June 2021

Optimization and Machine Learning Methods for Solving Combinatorial Problems in Urban Transportation

Aigerim Bogyrbayeva
University of South Florida

Optimization and Machine Learning Methods for Solving Combinatorial Problems in Urban Transportation

by

Aigerim Bogyrbayeva

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Industrial Engineering
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Date of Approval:
June 10th, 2021

Keywords: Vehicle Routing, Neural Networks, Reinforcement Learning, Auction Design

Dedication

For the lasting memory of my grandparents, Duisenbek, Zhumash, Makan and Kamash. Their lifelong examples of self-sacrifice, strength and optimism have always guided me through difficult times.

Acknowledgments

I would like to express my greatest gratitude to Dr.Kwon for his guidance, support and patience in writing this dissertation. I deeply thank him for working with me and teaching me everything I know about research.

I also thank my mother Zhanat, father Askar, my mother-in-law Fatma and sister Mereili for their prayers, encouragement and trust. I would like to deeply thank my sister, Akzharkyn, for being a loving, caring and strong daughter and covering up for me at the moment we needed it the most.

My special appreciation goes to my husband, Yasin, for everything he has done for me. I thank Yasin for motivating me to start my Ph.D. in general and his wisdom, sense of humor and love shown every day that helped a lot to finish this dissertation.

Table of Contents

List of Tables

List of Figures

Abstract

This dissertation investigates three applications of emerging technologies for urban transportation. In the first chapter, we design a new market for fractional ownership of autonomous vehicles (AVs), in which an AV is co-leased by a group of individuals. We present a practical iterative auction based on the *combinatorial clock auction* to match the interested customers together and determine their payments. In designing such an auction, we consider continuous-time items (time slots) which are defined by bidders, and naturally exploit driverless mobility of AVs to form co-leasing groups. To relieve the computational burdens of both bidders and the auctioneer, we devise user agents who generate packages and bid on behalf of bidders. Through numerical experiments using the California 2010–2012 travel survey, we test the performance of the auction design.

In the second chapter, we propose a reinforcement learning approach for nightly offline rebalancing operations in free-floating electric vehicle sharing systems (FFEVSS). Due to sparse demand in a network, FFEVSS requires relocation of electrical vehicles (EVs) to charging stations and demander nodes, which is typically done by a group of drivers. A shuttle is used to pick up and drop off drivers throughout the network. The objective of this study is to solve the shuttle routing problem to finish the rebalancing work in minimal time. We consider a reinforcement learning framework for the problem, in which a central controller determines the routing policies of a fleet of multiple shuttles. We deploy a policy gradient method for training recurrent neural networks and compare the obtained policy results with heuristic solutions.

In the final problem, the application of the tandem of drone and truck for last mile delivery is proposed. To take advantage of the different properties of drone and truck to deliver goods to customers, an efficient routing algorithm is introduced based on reinforcement learning. The proposed method produces the routing policies of both drone and truck that identifies customers served by drone, customers served by turck and includes recharging nodes for drone. In this study we present, a novel model, called hybrid, consisting of an attention encoder and a LSTM decoder to effectively route both drone and truck.

Chapter 1: Introduction

Urban transportation refers to various systems designed to transport people and goods inside a city or to the nearest suburban areas. In the last decades, urban transportation has faced significant challenges with increasing urban population and density, as well as congestion and pollution. Nevertheless, emerging technologies such as electrical vehicles (EVs), unmanned aerial vehicles (drones), and autonomous vehicles (AVs) have a great potential to offer new forms of urban transportation. Especially, collaborative consumption trends in urban settings such as shared rides and on-demand services lays down a road to more sustainable transportation options that can be enhanced with emerging technologies. This dissertation discussed three applications of novel technologies in urban transportation and proposes practical tools such as models and algorithms to overcome their operational challenges.

In the first application, we discuss using AVs in a fractional ownership market, where a group of individuals can co-own a single AV. Such a novel form of vehicle ownership allows for better utilization of resources in urban transportation. However, for the fractional ownership market of AVs to work, a leasing company needs to match customers into groups based on their travel needs and determine prices for each customer. We propose a combinatorial iterative auction with bidder-defined items that allows customers to bid on a combination of their trips with no restriction on their time and length as well as to learn bidding prices needed to win bids. The auction outcomes determine both matching groups of customers and their payments. The auction consists of two stages, wherein the first stage customers bid for each time slot they are interested in and learn their individual bidding prices. We also introduce activity rules in the first stage to suppress the strategic behavior of bidders. In the second stage customers bid for the combination of trips as a final bid. We propose a bidding agent with several bidding options to bid on the behalf of the customer. In computational studies, we compare various bidding strategies and study the effect of *activity rules* on the bidders' payoffs. We find that the designed activity rules successfully remove the strategic behavior of bidders. We also find that *core-selecting payment rule* brings the largest revenue to the auctioneer in most cases.

In the second application, we solve the operational challenges of free-floating electrical vehicle sharing systems (FFEVSS), which represents an EV sharing system on-demand. FFEVSS offers a sustainable option for urban transportation, but faces a challenging problem of rebalancing the system, when EVs need to be charged and relocated to expected demand locations every night. In the current practice, a group of drivers in a shuttle is routed around for such a purpose. However, deploying a group of drivers to rebalance the system is costly, thus requiring an efficient routing of shuttles to rebalance the system in a minimal time. We propose a reinforcement learning approach to route multiple shuttles deployed in a network. In particular, we discuss the Markov Decision Process (MDP) formulation of the problem and introduce a policy learning method based on sequence to sequence to learn the routing policies of multiple shuttles. Our numerical studies show that unlike the existing solutions in the literature, the proposed methods allow solving the general version of the problem with no restrictions on the urban EV network structure and charging requirements of EVs. Moreover, the learned policies offer a wide range of flexibility, resulting in a significant reduction in the time needed to rebalance the network.

In the last application, we consider recent advancements in drone technology that lays out new opportunities in cost-effective last mile delivery. In particular, combining truck and drone to deliver products to customers is a promising tandem. For instance, truck has unlimited capacity and traveling range, but it has slow speed. Drone on the other hand has high speed, but it has limited traveling range and weight carrying capacity. Therefore, to take the advantages of their different properties, we must develop new routing decisions that enable recharging and reloading of drone. Formally known as Travelling Salesman Problem with Drone (TSPD), routing of both drone and truck in a shared urban network is NP-hard, highlighting the need for heuristic solutions. In this study we present, a novel model, called hybrid, consisting of an attention encoder and a LSTM decoder to effectively route both drone and truck. As our numerical studies show, the hybrid model performs well both in solution quality and computational time as compared to the heuristics existing in the literature.

In summary, the goal of this dissertation is to answer the following questions:

- How to design a market for fractional ownership of AVs?
- How to efficiently route multiple shuttles to rebalance FFEVSS?
- How to efficiently route drone and truck for last mile delivery?

The dissertation can be summarized as follows: in Chapter 2, we propose the design of an iterative combinatorial auction with bidder-defined items for fractional ownership of autonomous vehicles. Chapter 3 presents a reinforcement learning approach to route multiple shuttles to rebalance FFEVSS. In Chapter 4, we introduce a novel hybrid model to efficiently route drone and truck for last mile delivery. Chapter 5 gives the final remarks by summarizing the discussed problems and provides future research directions.

Chapter 2: Combinatorial Auction with Bidder-defined Items for Fractional Ownership of Autonomous Vehicles

The copyright permissions for reuse previously published material in this chapter can be found in Appendix A.

2.1 Introduction

Since the invention of the assembly line, vehicle ownership has been a distinctive part of the American culture, placing the country on the leading positions by vehicles per capita. The current model of vehicle ownership is, however, neither cheap nor efficient. In fact, the total auto-loan amount in the country exceeded \$1.24 trillion in 2018, while in 95% of the time cars are parked (Center Microeconomic Data , 2018; Shoup, 2005). Nevertheless, in recent years collaborative consumption has been accounted for dramatic changes in transportation. The wide spectrum of new services ranging from ride-sharing (Uber, Lyft and etc.) to peer to peer car renting (Buzzcar, Drivy and etc.) has influenced the traditional view of vehicle ownership. This can be observed in the increase of 7 years on the average age of new vehicle buyers (Kurz et al., 2016). Consequently, to satisfy the potential shift in consumer demand, car-manufacturers have launched various car-sharing services (Maven by GM, Audi on Demand, etc.), thus seeking new forms in vehicle ownership.

