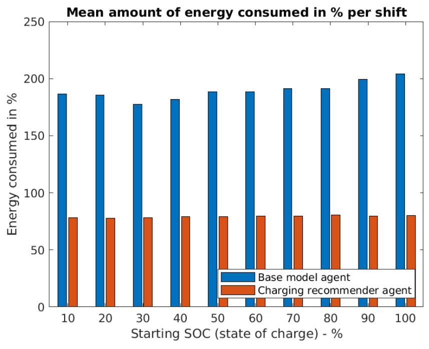
(c) : Mean distance to reach one passenger.



(d) : Mean reward from one trip.



(e) : Mean distance of one trip.



(f) : Mean amount of energy consumed in % per shift.

Figure 8.1: Prague data set evaluation results – BMA vs. CRA.

There are also very slight differences in Figures 8.1d and 8.1e indicating, that CRA is able to slightly overcome BMA in "lucrative trip" choosing.

Different Starting SOC

As shown in corresponding Figures 8.1a and 8.1f, there is an explicit dependency between a starting SOC and a resulting reward per shift. This is caused mainly by the higher need to charge an electric vehicle and so higher expenses.

8.4.2 Grid World Environment vs. K-Means Environment

As the behavior of the BMA agent is independent of the environment representation of CRA, I decided to omit a comparison of BMA and CRA with K-Means environment representation. In contrast to that, I highlight the differences between two CRAs with different environment representations to bring a more informative illustration.

It is essential to point out that there are no much differences in the results of these agents, as in the previous case. As can be seen in Figure 8.2b the K-Means based agent proves his original purpose and outperforms the Grid World environment in the mean distance to reach a passenger. Nevertheless, the difference is only in units of meters, and as indicates Figure 8.2a the Grid world agent outperforms the K-Means variant in the most critical performance indicator – reward. These slight deviations can be caused by the fact that an effort to model the environment concerning a distance to passenger pick-up points slightly reduces an agent's ability to choose a favorable charging station, as shown in Figure 8.2c.

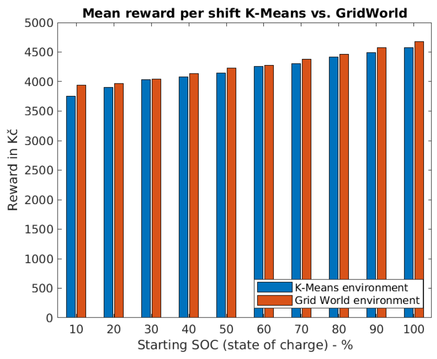
Other figures also indicate that K-Means environment based agent tends to transfer slightly more distance in some cases, and so needs to spend more time charging. This fact naturally leads to higher expenses.

8.5 New York Historical Taxi Trip Data Set Evaluation

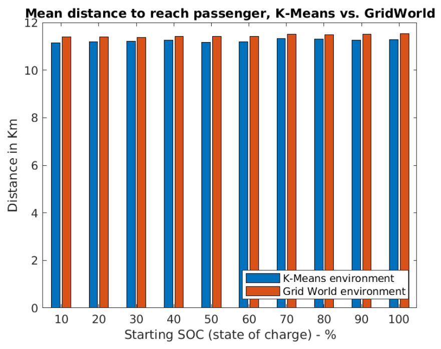
In case of the New York historical taxi trip data set evaluation, used OSM `Graph` contains 75 701 nodes and 173 433 edges covering an area of 20×35 Km. Contrarily to the Prague data set, the low level of BMA's SOC was set to 10 % as agents tend to move around a smaller area of Manhattan and Brooklyn, as discussed in Chapter 3. The taxi trip offer probability is set to 0.10.

8.5.1 BMA vs. CRA (Grid World)

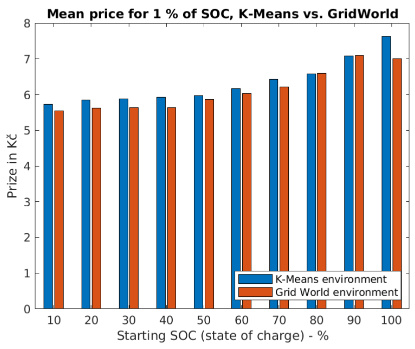
The same as in the case of the Prague data set, the height and width of a one cell in the Grid World environment representation were set to 2500 m. Here it turned out into the Grid World with 8 columns and 14 rows leading to 112 `GridWorldEnvironmentNode`s.



(a) : Mean reward per shift.



(b) : Mean distance to reach passenger.



(c) : Mean Price paid for 1 % of SOC charged.

Figure 8.2: Prague data set evaluation results – K-Means vs. Grid World.

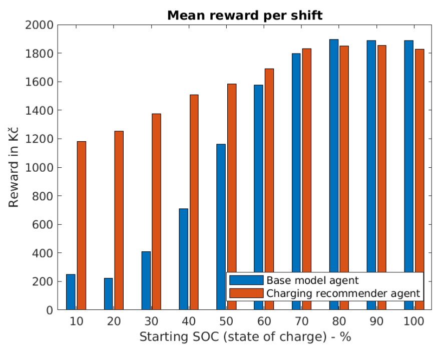
Reward

As already indicated in the analysis of the New York taxi trip data set in Section 3.1, results of a comparison of BMA and CRA slightly differ from those received in the previous section. The main argument of such a difference is a substantially shorter distance transferred per a shift by both agents showed in Figure 8.3c. The distance is around 200 – 250 Km per a shift, which is less than half of a range reached in the Prague's case. The same observation is received from Figure 8.4a, showing the mean distance of a one trip or Figure 8.3d with the mean distance to reach a passenger.

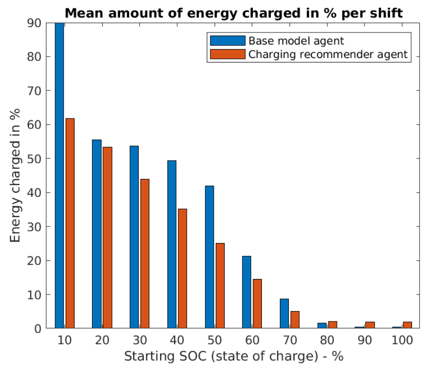
All of these features were triggered by a strong taxi demand within a relatively small area of Manhattan and Brooklyn compared to the Prague.

Such a variance leads to several consequences that appeared in results of the evaluation. The limited distance to transfer naturally reduces a need to charge electric taxi during a shift in case of a higher starting SOC. This fact causes a trend seen, for example, in Figures 8.3a, 8.4b or 8.4c, where CRA keeps his performance almost independent on a starting SOC, whereas BMA suffers a lot from a need to go charging brought by a lower starting SOC.

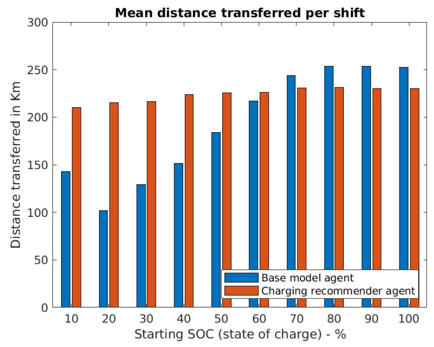
As a starting SOC rises, BMA benefits from an absence of charging actions and balances it's performance to CRA. This fact again proves CRAs ability



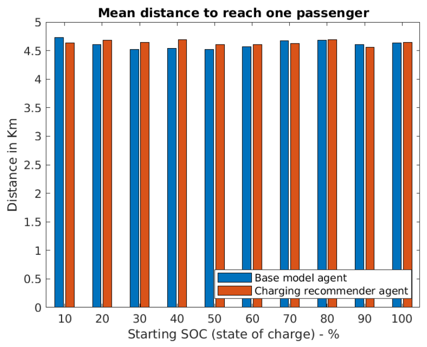
(a) : Mean reward per shift.



(b) : Mean amount of energy charged in % per shift.



(c) : Mean distance transferred per shift.



(d) : Mean distance to reach one passenger.

Figure 8.3: New York data set evaluation results – BMA vs. CRA – part one.

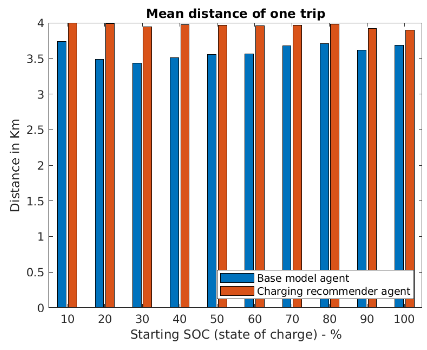
to choose a right time, place, and length for a charging action, which brings him a huge benefit. On the other hand, the structure of the New York data set, in principle, eliminates a CRA's ability to minimize a distance to reach passengers as it is enough to stay in places with the highest demand density. This is perfectly observable in Figure 8.3d.