Motivated by the recent advancements in autonomous transportation technologies (Akbar et al.; Gu et al.), this study investigates a novel form of vehicle ownership called *fractional ownership* of autonomous vehicles (AVs), where an AV is co-leased by a group of individuals. Thus, 'fractional ownership' means that customers co-lease a vehicle and 'co-owners' mean co-lessees. The benefits of such ownership include reduced costs and increased utilization of vehicles coming from collaborative consumption. A fractional ownership model had been tested with regular vehicles (Ford Credit Link, Nissan Micra Go & Get) and it had encountered significant challenges. The most critical challenge came from the need to relocate a vehicle from the location of one customer to the location of another customer. Therefore, only customers living in the closest neighborhoods were allowed to group together, thus offering limited options. In addition, in the fractional ownership of the regular vehicles market, customers were asked to find co-leasing groups themselves, highlighting the absence of market design mechanisms. Because of such limitations, the above-mentioned programs have been discontinued.

In this study, we propose using AVs for the fractional ownership market. AVs naturally solve the above-mentioned relocation issue since they have driverless mobility. Therefore, using AVs in fractional ownership model allows to attract a large pool of customers from various neighborhoods with diverse travel needs. Also, when using AVs, *co-leasing* may be a more viable service due to the maintenance and parking costs. As compared to ride-sharing (Smet, 2021) or car rental services, the fractional ownership of AVs do not require customers to look for rides or rental cars every time, but instead provides a long-term predictable service on fixed rates. The premise of the fractional ownership model is in its convenience for customers to use the same AV for each ride while co-owning it with the fixed set of customers. For instance, a customer may store a child safety seat in the AV while taking low health risks associated with a shared vehicle. Lastly, the long term commitment nature of the fractional ownership allows both for customers and service companies to have a long planning horizon.

For a practical fractional AV ownership model to be successful, there must be no time conflicts among co-owners. In fact, the origins and destinations of trips also need to be considered to incorporate traveling times of empty AVs. Therefore, a suitable mechanism is needed to match customers with non-overlapping time-schedules together and avoid conflicts. There are two popular mechanisms to match customers. The first is matching-theoretic approaches, such as stable or maximal matching, commonly used by ride-sharing services like Uber and Lyft; see Wang et al. (2018) and Zhang et al. (2020), for example. In this case, a leasing company needs to solve two problems: to determine matching groups of customer and to set prices for each customer. Since fractional ownership of autonomous vehicles (AV) is a novel service, there is no benchmark for pricing, which poses substantial challenges for leasing companies. The second mechanism is based on auction theory. The outcomes of auctions not only determine matching groups of customers, but also determine the prices. Indeed, auction mechanism has a potential to generate more revenue compared to matching mechanism and it allocates time slots efficiently offering them to customers who value them the most. Also, we need to note that customers are most likely interested in using AVs for a combination of time slots (e.g. from home to work and from work to home). With these in mind, in this paper, we design a *combinatorial* auction market for fractional ownership of AVs as an alternative to the traditional full ownership model. In particular, the proposed auction exploits the unique feature of AVs, thus their driverless mobility in forming co-leasing groups.

Combinatorial auctions are suitable mechanisms to sell items or allocate resources in packages, instead of single items separately. They have been used widely across various industry sectors (De Vries and Vohra, 2003; Pekeč and Rothkopf, 2003; Milgrom, 2019) including allocation of the spectrum right licenses to telecommunication companies and Internet pricing (Hershberger and Suri, 2001). In transportation and logistics, combinatorial auctions have gained attention for selling airport departure and arrival slots (Rassenti et al., 1982), truckload transportation (Zhang et al., 2015), city bus route market (Cantillon and Pesendorfer, 2006), and railway industry (Kuo and Miller-Hooks, 2015). Recently, researchers suggest combinatorial auctions in the ride-sharing market for designing a more efficient shared mobility system (Hara and Hato, 2018), collaborative vehicle routing (Gansterer and Hartl, 2017) and public transportation systems (James et al., 2018).

In the proposed auction market, the auctioneer is a car manufacturer or leasing company that sells AVs, and the bidders are customers who are interested in co-leasing a car. The auctioneer sells time-slot packages to bidders through an auction that gives the winners the right to use the same vehicle in these time-slots within a week for a certain period. Each time-slot package includes several time-slots covering the travel needs of customers.

Combinatorial auctions involve complex package valuation problems for bidders and allocation problems for the auctioneer. Customers have to value time slot packages. The value of these packages may be different from the summation of the values of the individual time slots. Iterative combinatorial auctions have been introduced to address this *preference elicitation problem.* In particular, in iterative auctions bidders can bid iteratively, receive feedback based on their rivals' valuation and adjust their valuations (Pekeč and Rothkopf, 2003). This feedback information is valuable for the new products, where there is no benchmark for the pricing. Indeed, fractional ownership of AVs can be considered as a new product with limited valuation insight for a customer. Furthermore, the dynamic nature of iterative combinatorial auctions, where valuation information of time slots is exchanged between customers, may potentially lead to higher revenue compared to a single round combinatorial auction (Parkes, 2006). Because of the above-mentioned reasons, the majority of combinatorial auctions with applications in various industries (spectrum auction, real estate and etc.) are iterative in nature.

The existing combinatorial auction designs, however, do not fully capture the nature of the bids in the market for fractional ownership of AVs. For instance, in determining winning bids, the problem under study involves additional constraints to avoid time-conflicts between bidders. To address such issues, Takalloo et al. proposed a single-round, combinatorial auction market with user defined continuous-time items for the fractional ownership of AVs. The auction design of Takalloo et al. is based on the well-known Vickrey-Clarke-Groves (VCG) mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973), which possesses many desirable properties, but can suffer from low revenue for the auctioneer. In addition, due to the complex nature of combinatorial auctions, a single-round auction may limit opportunities for bidders to fully learn about the market and express their preferences. In this paper, we extend the settings of Takalloo et al. and design an *iterative* auction, as opposed to a single-round auction, to overcome these limitations of the VCG mechanism.

The proposed Combinatorial Clock Auction (CCA) includes two stages: the clock stage and the supplementary stage, which are designed to help customers to bid efficiently. The *clock stage* consists of multiple rounds and gives insight to the bidders about the market value of items. In each round, bidders submit bids for a package of time slots and observe the ask prices from an auctioneer. Even though it is possible to design a single round auction and avoid iterative bidding, in that case, customers have only a single shot to bid. Thus, in a single round auction customers bid without any insights about the competitiveness of their bids. Instead, in the proposed auction the clock stage serves as the *price discovery* for bidders (Ausubel et al., 2006). In the *supplementary stage* consisting of a single round, bidders submit their final bids considering the ask prices from the clock stage. Consequently, the auctioneer solves the *winner determination problem* (WDP) considering the bids from both clock and supplementary stages and calculates the payments. To suppress the strategic behavior of bidders such as when bidders avoid bidding until the last round or bid intentionally on undesired time slots, we use *activity rules* in both the clock stage and the supplementary stage. In the auction, to elevate the burden of iterative bidding, customers are given a choice to use user agents, a software tool designed to bid on behalf of bidders. The proposed user agents offer several bidding strategies to customers and propose bidding packages based on customers' travel needs. We also consider different payment rules and compare them together to study the revenue of auctioneer. To the best of our knowledge, this is the first study investigating the CCA as a market design for applications in transportation.

2.1.1 Unique Challenges and Contributions

The setting of the problem under study is unique in several aspects. In most existing combinatorial auctions, products are pre-defined discrete items. In the proposed auction in this study, items are neither pre-defined nor discrete; instead, we consider *bidder-defined continuous-time items*. Every possible time interval becomes an item when a bidder finds it valuable. While bidder-defined continuous-time slots serve the interests of customers best and result in greater social welfare compared to discrete-time slot (Takalloo et al.), the infinite number of possible items cause new challenges as well. In particular, customers face large scale valuation problems for all possible items (time slots). Selecting a suitable set of time-slots requires substantial computational resources. In addition, due to its iterative nature the proposed auction, compared to a single round auction by Takalloo et al., results in the substantial increase in the number of bids, thus requiring new solution techniques both for customers and the auctioneer.

The contributions of this study can be summarized as follows. First, we design a unique iterative combinatorial auction for the market design of fractional ownership of AVs, namely *combinatorial clock auction with bidder-defined items*. Second, we devise a fast algorithm for determining the ask prices in the proposed auction setting. Third, we develop user agents with different bidding strategies for generating packages as a support tool for customers. Fourth, we test the performance of the proposed auction design under different *payment rules* with the simulation studies.

Our numerical experiments show that core-selecting payment rule results in a relatively high revenue compare to the other payment rules. We also find that activity rules, which are based on the eligibility points, are effective under the *bidder-defined continuous-time items* setting as they support consistent bidding while suppressing the strategic behavior of customers.

2.1.2 Outline

The remainder of the paper will proceed as follows: In Section 2.2, we describe the proposed auction design in the clock and the supplementary stages. In Section 2.3, we investigate several payment rules and their effect on the auction outcome. In Section 2.4, we introduce user agents who assist bidders in bidding in the clock stage and generating packages in the supplementary stage. In Section 2.5, we present numerical experiments based on the California 2010–2012 travel survey dataset and derive insights into the auction market for fractional AV ownership. Lastly, in Section **??**, we conclude the paper.