Focusing exclusively on results received after higher starting SOC, BMA tends to outperform CRA in means of reward slightly. This fact is caused mainly by CRA's light tendency to keep his SOC higher and to go to the charging station even if it is not necessary, as indicated in Figure 8.3b.

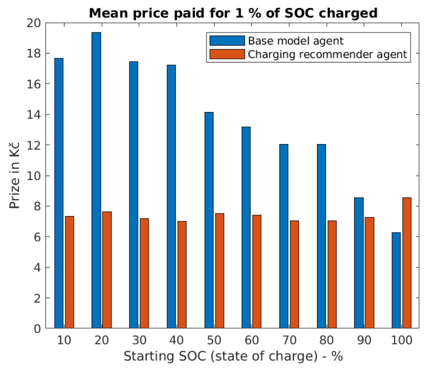
8.5.2 Grid World Environment vs. K-Means Environment

The comparison of individual CRA's environment representations does not report such a considerable difference from the Prague taxi trip data set even though there are some subtle nuances.

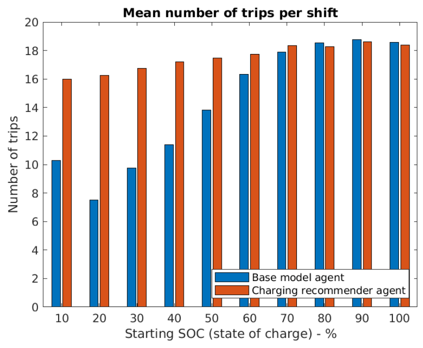
First of all, the K-Means environment based agent loses his only one strong point, which was the mean distance to reach the passenger's pick-up location. Figure 8.5b shows that Grid World environment based agent outperforms the K-Means variant by about 500 meters, which represents



(a) : Mean distance of one trip.



(b) : Mean price paid for 1 % of SOC charged.

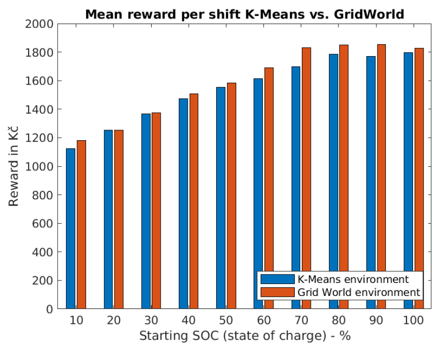


(c) : Mean number of trips per shift.

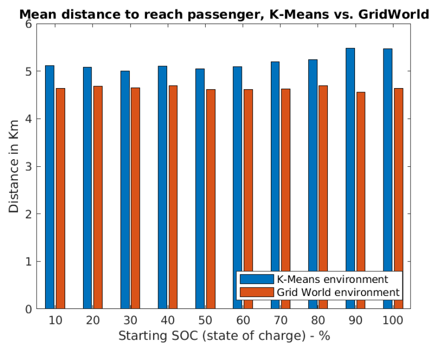
Figure 8.4: New York data set evaluation results – BMA vs. CRA – part two.

quite a difference in connection with generally short distances transferred in the New York trip data set.

Even though the K-Means based agent tends to transfer more distance, consume more energy and spent more time charging, it does not turn into the final reward considerably as the time lost charging moves in tens of minutes.



(a) : Mean reward per shift.



(b) : Mean distance to reach passenger.

Figure 8.5: New York data set evaluation results – Grid World vs. K-Means.

Chapter 9
Conclusion

My bachelor's thesis's essential objective was to formally define the problem of charging recommendations for electric taxis and propose a suitable solving algorithm concerning a taxi driver's profit. An indispensable component of such a target was implementing the presented algorithm and evaluation of real-world data sets, proving its performance.

For the very beginning, I researched the state of the art in a taxi movement recommending field identifying the MDP framework as one of frequently employed tools to model such a problem. After a short introduction of historical taxi trip data sets used to mine information regarding the taxi demand, I introduced a detailed problem representation focusing on a generality to facilitate future extensions and improvements followed by an estimation of parameters essential for the subsequently proposed solving algorithm. Finally, implementation and evaluation chapters verified theoretical assumptions made during the solution design by comparing capabilities of agents lead by the generated policy and agent adopting the base model of a taxi driver's behavior.

Results of all executed experiments proved an ability of the proposed recommending algorithm to determine a convenient place and time for charging actions with an amount of charged energy. Such a capability was identified as the critical factor of an outperformance of the base model agent concerning the overall profit. The evaluation also proved my solution's ability to minimize a distance to reach the next passenger under certain conditions for a shape of the taxi trip data set and so taxi demand in the concrete city. An additional outcome of the evaluation was the indication of unfulfilling the K-Means environment potential. Such a form of environment representation was able to achieve minor improvement in the minimization of a distance to reach passengers. However, it was observable only under the condition of suitable taxi trip data set structure and despite that K-Means based agent lost in comparison of terminal profits.