2.2 The Auction Design for Fractional Ownership of AVs

In this section, we design CCA for the fractional ownership of autonomous vehicles. First, we define the continuous-time items and packages in the fractional ownership CCA in Section 2.2.1. Next, we describe the first stage in the CCA which is the clock stage in Section 2.2.2. The clock stage can be viewed as a multi-round auction. In each round, the auctioneer announces the ask prices, and the bidders bid on the desired items. The auctioneer calculates the ask price based on the supply-demand balance. We present Algorithm 1 for calculating the ask prices in Section 2.2.2.1. To ensure that bidders bid actively throughout the clock stage and to remove the strategic behavior of bidders, we design some activity rules for the auction which are based on the eligibility points of the bidders. We describe the activity rules in the clock stage in Section 2.2.2.2. Next, we describe the supplementary stage which is the final round of the auction in Section 2.2.3. In the supplementary stage, bidders submit new bids based on the information they gain through the clock stage. Then, the auctioneer considers all the bids submitted in the clock stage and the supplementary stage to solve the WDP considering the spatial information of the bidders and the continuous-time items.

2.2.1 Auction Setting

To better understand the fundamental elements of the proposed auction design, consider a customer who is interested in using an AV in the following time slots: 7:30–8:00 AM, 4:40–5:10 PM. In this case, the items are two mentioned time slots. The customer may submit the following set of packages from the items: {7:30–8:00 AM}, {4:40–5:10 PM} and {7:30–8:00 AM, 4:40–5:10 PM}. However, each bidder can receive at most one package. More formally, we introduce the definitions of an *item*, a *package* and a *bid* in the combinatorial auction as follows:

Definition 1. *An item* in the CCA for fractional ownership of vehicles is a continuous time-slot uniquely defined by its start and end time that a customer selects based on her schedule. Unlike the traditional CCA, there are no predefined items in the proposed auction and a customer is given a choice to define an item.

Definition 2. *A package* is a combination of continuous time slots defined by a customer.

Definition 3. *A bid* is a package submitted by a customer indicating start and end time of her desired time slots along with spatial information of her trips.

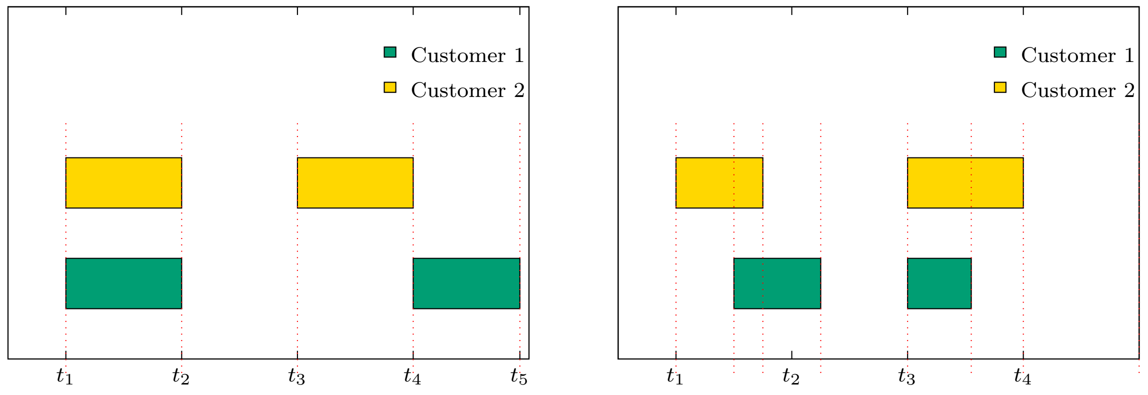
In the proposed combinatorial auction, the auctioneer is a car leasing company who wants to sell AVs, and the bidders are customers who are interested in co-leasing a car. The auctioneer offers a homogeneous fleet of vehicles for co-leasing, while interested customers enter into the iterative bidding. Customers submit their bids as combinations of time slots and provide spatial information for their trips. The *bidding language* in the auction is XOR; that is, the auctioneer will accept at most one bid from each bidder. The auction itself consists of two stages: *clock* and *supplementary* stages with a set of rules specified below.

2.2.2 The Clock Stage

The clock stage in the proposed auction serves as *price discovery* for each possible item and subsequently for each package, where package's ask price equals to the summation of ask prices of its items. In particular, at the beginning of each round of the clock stage, the auctioneer announces ask prices for items and bidders bid on the desired items. Note that the collection of items submitted by a bidder in round r form a package, which we call a clock stage package. The bid price for a clock stage package is computed as the summation of ask prices for items in the package. The customers can bid on the items that they bid on previously or on the new items. Ask prices increase for the items with excess demand. The clock stage continues until we have demand-supply balance or bids do not change in two consecutive rounds. At the end of the clock stage, bidders learn the minimum bid price required to win a particular item.

2.2.2.1 Ask Price Calculation

In the traditional CCA setting, finding supply-demand balance and determining the ask prices is relatively easy. Since items are predefined, the auctioneer can easily count the total demand for each item. Figure 2.1a shows the conventional CCA, in which items are defined as hourly time slots. The auctioneer counts demand for each hour and indicates demand-supply balance. Figure 2.1b represents the proposed CCA with continuous bidder-defined items (time slots). Each time slot can be identified uniquely by a start time and an end time. Placing the start time and the end time of all submitted time slots in the time horizon results in a set of *time slices*. Each item (time slot) includes a set of consecutive time slices. To determine the ask prices for each item, the auctioneer specifies its corresponding set of time slices. For a particular time slice, if the demand (the number of items which include that time slice) is greater than the supply (the fleet size), then there will be a price increment ϵ_p for that particular time slice. The ask prices for an item in each round will be updated by adding the summation of price increments for the corresponding time slices. Note that we need to calculate the ask prices in each round of the auction. Algorithm 1 shows the pseudo-code of the algorithm for determining the ask price efficiently, which is explained below.

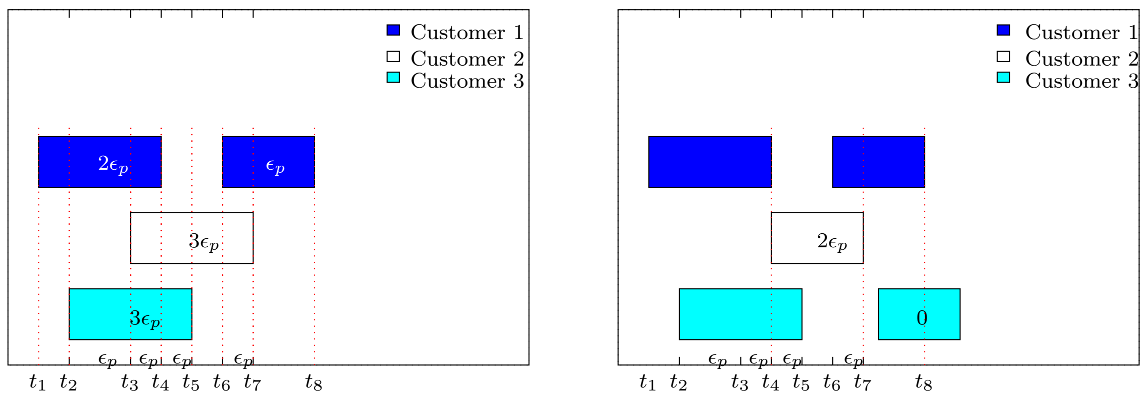


(a) Discrete auctioneer-defined items (b) Continuous bidder-defined items

Figure 2.1: Comparing the items in the conventional CCA and the proposed auction

Algorithm 1: Ask price calculation in round r

```
Input: All submitted items in round $r$ denoted by $\mathcal{I}$, fleet size $|\mathbf{H}|$, and price
       increment $\epsilon_p$
Output: Vector of ask price increment in $r$-th round $\mathbf{p}$
Initialization: times $\leftarrow \emptyset$, soe $\leftarrow \emptyset$ ; // soe for 'start or end'
for $j = 1$ to length($\mathcal{I}$) do
    push(times, $s_j$); push(soe, $1$); // 1 for start time
    push(times, $e_j$); push(soe, $-1$); // -1 for end time
sorted_times, sorted_idx $=$ sort(times);
sorted_soe $=$ soe[sorted_idx]; // soe is sorted in the same order as sorted_times
$j \leftarrow 1$; count $\leftarrow 0$; conflicts $\leftarrow \emptyset$; $\mathbf{p} \leftarrow \mathbf{0}$;
while $j \leq$ length(sorted_times) do
    same_time_idx $\leftarrow$ findall(sorted_times, sorted_times[$j$]) // Finding a vector of
        indices with the same time value as sorted_times[$j$]
    foreach $m \in$ same_time_idx do
        if sorted_soe[$m$] $= 1$ then
            count $\leftarrow$ count $+ 1$; // for start time
            push(conflicts, same_time_idx[$m$]);
        else
            count $\leftarrow$ count $- 1$; // for end time
            remove(conflicts, same_time_idx[$m$]);
    if count $> |\mathcal{H}|$ then
        foreach $n \in$ conflicts do
            $p_n \leftarrow p_n + \epsilon_p$;
    $j \leftarrow j +$ length(same_time_idx);
```