From my point of view, the space complexity is the most significant limitation of the proposed solution. It manifests the most in a need to limit the number of environment nodes to keep the computation bearable, which decreases the accuracy of recommendations. Nevertheless, as the calculation is only to be performed after each historical data sets actualization, the resulting recommending application could be easily implemented as a policy database responding to individual queries.

On the other hand, I consider the generality of the proposed solution as its strongest point. It is designed to facilitate an implementation of additional features modeling the real-world situation to achieve better results. Such an improvement could be accomplished by including historical data concerning an availability of charging stations, developing information about charging rates in individual charging stations, attaching historical data of traffic in a city, or implementing other environment representations. All these improvements could produce a complex recommending system taking into account multiple constraints balancing them to maximize the profit.

Bibliography

[1] L. Bin, Z. Daqing, S. Lin, C. Chao, L. Shijian, Q. Guande, and Y. Qiang. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. *2011 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2011*, pages 63–68, 2011. doi:10.1109/PERCOMW.2011.5766967.

[2] G. Boeing. Clustering to Reduce Spatial Data Set Size. *SSRN Electronic Journal*, pages 1–7, 2018. arXiv:1803.08101, doi:10.2139/ssrn.3145515.

[3] F. Chiara, A. Kyoungho, and R. Hesham. Power-based electric vehicle energy consumption model: Model development and validation. *Applied Energy*, 168:257–268, 04 2016. doi:10.1016/j.apenergy.2016.01.097.

[4] GOV.UK. London electric vehicle infrastructure delivery plan, 2019. URL: http://lruc.content.tfl.gov.uk/london-electric-vehicle-infrastructure-taskforce-delivery-plan.pdf.

[5] R. Huigui, Z. Xun, Y. Chang, S. Zubair, and L. Alex. The rich and the poor: A Markov decision process approach to optimizing taxi driver revenue efficiency. *International Conference on Information and Knowledge Management, Proceedings*, 24-28-Octo:2329–2334, 2016. doi:10.1145/2983323.2983689.

[6] IEA. Global ev outlook 2019. Technical report, IEA, 2019. URL: https://www.iea.org/reports/global-ev-outlook-2019.

[7] M. Kolobov, A. Kolobov, and M. Mausam. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6:3–14, 06 2012. doi:10.2200/S00426ED1V01Y201206AIM017.

[8] P. Li, S. Bhulai, and J.T. van Essen. Optimization of the revenue of the new york city taxi service using markov decision processes. In S. Bhulai and D. Kardaras, editors, *6th International Conference on Data Analytics, Barcelona (Spain), November 12-16*, pages 47–52. IARIA, 2017.

[9] Liftago. Prague historical taxi trip data set, 2018. URL: `https://www.liftago.cz/`.

[10] S. Lihao, Y. Jinfeng, and Y. Zaiyue. Optimal charging strategy of Plug-in Electric taxi. *IEEE International Conference on Control and Automation, ICCA*, pages 1532–1537, 2013. `doi:10.1109/ICCA.2013.6565122`.

[11] J. Matas and O. Drbohlav. K-means, 2015. URL: `https://cw.fel.cvut.cz/b191/_media/courses/b4b33rpz/pr_10_k_means_2015_12_04.pdf`.

[12] Q. Meng, Z. Hengshu, L. Junming, L. Guannan, and X. Hui. A cost-effective recommender system for taxi drivers. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2014. `doi:10.1145/2623330.2623668`.

[13] T. Chien Ming, C. Sid Chi Kin, and L. Xue. Improving Viability of Electric Taxis by Taxi Service Strategy Optimization: A Big Data Study of New York City. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):817–829, 2019. `arXiv:arXiv:1709.08463v2`, `doi:10.1109/TITS.2018.2839265`.

[14] Government of Singapore. Taxi info 2016, 2016. URL: `https://www.lta.gov.sg/content/dam/ltagov/who_we_are/statistics_and_publications/statistics/pdf/taxi_info_2016.pdf`.

[15] N. SÖNNICHSEN. N. carbon dioxide (co2) emissions of taxis and taxi-like services in the united kingdom (uk) from 1990 to 2017. Technical report, Statista, 2019. URL: `https://www.statista.com/statistics/486067/co2-emission-of-taxi-services-uk/`.

[16] Taxi and Limousine Commission (TLC). 2018 yellow taxi trip data, 2018. URL: `https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq`.

[17] the Government of Canada. Zero emission vehicle infrastructure program, 2020. URL: `https://www.nrcan.gc.ca/energy-efficiency/energy-efficiency-transportation/zero-emission-vehicle-infrastructure-program/21876`.

[18] P. TROTTENBERG. New york city mobility report. Report, NYC Department of Transportation, 2018. URL: `https://www1.nyc.gov/html/dot/downloads/pdf/mobility-report-2018-print.pdf`.

[19] R. TSANG, P. BOUTOT, and D. CAI. China's mobility industry picks up speed. Report, Bain & Company, 2018. URL: `https://www.bain.com/insights/chinas-mobility-industry-picks-up-speed/`.

[20] VerticalScope. Top 10 best home ev chargers, 2019. URL: `https://www.autoguide.com/top-10-best-home-ev-chargers`.

[21] Wikipedia contributors. Backward induction — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/Backward_induction`.

[22] Wikipedia contributors. Exhaust gas — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/Exhaust_gas`.

[23] Wikipedia contributors. k-means++ — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/K-means%2B%2B`.

[24] Wikipedia contributors. k-means clustering — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/K-means_clustering`.

[25] Wikipedia contributors. k-medoids — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/K-medoids`.

[26] Wikipedia contributors. List of electric cars currently available — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/List_of_electric_cars_currently_available`.

[27] Wikipedia contributors. Ride hailing services — Wikivoyage, the free encyclopedia, 2020. URL: `https://en.wikivoyage.org/wiki/Ride_hailing_services`.

[28] Wikipedia contributors. Voronoi diagram — Wikipedia, the free encyclopedia, 2020. URL: `https://en.wikipedia.org/wiki/Voronoi_diagram`.

[29] X. Wu, D. Freese, A. Cabrera, and W. A. Kitch. Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, 34:52 – 67, 2015. URL: `http://www.sciencedirect.com/science/article/pii/S1361920914001485`, `doi:https://doi.org/10.1016/j.trd.2014.10.007`.

[30] Science X. Giving up gas: China's shenzhen switches to electric taxis, 2019. URL: `https://phys.org/news/2019-01-gas-china-shenzhen-electric-taxis.html`.

[31] J. Yinghao, C. Huimiao, L. Jiaoyang, H. Fang, L. Meng, H. Zechun, and S. Zuo Jun Max. Planning for electric taxi charging system from the perspective of transport energy supply chain: A data-driven approach in Beijing. In *2017 IEEE Transportation Electrification Conference and Expo, Asia-Pacific, ITEC Asia-Pacific 2017*. Institute of Electrical and Electronics Engineers Inc., oct 2017. `doi:10.1109/ITEC-AP.2017.8080844`.

[32] T. Zhiyong, J. Taeho, W. Yi, Z. Fan, T. Lai, X. Chengzhong, T. Chen, and L. Xiang Yang. Real-Time Charging Station Recommendation

System for Electric-Vehicle Taxis. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3098–3109, 2016. `doi:10.1109/TITS.2016.2539201`.

Appendix A

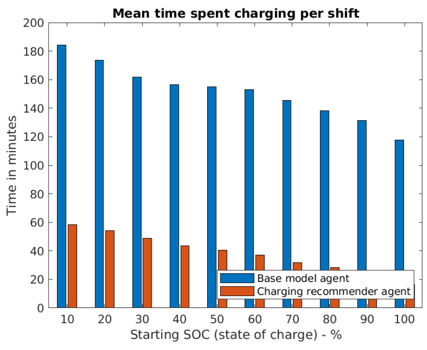
Charging Station Data Set Example.

```
{
   "IsRecentlyVerified":true,
   "DateLastVerified":"2020-03-02T06:53:00Z",
   "ID":149645,
   "UUID":"C5526F7B-1C39-4E7B-A7F1-F083D7C0966C",
   "DataProviderID":1,
   "OperatorID":23,
   "OperatorsReference":"53077",
   "UsageTypeID":4,
   "AddressInfo":{
     "ID":149998,
     "Title":"Hotel Pyramida",
     "AddressLine1":"Slatina 91 Frantiskovy Lazne",
     "Town":"Frantiskovy Lazne",
     "Postcode":"351 01",
     "CountryID":64,
     "Latitude":50.109416,
     "Longitude":12.360282,
     "ContactTelephone1":"+420 605 440 565",
     "DistanceUnit":0
   },
   "Connections":[
     {
      "ID":207763,
      "ConnectionTypeID":30,
      "StatusTypeID":50,
      "LevelID":2,
      "PowerKW":11.0,
      "CurrentTypeID":20,
      "Quantity":2
     }
   ],
   "NumberOfPoints":2,
   "GeneralComments":"2 Tesla Connectors, up to 11kW.Available
    for customers. Please call ahead.",
   "StatusTypeID":50,
   "DateLastStatusUpdate":"2020-03-02T06:53:00Z",
   "DataQualityLevel":1,
   "DateCreated":"2020-03-02T06:53:00Z",
   "SubmissionStatusTypeID":200
  }
```