Customer 1
Customer 2
Customer 3
$2\epsilon_p$
ϵ_p
$3\epsilon_p$
$3\epsilon_p$
ϵ_p ϵ_p ϵ_p ϵ_p
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8
Customer 1
Customer 2
Customer 3
$2\epsilon_p$
0
ϵ_p ϵ_p ϵ_p ϵ_p
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

(a) Regular bids (b) New bids

Figure 2.2: Price determination through conflict detection in an auction with a single vehicle ($|\mathcal{H}| = 1$)

An intuitive explanation of the algorithm is illustrated in Figure 2.2a which represents a CCA with three customers and a single vehicle. To find the ask price increments, we first collect all the bids and place all the time slots into the time horizon and construct the time slices (from t_1 to t_2, from t_2 to t_3, and so on). Next, we set a price increment ϵ_p, where p stands for price, for each time slice with excess demand. Note that the value of ϵ_p is the same in all rounds of clock stage. Then, the ask price for each item is updated by adding the summation of price increments for its corresponding time slices. For example, as Figure 2.2a represents, the second customer's bid consists of a single item which overlaps in three time-slices with other items. In this case, the ask price for the second customer will be increased by $3\epsilon_p$.

While determining the ask prices for the items that have been present in the previous rounds is a relatively easy task, it is not the case for new items. For example, as shown in Figure 2.2b, the second customer submits a single item bid again in the next round, but with a new item. Since this item has not been present in the previous rounds, the auctioneer does not know the initial ask price. The auctioneer may set the initial ask price to 0 for all new items, but it may lead the customers to submit new items all the time by slightly changing their previous bids. Therefore, the auctioneer requires a more effective way to determine the initial ask prices for the new items. We propose an alternative approach for setting the ask prices for new items. As can be seen in Figure 2.2b, for the new items submitted by the second customer, we consider the corresponding time slices in the previous rounds. The ask price for the new item will be set to the sum of price increments for the corresponding time slices in the previous rounds. For example, if we consider Figure 2.2a as the first round, and Figure 2.2b as the second round, the initial ask price for Customer's 2 new items in the second round will set to $2\epsilon_p$. Similarly, the initial ask price for Customer 3's new item will be 0, since there is no conflict in any corresponding time slice in the first round. In the clock stage, as the auction proceeds, the ask prices for the items increase and as a result, the demand decreases until we achieve the supply-demand balance.

2.2.2.2 Activity Rules for the Clock Stage

Iterative combinatorial auctions, in general, are vulnerable to strategic behaviors of bidders. For instance, a bidder may not bid on her preferred items until the last round to keep their prices from rising. To suppress such behavior, the CCA has another design element called activity rules which are present in both the clock stage and the supplementary stage. These rules restrict bidders to enforce them to bid actively and constantly through the clock stage in order to be able to submit competitive bids in the supplementary stage.

Ofcom (2011) proposes some activity rules that have gained popularity among practitioners because of their simplicity. These activity rules are based on *eligibility points* that bidders purchase with their initial deposit before entering an auction. The auctioneer assigns eligibility points to each item based on historical demand for that item. High historical demand for a time-slot results in a high value of the eligibility point for that time slot. In particular, the auctioneer determines the eligibility point for each hour (12:00 AM–1:00 AM, 1:00 AM–2:00 AM, ...) based on the historical data for the demand. Then the eligibility point for each item i (time slot) will be determined as:

$$e_i = \sum_{t \in \mathcal{T}} \delta_i^t e_t$$

where e_t is the eligibility for each hour time-slot, and e_i is the eligibility for item i, and δ_i^t is the proportion of item i that falls in an hour time slot t. Formally, we can define the eligibility point of items and packages as follows:

Definition 4 (Eligibility Points of Items and Packages)**.** Let **I** denote the set of items in an auction and let e_i denote a preassigned eligibility point of item i. Then the eligibility point of package k denoted by $E(k)$ is defined as the sum of Eligibility Points of items in the package, i.e. $E(k) = \sum_{i \in k} e_i$.

Definition 5 (Eligibility Points of Bidders)**.** If bidder j has submitted a bid on package k in round $r - 1$, we say that bidder j possesses eligibility point $E_r^j = E(k)$ at the beginning of round r. This quantity is known as *eligibility* of bidder j in the literature (Ausubel et al., 2011).

The bidders start with an initial eligibility point based on the bidders' initial deposit. We can state the following activity rule for the clock stage: for each bidder, the eligibility points of her package at any round should not exceed her eligibility points at the beginning of the round. Thus, bidders are required to bid large quantities to maintain their eligibility in future rounds.

According to the activity rule mentioned above, the bidders need to bid constantly throughout the clock stage. If they do not bid actively in each round of the clock stage, their eligibility level decreases and they will lose the chance to submit packages in the supplementary stage. Figure 2.3 summarizes the clock stage of the proposed auction.

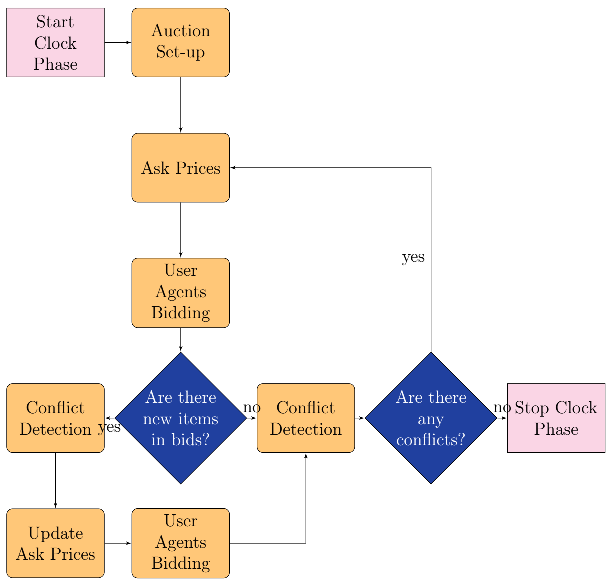


Figure 2.3: Clock stage flow chart

2.2.3 The Supplementary Stage

The supplementary stage is the final round for the submission of bids. Unlike the clock stage packages, whose bid prices are calculated as the summation of ask prices for the corresponding items, the bid prices of supplementary packages are directly determined by the bidders. In fact, bidders submit new packages based on the price discovery in the clock stage (where they learn the ask price for each item) and their budget.

Next, considering the clock stage packages and supplementary stage packages, the auctioneer solves the winner determination problem (WDP) to determine the set of winners in the auction. With the aim to relate the clock stage bids with the supplementary stage bids, the following activity rules for the supplementary stage were introduced in Ofcom (2011):

- If a bidder submits a package exactly the same as in the final clock round (FCR), the bidding price for that package should be greater or equal to the FCR package bidding price
- If the FCR package is a subset of the submitted package in the supplementary stage, in other words, new bids are introduced in the supplementary stage, then the following

condition should hold:

$$c_j(b) \leq C_j(b^r) + p^r(b) - p^r(b^r) \tag{2.1}$$

where $c_j(b)$ is the bidding price of bidder j for package b in the supplementary stage; r is the last round when bidder j was eligible for package b; b^r is the package submitted by bidder j in the r-th round of clock stage; $C_j(b^r)$ is the maximum bid price by bidder j for package b^r in any round of the clock stage or in the supplementary stage; $p^r(b^r)$ is the ask price in round r for the clock stage package b^r; and $p^r(b)$ is the ask price for package b in round r (note that if this package has not been bid in round r, we determine the bidding price for that package in the same way as we determine the ask price for newly introduced items in round r).

As mentioned above, the activity rules in the supplementary stage, put some restriction on bidders' bidding based on their bidding history in the clock stage. Thus, from Equation (1) we can see that the difference between the bids prices for the newly introduced bid b and a bid b^r cannot exceed their respective ask prices' difference in round r. As for the posted bids, their bid price in the supplementary stage should be no less than their maximum bid price in the clock stage. This way the supplementary stage activity rules impose a relative cap on a bid amount, which can strongly limit the competitiveness of packages not submitted in the clock stage.