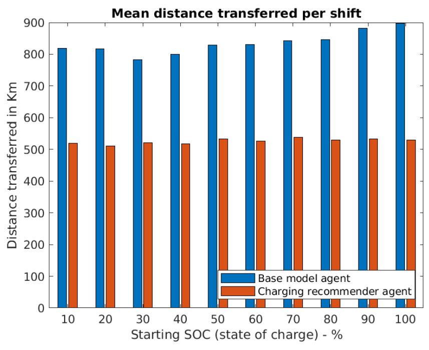
Appendix B
Evaluation Results

B.1 Prague Data Set Evaluation

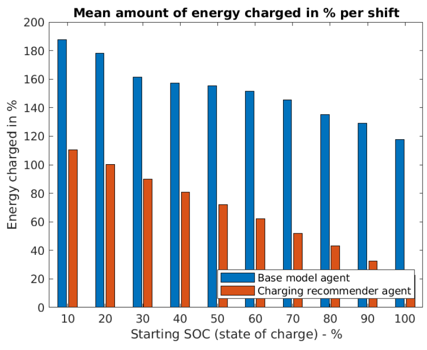
B.1.1 BMA vs. CRA (Grid World)



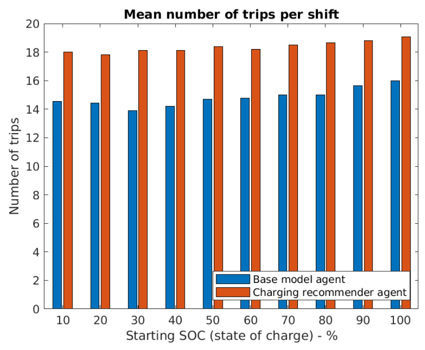
(a) : Mean time spent charging per shift.



(b) : Mean distance transferred per shift.



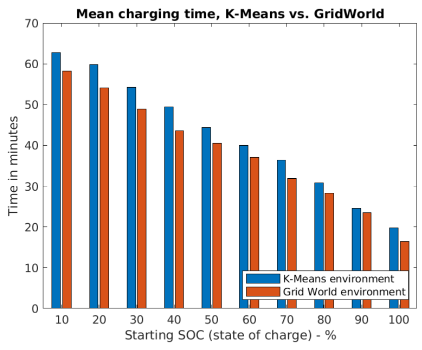
(c) : Mean amount of energy charged in % per shift.



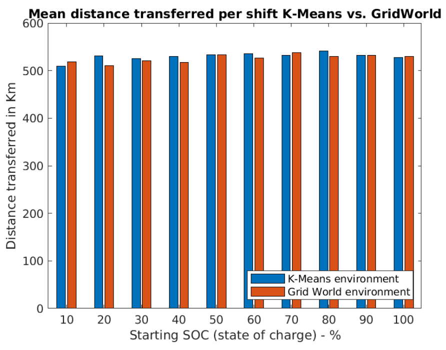
(d) : Mean number of trips per shift.

Figure B.1: Prague data set evaluation results – BMA vs. CRA part two.

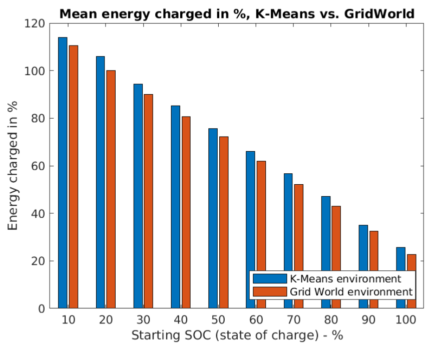
B.1.2 Grid World Environment vs. K-Means Environment.