We also note that the proposed auction design may include reservation prices for an auctioneer. Thus, if the trips' costs for the winning bids exceed the bids' prices, an auctioneer may not accept the auction outcome. In order to avoid such situation, all bidders before entering the auction will be provided with the cost calculation formula. Then in the supplementary stage of the auction, the bidders submit bid prices no less than the summation of the costs of trips in their packages. Next, we formulate the WDP for the fractional ownership CCA.

2.2.3.1 Winner Determination Problem for the Fractional Ownership CCA

Note that in the supplementary stage, customers submit start and end times of their trips along with their bidding prices and spatial information. This information can be used to obtain the approximated commuting time between the bidder's locations. Also, only a single bid of each bidder can be determined as a winner as mentioned before. Then each winning bidder is assigned to a single AV, which will serve all her winning trips. Next, a simple algorithm can be used to determine the conflicting bids, which are defined as bids with overlapping trips. In particular, each pair of bids are examined to check if they overlap by considering the start location, start time and end location, end time of each trip in the bids. Based on the locations of the bidders, we estimate the travel time of an empty AV between locations of two bidders. If the end time of the first bidder plus the travel time of an empty AV is greater than the start time of the second bidder, we consider two bids as conflicting. Once the conflicting bids determined, the auctioneer can solve the following winner determination problem (WDP) with the objective to maximize social welfare, as formulated in Takalloo et al., considering all the bids from the clock and supplementary stages:

$$
\begin{aligned}
\textsf{(WDP)} \quad \max_{x_{bh}} \quad & \sum_{j\in\mathcal{J}}\sum_{b\in\mathcal{B}_j}\sum_{h\in\mathcal{H}} c_j(b)x_{bh} && (2.2)\\
\text{s.t.} \quad & \sum_{b\in\mathcal{B}_j}\sum_{h\in\mathcal{H}} x_{bh} \le 1 \quad \forall j\in\mathcal{J} && (2.3)\\
& x_{bh}+x_{lh}\le 1 \quad \forall h\in\mathcal{H}, j,q\in\mathcal{J}, b\in\mathcal{B}_j, l\in\mathcal{B}_q : b,l \text{ are conflicting} && (2.4)\\
& x_{bh}\in\{0,1\} \quad \forall j\in\mathcal{J}, b\in\mathcal{B}_j, h\in\mathcal{H} && (2.5)
\end{aligned}
$$

In the above formulation, $\mathcal{J}$ denote the set of bidders and $\mathcal{B}_j$ denote the set of bids submitted by bidder $j\in\mathcal{J}$. Similarly, $\mathcal{B}_q$ is the set of bids of bidder q, who has a conflicting bid $l\in\mathcal{B}_q$ with bid $b\in\mathcal{B}_j$ of bidder j. A binary decision variable x_{bh} indicates whether bid b is assigned to vehicle $h\in\mathcal{H}$ or not where $\mathcal{H}$ is the set of vehicles. Constraint (2.3) ensures that at most one bid of each customer will be determined as a winner. Constraint (2.4) is a conflict constraint, which ensures that two conflicting bids do not share a vehicle. It usually takes a considerable amount of time to find a high-quality feasible solution for problem (WDP) for large instances.

However, we can use a sequential heuristic to solve the problem by decomposition. Since we assume a homogeneous fleet of vehicles, we can decompose the combinatorial auction problem into a $|\mathcal{H}|$-round single vehicle combinatorial auction, following the approach of Takalloo et al.. At each round, considering the set of remaining bidders, we solve the winner determination problem for a single vehicle and find the winners. Then, we update the set of bidders by excluding the winners from the list of bidders and go to the next round. This procedure continues until we assign all the vehicles to the bidders.

At the end of the supplementary stage, after determining the winners, the auctioneer calculates payments based on payment rules that will be discussed in Section 2.3.

2.3 Payment Rules

Choosing a suitable payment rule is an important part of the auction design, as it influences the bidding strategy of bidders and the revenue for the auctioneer. However, there is no universal pricing scheme that guarantees both incentive compatibility and high revenue. For instance, even though setting payments to submitted bid amounts is an intuitive choice, Ausubel et al. (2014) have shown that such payment rule results in demand reduction. In this study, we consider three pricing rules; namely, proxy payments, VCG payments (Vickrey-Clarke-Groves payments) and core-selecting payments. We introduce these payment rules in this section.

2.3.1 Proxy Payments

The proxy phase (Ausubel et al., 2006) has been proposed as an alternative to the supplementary stage bidding. In the proxy phase, which follows the clock stage, first, each bidder submits her packages and their valuations to her own *proxy agent*. Then, the proxy agents enter an iterative proxy auction on behalf of bidders. After each round r, the auctioneer determines the provisional winners and increases the ask price p_b^r for package b by ϵ_p. Then the agents select the set of packages with the nonnegative utility values to bid in the next round by solving the following problem.

$$\max_{x_b} \sum_{j \in \mathcal{J}} \sum_{b \in \mathcal{B}_j} (v_b - p_b^r - \epsilon_p) x_b \tag{2.6}$$

$$\text{s.t. } x_b \in \{0,1\} \qquad \forall j \in \mathcal{J}, b \in \mathcal{B}_j \tag{2.7}$$

where x_b indicates whether bid b is selected by proxy agent or not. The auction finishes when there are no more bids offered by any agents and the final outcome determines winners and their payments. Thus, we can look at *proxy payments* as iterative first-price payments that are in the core; there is no other set of bidders willing to pay more than the selected set of winners.

2.3.2 VCG Payments

The *Vickrey-Clarke-Groves (VCG)* mechanism is known to satisfy incentive compatibility. Under the VCG payment rule, truthful bidding is the dominant strategy for the bidders. VCG payment π_j for bidder j can be determined according to the following rule:

$$\pi_j^{\text{VCG}} = Z^*_{\text{WDP}_{-j}} - \left(Z^*_{\text{WDP}} - \sum_{b \in \mathcal{B}_j} c_j(b) x_b^* \right) \tag{2.8}$$

where Z^*_{WDP} and $\mathbf{x}^*$ are the optimal objective function value and optimal allocation of winner determination problem under the set of bidders $\mathcal{J}$ and $Z^*_{\text{WDP}_{-j}}$ is the optimal objective function value of the winner determination problem under the set of bidders $\mathcal{J} \setminus \{j\}$.

Although theoretically interesting, the VCG mechanism has shown serious practical problems (Rothkopf, 2007). It usually takes a considerable amount of time to solve the winner determination problem optimally. However, it is possible to solve relatively large problems with a small optimality gap. When the solution is suboptimal, incentive compatibility and rationality do not necessarily hold under the VCG mechanism.

Next, we consider core-selecting payment (Cramton, 2013), which is more practical compared to VCG and entails the desirable auction properties such as rationality, efficiency, and core property (Day and Cramton, 2012).

2.3.3 Core-Selecting Payments

Payments are in the core, if there is no other set of bidders willing to pay more than the set of selected winners (Day and Cramton, 2012). Suppose b^* denote the winning bid of bidder j, and $c_j(b^*)$ denote the bidding price for package b^*. We wish to determine the payment amount for b^* which satisfies the core property. Day and Cramton (2012) propose the following quadratic program to compute the core-selecting payments by minimizing the Euclidean distance between VCG payments and core-selecting payments:

$$\min_{\pi_j} \sum_{j \in \mathcal{W}} \left(\pi_j - \pi_j^{\text{VCG}}\right)^2 \tag{2.9}$$

$$\text{s.t.} \sum_{j \in \mathcal{W} \setminus \mathcal{S}} \pi_j \geq \beta_{\mathcal{S}} \qquad \forall \mathcal{S} \in \mathcal{S}' \tag{2.10}$$

$$\pi_j \leq c_j(b^*) \qquad \forall j \in \mathcal{W} \tag{2.11}$$

where $\mathcal{S}$ denotes any set of bidders who are willing to offer more than the total payment of winners in $\mathcal{W}$ and $\mathcal{S}'$ denotes the set of all such possible sets. Variable π_j is the core-selecting payment for winner $j \in \mathcal{W}$, π_j^{VCG} is the VCG payment for bidder j and $c_j(b^*)$ is the bidding price for the winner j. Parameter $\beta_{\mathcal{S}}$ denote the aggregate payment offered by set of bidders $\mathcal{S}$. Constraint (2.10) ensures that the total payment of the winners is greater than or equal to the payments offered by any other set of bidders. Constraint (2.11) makes sure to satisfy rationality by enforcing the individual payments of the winners to be not greater than their offered bid amounts.

Day and Cramton (2012) propose an iterative approach to solve (2.9)–(2.11) which finds a set of bidders $\mathcal{S}$ in each iteration by changing the bids' prices of the bidders in set $\mathcal{W}$. In particular, the WDP is resolved at each iteration by setting the bid prices of the bids of each bidder in set $\mathcal{W}$ to $c_j(b) + (c_j(b^*) - \pi_j^k)$. The resulting new set of winners, $\mathcal{S}$ is then used to solve (2.9) - (2.11). We can summarize this procedure as follows:

- Step 0. Set $k = 0$, $S' = \emptyset$ and $\pi_j^0 = \pi_j^{\text{VCG}}$.
- Step 1. Set $c_j^{\text{new}}(b) = c_j(b) + (c_j(b^*) - \pi_j^k) \quad \forall j \in \mathcal{W}, b \in \mathcal{B}_j$.
- Step 2. Solve the WDP in (2.2)–(2.5) and define $\mathcal{S}$ as the corresponding set of winners. Add $\mathcal{S}$ to set $\mathcal{S}'$.
- Step 3. Set $\beta_{\mathcal{S}}^{k+1} = \sum_{j\in\mathcal{S}} c_j(b^*) - \sum_{j\in\mathcal{S}\cap\mathcal{W}}(c_j(b^*) - \pi_j^k)$ in (2.10)–(2.11).
- Step 4. Solve (2.9)–(2.11) to generate a new payment π_j^{k+1}, go to Step 1.
- Stopping rule: The process repeats until the WDP with modified bidding prices does not generate any new set of bidders $\mathcal{S}$.

It is worth mentioning that in core-selecting payment method, we do not change the original set of winners $\mathcal{W}$, but we update their payments each time we solve (2.9) - (2.11).

2.4 User Agents in the CCA for Fractional Ownership of AVs

In general, vehicle ownership requires a substantial investment from household income. As a result, customers may need to spend a considerable amount of time (e.g. several weeks) in bidding in CCA for AVs. Moreover, it is not uncommon to see hundreds to thousands of bids from each bidder in combinatorial auctions (Olivares et al., 2012). Considering the time-consuming and the complex bidding process of the CCA, customers may be interested in choosing user agents. User agents communicate with bidders to assist them in the bidding process. In particular, as a supporting tool, user agents assist bidders by bidding through the clock stage on behalf of them and by generating competitive packages in the supplementary stage.

2.4.1 Bidding Strategies in the Clock Stage

The user agents relieve the computational burden of bidders. In particular, user agents select items to bid in the clock stage according to some strategies which are determined by the bidders. We propose the following bidding strategies for the user agents.

- *Strategy 1*: Under the first bidding strategy, it is assumed that bidders know the customers' exact trip schedule. Under this strategy, customers submit the set of must-have and optional trips to the user agents. Considering the budget restriction and the eligibility points, user agents first consider only the must-have items and solve the following binary optimization problem to select the must-have items to bid in the r-th round of the clock stage for bidder j:

$$\max_{\mathbf{w}} \sum_{i \in \mathcal{I}_j} e_i w_i \tag{2.12}$$

$$\text{s.t.} \sum_{i \in \mathcal{I}_j} e_i w_i \leq E_j^r \tag{2.13}$$

$$\sum_{i \in \mathcal{I}_j} p_i^r w_i \leq B_j \tag{2.14}$$

$$w_i \in \{0,1\} \qquad \forall i \in \mathcal{I}_j \tag{2.15}$$

where $\mathcal{I}_j$ is the set of bidder j's must-have items, w_i is a binary decision variable which indicates whether item $i \in \mathcal{I}_j$ is selected for a bid or not, p_i^r is the ask price of item i in round r, e_i is the eligibility point required for item i, and E_j^r is the eligibility of customer j at the beginning of round r. Constraint (2.13) satisfies activity rules in the clock stage and Constraint (2.14) considers the budget limitations of the customer. The objective function maximizes the eligibility of the bidder, which helps her maintain her eligibility in the future rounds and bid on more items. After selecting must-have items, if any eligibility points remained, the user agent solves Problem (2.12)–(2.15) for optional trips, by letting $\mathcal{I}_j$ be the set of optional trip items of bidder j. In such a case, we must first update the values of both E_j^r and B_j to account for used resources to select must-have items.

- *Strategy 2*: In the second bidding strategy, customers are required to submit their valuation for desired time slots along with the budget constraints. We believe this is not a demanding task for customers given the ride-sharing services benchmark prices. Then user agents solve the following optimization problem for customer j in each round r:

$$\max_{\mathbf{w}} \sum_{i \in \mathcal{I}_j} (v_{ij} - p_i^r) w_i \tag{2.16}$$

$$\text{s.t. } (2.13), (2.14), (2.15) \tag{2.17}$$

where v_{ij} is the value of item i for bidder j. In the second strategy, the user agent maximizes the utility of customer j by taking into account the ask prices for items in the current round.

- *Strategy 3*: Under the third strategy, at the beginning of auction, instead of submitting the exact time schedule, customers submit an acceptable time range for each trip. Then, in each round of the clock stage, the user agents select time slots in the submitted ranges with the lowest ask prices. In order to find the lowest priced time slots, the user agents consider the demand in each hour provided by the auctioneer. Algorithm 2 presents a suitable approach to find a set of items with the lowest price for the bidders. (Note that since the number of continuous time-slot in each range is infinite, the user

Table 2.1: Bidding strategies item selection in clock stage

Trips	EP	Ask Price	Trip type	Valuation	Time range
M 7:15–8:00 AM	15	16	must-have	11.03	6:45–8:30 AM
W 5:00–6:00 PM	20	14	must-have	15.72	4:30–6:30 PM
Th 12:30–1:00 PM	10	8	optional	10.90	12:00–1:30 PM

agents consider only a finite subset of them by discretizing the possible start time within the range). After obtaining the set of items with the lowest prices, user agents solve the same optimization problem as in Strategy 2 to determine the items.

Table 2.1 gives an example of the bidding strategies in fractional ownership CCA. A customer provides as an input three trips with start time and end time information, out of which a user agent needs to select some items for bidding in the next round. The customer also indicates her budget as $50 and her current eligibility as 40 points.

After receiving the given input, a user agent calculates eligibility points and ask prices for each trip (columns EP and Ask Prices in Table 2.1). If the customer selects the first strategy, the user agent bid on the first and the second trip, since both of them are must-have trips and have high eligibility. Under the second bidding strategy, the customer provides her valuation for each trip (see Valuation column). In this case, the user agent bid on the second and the third item, because they generate nonnegative payoffs. Lastly, under the third strategy, the customer provides a time range for each trip, which suits customer travel needs (see Time range column). For instance, for the first trip, a customer may consider 6:45–8:30 AM as her acceptable time range. Then, user agent may consider 6:45–7:30 AM, 7:15–8:00 AM and 7:45–8:30 AM as possible items to bid. To choose the best item, the user agent determines the time slot with the lowest price.

Note that as discussed before, the activity rules aim to remove the strategic behavior of bidders. Hence, under an effective set of activity rules, we expect that the bidders' payoff does not differ from each other significantly, where payoff is defined as the difference between a customer's valuation of the bid and bid price. However, without any set of activity rules, we

Algorithm 2: Package generation algorithm in the r-th round of Clock Stage under Strategy 3

```
Input: The set of desired items for the customer $\mathcal{I}^r$, aggregate demand until $r$-th
       round each item $i$, $D_i^r$, demand in $r$-th round for item $i$, $d_i^r$, set $\mathcal{R}^r$ which
       includes the desired acceptable time range $(l_i, u_i)$ for each item $i$, number of
       vehicles $|\mathbf{H}|$, price for item $i$ submitted in $r$-th round $p_i^r$, time increment by
       user agents $\epsilon_t$, and trip length $(e_i - s_i)$ for each item $i$.
Output: Set of items with the lowest prices $\mathcal{I}^{\min}$
1  Initialization: $\mathcal{I}^{\min} \leftarrow \emptyset$;
2  foreach $i \in \mathcal{I}^r$ do
3      $D_i^r = D_i^{r-1} + d_i^r$;
4      if $p_i^r = p_i^{r-1}$ then
5          push($\mathcal{I}^{\min}, i$);
6      else
7          if $D_i^r > |\mathcal{H}|$ then
8              remove($\mathcal{I}^r, i$);
9              $i \leftarrow$ find_item($i, \mathcal{R}^r, \epsilon_t$); // finding set of items with the lowest
                  prices
10             push($\mathcal{I}^r, i$);
11         else
12             push($\mathcal{I}^{\min}, i$);

13 function find_item($i, \mathcal{R}^r, \epsilon_t$):
14     $k \leftarrow [l_i, l_i + e_i - s_i]$;
15     $k_{\min} \leftarrow k$;
16     $p_{\min} \leftarrow p_k^r$;
17     $t \leftarrow l_i$;
18     while $t + e_i - s_i \leq u_i$ do
19         $t \leftarrow t + \epsilon_t$;
20         $k \leftarrow [t, t + e_i - s_i]$;
21         if $p_k^r < p_{\min}$ then
22             $k_{\min} \leftarrow k$ ;
23             $p_{\min} \leftarrow p_k^r$;
24     return $k_{\min}$;
```