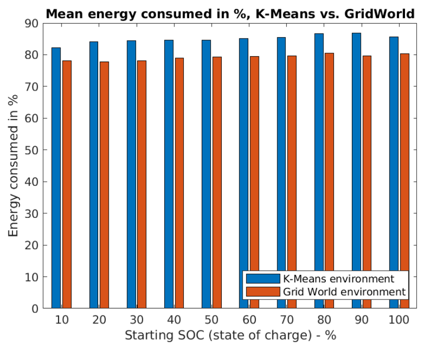
(a) : Mean time spent charging per shift.



(b) : Mean distance transferred per shift.

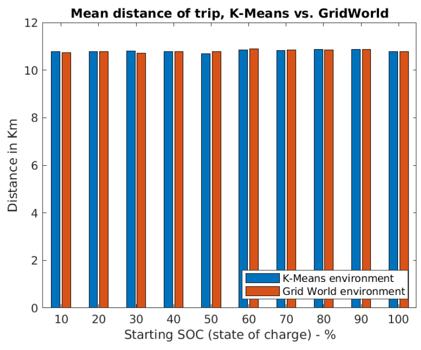


(c) : Mean amount of energy charged in % per shift.

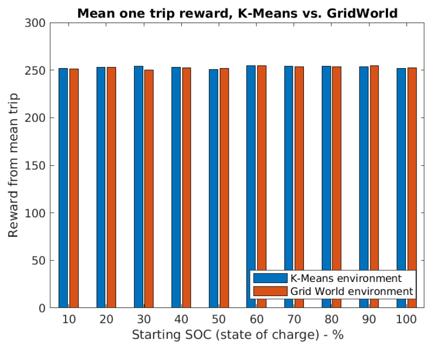


(d) : Mean amount of energy consumed in % per shift.

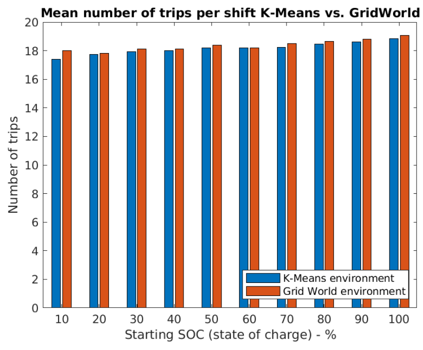
Figure B.2: Prague data set evaluation results – K-Means vs. Grid World part one.



(a) : Mean distance of one trip.



(b) : Mean reward from one trip.

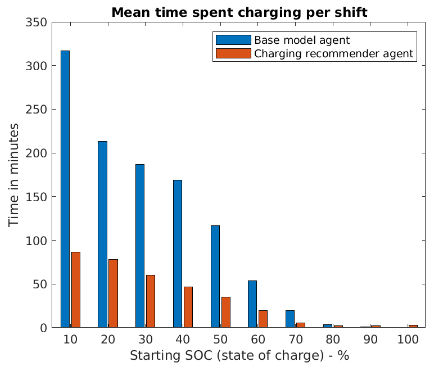


(c) : Mean number of trips per shift.

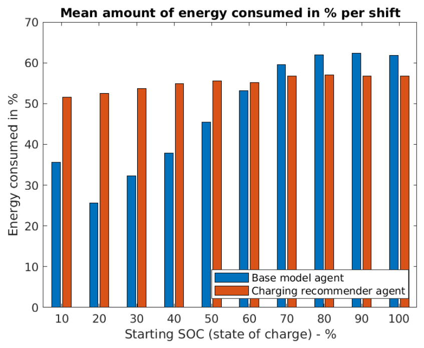
Figure B.3: Prague data set evaluation results – K-Means vs. Grid World part two.

B.2 New York Data Set Evaluation

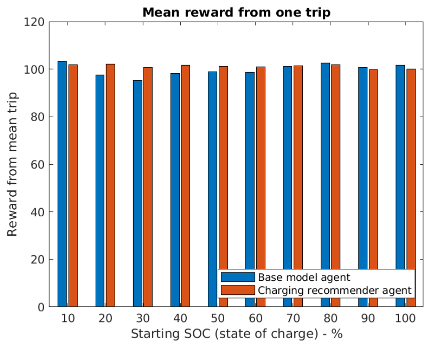
B.2.1 BMA vs. CRA (Grid World)



(a) : Mean time spent charging per shift.



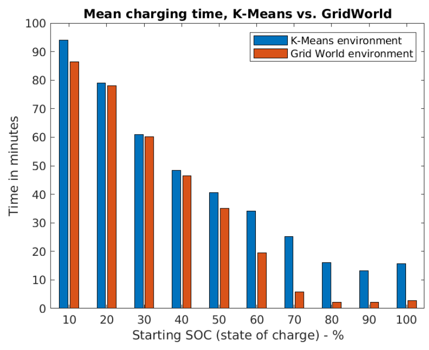
(b) : Mean amount of energy consumed in % per shift.