Table 2.2: Time slots submitted by a customer to the user agent

Trips	Monday	Tuesday	Wednesday
1	8:00–9:00 AM	8:00–9:00 AM	8:00–9:00 AM
2		5:00–6:00 PM	12:00–1:00 PM
3			5:00–6:00 PM

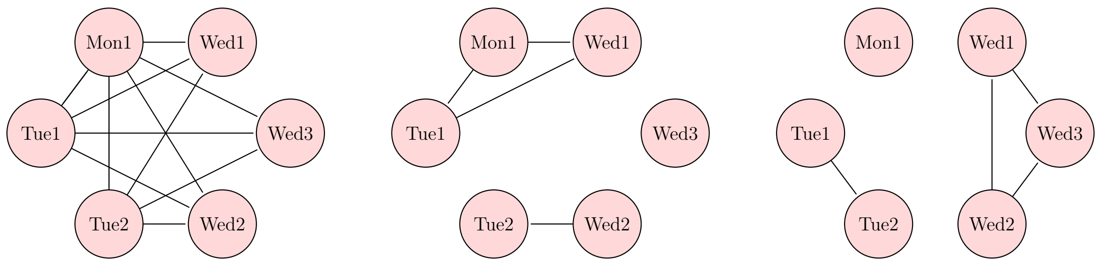
expect that the bidder's payoff under the third strategy becomes higher. We will study the impact of activity rules on bidders' payoff under different strategies by presenting numerical experiments in Section 5.

2.4.2 Automatic Package Generation in the Supplementary Stage

Generating competitive packages is challenging for bidders. The problem intensifies under the proposed customer-defined continuous time slots, where an infinite number of items exist. While iterative bidding in the clock stage reveals the minimum bid amount to win a certain package of time slots, customers may find it useful to bid on other packages which are generated by the user agents in the supplementary stage. In this section, we propose two sets of automatically generated packages by user agents.

The first set of automatically generated packages rely on the bidding strategy selected by a customer in the clock stage. For example, for customers who select the first bidding strategy, the user agent may include all must-have trip time slots while generating other packages which also include several optional trips. For the second strategy, we propose to use a package generation technique similar to the internal-based strategy presented in An et al. (2005), by adding items with the highest utility first while considering the budget constraints. For the third bidding strategy, we propose selecting items that were chosen most frequently in the clock stage by the user agent, since these items tend to have lower prices.

The second group of packages are generated based on the day and the time of the trips submitted by bidders. These packages are common among all bidding strategies. We propose



(a) A diverse package (b) A consistent package (c) A single-day package

Figure 2.4: Diverse, consistent and single-day packages

generating *diverse, consistent,* and *single-day* packages. Diverse packages include time-slots from different days. Consistent packages include the trips in different days which take place in the exact same time. Single-day packages include only the trips within the same day.

To understand the package generation procedure for the second group, let us look at the trips submitted by a customer given in Table 2.2. Figure 2.4 shows graphs used for generating packages, where nodes represent time slots for submitted trips. For instance, Figure 2.4a visualizes a diverse package, where time slots from different days have been connected with edges while time slots from the same days do not. We use maximum cliques to generate the second group of packages. Note that in graph theory a clique is a complete subgraph of a given graph. A maximum clique is a clique with the maximum number of vertices. By looking for a maximum clique in the graph, user agents generate a package, which consists of all time slots within the clique. Another example is given in Figure 2.4b, which displays a network for a consistent package. In this case, we connect nodes (time slots) with similar start and end time of trips such as the first trips on Monday, Tuesday and Wednesday. Then user agents again look for a maximum clique in the graph and select time slots within the clique for a consistent package. Lastly, in the single-day package, we connect time slots from the same day as shown in Figure 2.4c. The summary of all automatically generated package types can be found in Table 2.3.

Table 2.3: Summary of all types of packages.

Packages	Strategy 1	Strategy 2	Strategy 3
Must-have trips only	✓		
Must-have trips plus optional	✓		
Highest utility trips		✓	
Frequently chosen trips			✓
Diverse package	✓	✓	✓
Consistent package	✓	✓	✓
Single-day package	✓	✓	✓

2.5 Numerical Experiments and Case Studies

The goal of the numerical experiments is two-fold. First, we investigate the efficiency of the proposed auction elements such as activity rules, the ask price algorithm and pricing rules. Second, we discuss the auction outcomes to bidders and an auctioneer in terms of winning bids and generated revenue.

For the numerical studies, we build a simulation module based on 2010–2012 California Household Travel Survey, which indicates start and end time of trips as well as miles for 2908 vehicles for one week. Based on this data, we generate bids for customers. In practical, we remove trips with a length less than 15 minutes from the data. Then each vehicle represents a customer with the respective trips information out of which we generated bids as discussed in Section 4. The location distances of each pair of customers are randomly selected between 0 and 60 minutes. Further, we use Caltrans Traffic Counts data, namely Annual Average Daily Traffic 2016, for assigning required eligibility points for each hour of a week. All instances are run 3 times using seeds 1, 3 and 5. For all our experiments we set the ask price increment $\epsilon_p = 2$.

We use the Julia Language (Bezanson et al., 2012) to implement the simulation model and CPLEX 12.5 to solve integer programs via the JuMP.jl package (Dunning et al., 2017). We also use the heuristic algorithm by Takalloo et al. to solve WDPs not only to determine the winners, but also solve sub-problems arising in various payment rules.

The simulation model includes three modules that have different functionalities; namely the general module, the auctioneer module, and the user agent module.

The general module in the simulation aims to generate trips, bidders, bids and etc. The general module takes as input the number of bidders and the percentage of bidders with a particular bidding strategy. Based on the given input, the module generates a set of bidders and assigns them their initial eligibility points, budget constraints, selected trips, including their start time, end time and miles.

The auctioneer module performs all the auctioneer's tasks such as calculating ask prices in the clock stage, enforcing activity rules, solving WDP and calculating payments. This module takes as an input the fleet size to be auctioned and price increment for ask prices in the clock stage in the absence of supply-demand balance.

The user agent module performs all the user agent's tasks. As discussed in Section 2.4, user agents bid on behalf of bidders both in the clock and the supplementary stages. User agents also generate packages for bidders. This module takes as an input the bidding strategies of the bidders.

2.5.1 Managerial Insights

To compare the different bidding strategies offered by user agents and check the efficiency of the proposed activity rules, we conduct experiments under 4 different scenarios discussed below. In each scenario, we simulate an auction with 21 bidders and a single AV. We let a bidder select strategy 1, 2, or 3 for the clock stage and consider the following scenarios for the rest of bidders:

- *Scenario 1*: The rest of bidders select Strategy 1.
- *Scenario 2*: The rest of bidders select Strategy 2.
- *Scenario 3*: The rest of bidders select Strategy 3.
- *Scenario 4*: The rest of bidders are split to select Strategy 1, 2, or 3.

Table 2.4: The strategies comparison with Activity Rules

	The total wins			The percentage of full wins		
Scenario	Str1	Str2	Str3	Str1	Str2	Str3
1	55	44	47	57.69%	19.23%	23.08%
2	62	46	51	48.78%	21.95%	29.27%
3	59	51	52	42.86%	28.57%	28.57%
4	58	48	50	44.44%	25.00%	30.56%

Table 2.5: The strategies comparison without Activity Rules

	The total wins			The percentage of full wins		
Scenario	Str1	Str2	Str3	Str1	Str2	Str3
1	32	36	42	23.08%	38.46%	38.46%
2	34	48	47	14.29%	47.62%	38.10%
3	27	38	50	5.88%	23.53%	70.59%
4	43	42	51	28.57%	32.14%	39.29%

We run the simulation for 3 instances for each class (strategy-scenario combination) and count the number of times (in 3 instances) a bidder has been selected as a winner under different scenarios and strategies.

Table 2.4 reports the total number of times bidders win (in 3 instances) using different strategies and under 4 scenarios and the percentage of full wins (when a bidder wins in all 3 runs) when the proposed eligibility based activity rules are used. We observe that the greatest number of wins is achieved under Strategy 1 regardless of any configuration of scenarios. Similarly, bidders who select Strategy 1 are more likely be selected as a full winners under any scenario followed by Strategies 3 and 2.

From computational studies, we may conclude that the activity rules effectively induce the first bidding strategy. When the bidders with the third bidding strategy (who seeks to bid for items with the lowest ask prices) enter the supplementary stage, they face a relative cap on their bidding price induced by the supplementary stage activity rules.