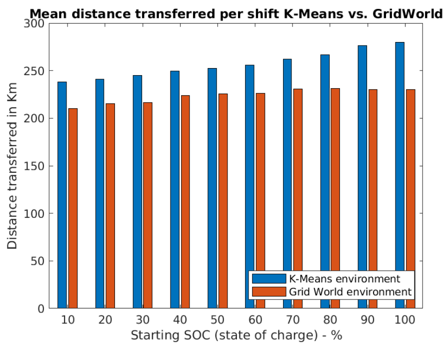
(c) : Mean reward from one trip.

Figure B.4: New York data set evaluation results – BMA vs. CRA.

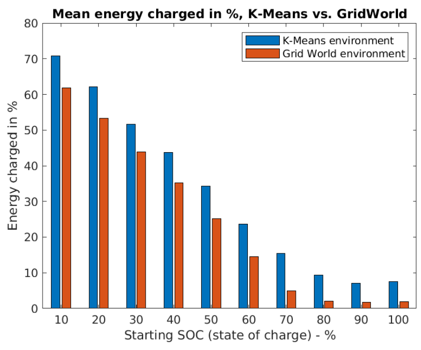
B.2.2 Grid World Environment vs. K-Means Environment



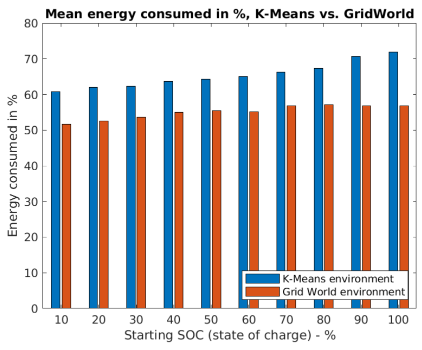
(a) : Mean time spent charging per shift.



(b) : Mean distance transferred per shift.

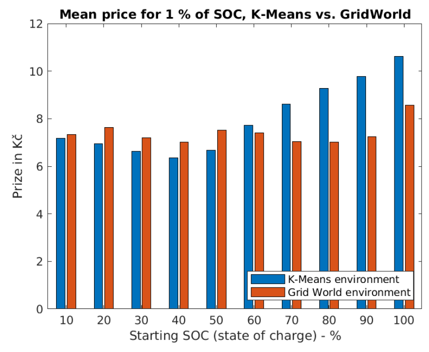


(c) : Mean amount of energy charged in % per shift.

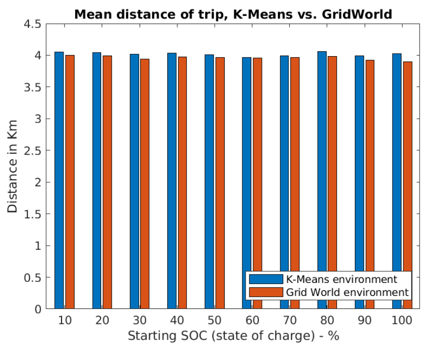


(d) : Mean amount of energy consumed in % per shift.

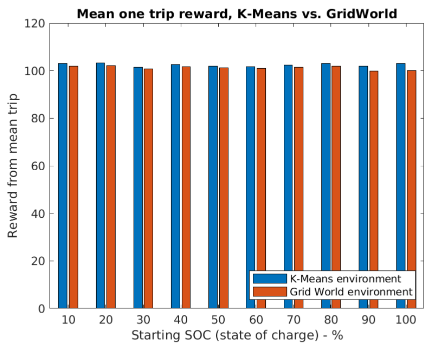
Figure B.5: New York data set evaluation results – K-Means vs. Grid World part one.



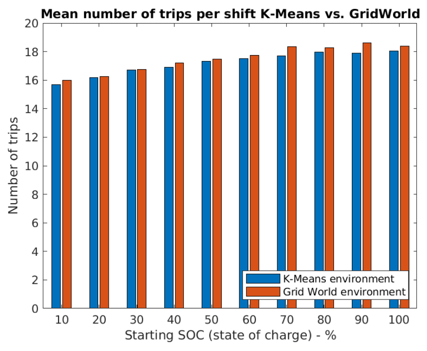
(a) : Mean Price paid for 1 % of SOC charged.



(b) : Mean distance of one trip.



(c) : Mean reward from one trip.



(d) : Mean number of trips per shift.

Figure B.6: New York data set evaluation results – K-Means vs. Grid World part two.