Thus, submitting competitive bids under Strategy 3 is relatively harder compared to bidding under the first strategy which is not restricted by activity rules. Indeed, under the first strategy bidders consistently bid for must-have trips and gradually increase bid prices, which is a favored behavior according to activity rules. This explains why a customer with Strategy 1 tends to be a winner. Similarly, since under Strategies 2 and 3, item selection is based on the value of ask prices, when entered into the competition for conflicting time slots, customers with such bidding strategies most likely to choose less competitive items. Then a bidder with the first bidding strategy encounters less competition with the bidders with the second or the third bidding strategy, which increases her chance of winning.

We also run similar experiments in the absence of the proposed activity rules to investigate the auction outcomes under different strategies and scenarios. In particular, we removed eligibility constraints in selecting bids presented in (2.13) for Strategies 1 and 2 and eliminated the objective function of Strategy 1 shown in (2.12). Also, we removed the relative caps imposed by (2.1). Table 2.5 presents the results, which clearly indicate the prevalence of Strategy 3 in the number of wins and in the percentage of full wins, while Strategy 1 results in the lowest number of wins in almost all scenarios. These experiment results clearly indicates the importance and efficiency of the proposed activity rules to prevent strategic bidding.

As discussed before, in order to select payment rules for the proposed auction, we used the simulation model. In particular, we measure the generated revenue and the computation time to compare VCG, core-selecting and proxy payments. We run the simulation model multiple times by generating instances based on the different number of bidders, vehicle numbers and bidding strategies in the clock stage while using the exact and heuristic approaches in solving the WDP. We implement proxy-auction using safe start with VCG prices (Hoffman et al., 2006). We also implement quadratic core-selecting payments suggested by Day and Cramton (2012).

Table 2.6 reports the results when the WDP is solved exactly and $|\mathcal{H}| = 1$. In this case, core-selecting payments generate the highest revenue under all bidding strategies, while proxy payments also demonstrate competitive revenues. We have to note that for proxy payments

Table 2.6: Payments rules comparison using exact solutions

		Revenue, $			Time, sec		
$\|\mathcal{J}\|$	Bidders' Strategy	VCG	Core	Proxy	VCG	Core	Proxy
30	Strategy 1	1,017	4,850	2,960	9.89	10.96	112.55
	Strategy 2	889	1,193	1,055	85.83	91.39	387.10
	Strategy 3	1,090	2,063	1,728	102.58	108.73	5,404.54
	Mixed	779	2,930	1,469	63.87	67.31	286.86
60	Strategy 1	3,370	7,189	4,858	57.95	60.24	193.32
	Strategy 2	1,478	2,267	2,012	457.23	471.25	656.90
	Strategy 3	2,429	4,121	3,666	682.35	703.44	11,695.41
	Mixed	1,970	4,582	3,052	247.25	254.72	662.30
90	Strategy 1	4,308	8,940	6,405	150.66	155.08	827.98
	Strategy 2	2,191	2,954	2,711	17,573.46	17,898.58	15,056.72
	Strategy 3	3,776	5,767	5,304	7,468.69	7,641.55	14,730.09
	Mixed	3,003	6,172	4,313	751.17	769.80	5,875.49

calculation, we place a time limit of 4 hours and report the revenues generated within the time limit. Nevertheless, the computation time of the proxy payment method is significantly larger than those of other payment methods, while core-selecting payments dominate both in terms of revenue and calculation time.

Table 2.7 reports the revenue and the computation time when the WDP is solved using the heuristic method discussed in Section 2.2.3.1 for $|\mathcal{H}| = 2$ and $|\mathcal{H}| = 3$. We observe that both under the exact and heuristic solutions the VCG payments dominate in terms of computation time. When the WDP is solved using the heuristic approach, core-selecting payments generate higher revenues compared to other payments except when $|\mathcal{H}| = 3$ and $|\mathcal{J}| = 120$ with the proxy payments being the highest. Since the WDP is solved using the heuristic solution, the VCG payments may violate individual rationality, thus resulting in higher payments compared to core-selecting payments which enforce individual rationality. Consequently, the warm start of proxy payments with the VCG payments results in larger proxy payments compared to core-selecting payments. We place a time limit of 4 hours for the proxy payments calculation and took the average run time for 3 replicates. Even though

Table 2.7: Payments rules comparison using heuristic solutions under mixed strategy

		Revenue, $			Time, sec		
$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{J}\rvert$	VCG	Core	Proxy	VCG	Core	Proxy
2	90	2,756	8,771	6,564	1,315	1,335	3,064
	120	4,904	11,018	8,805	3,038	3,077	10,986
3	90	170	10,069	6,694	1,649	1,671	10,561
	120	17,496	12,948	19,986	4,486	4,533	6,168

the time limit may cause lower revenues under the proxy payments, in general, we conclude that core-selecting payments generate the highest payments using the heuristic solutions. Therefore, based on the computational study we recommend using core-selecting payments as a pricing scheme for fractional ownership of AVs.

We also study the auction outcomes to bidders measuring a number of co-leasers defined as customers sharing the same AV and some statistics of the winning bids. For instance, average values of payments, miles, time slots' lengths and the number of trips are calculated by summing their respective values for the winning bids and dividing by the total number of winners. The number of bids indicates the average number of bids submitted by all winners. We report the results of 3 replicates for each instance taken as their average values. As shown in Table 2.8, the average payments increase with the increase in the number of participating bidders due to the rise in the competition. From the used dataset containing trips of customers, the average number of trips in the winning bids is above 2. We also note that the average number of trips in the winning bids change slightly across all instances. Such outcomes are related to the structure of the bids. For instance, bids with a large number of trips encounter a significant number of conflicts with other bids, thus, requiring large bid prices to win. In contrast, bids with a small number of trips have fewer chances to overlap with other bids. Then when solving the WDP selecting a large pool of bids with a small number of trips may contribute more to the social welfare compared to accepting a small pool of bids with a large number of trips.We also note that the trip costs per mile and per minute increase with the increased number of bidders. We may also enforce the minimum

Table 2.8: The auction outcomes

		Average Value						
$\lvert\mathcal{H}\rvert$	$\lvert\mathcal{J}\rvert$	# co-leasers	Payments	Miles	Time, min	# trips	# bids	WDP sol.
1	30	23.30	38.76	39.65	74.46	2.88	19.38	exact
1	45	31.33	47.91	30.33	60.66	2.49	17.01	exact
2	60	26.33	22.56	33.91	69.36	2.82	15.85	heuristic
2	90	34.67	69.80	29.04	58.93	2.45	16.15	heuristic

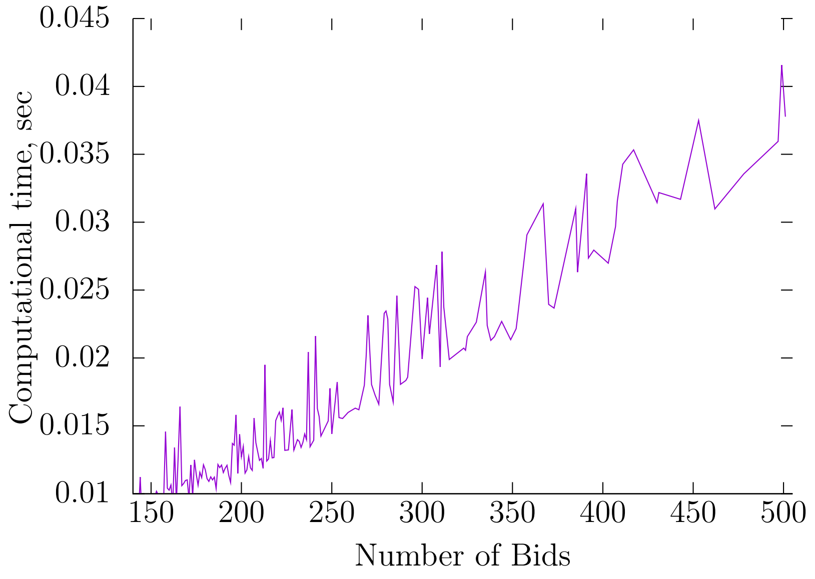


Figure 2.5: The performance of the ask price algorithm

trip length requirement in the WDP. For instance, the auctioneer may require customers to use AVs at least for 30 minutes per use. This may reduce the revenue of the auctioneer as it limits options for serving small trips and increases the idle time of AVs. However, it also offers more flexibility to winners in their schedule offering a wide time window between serving two customers and making such an option more attractive. In return, such a setting actually may boost the revenue.

Finally, to demonstrate the performance of the proposed ask price algorithm, we run the algorithm under various bid numbers, while measuring the computational time. As has shown in Figure 2.5, the algorithm solves all instances in a fraction of a second. Also, the computational time increases linearly with the number of bids, suggesting $O(n)$ complexity.

Chapter 3: A Reinforcement Learning Approach for Rebalancing Electric Vehicle Sharing Systems

The copyright permissions for reuse previously published material in this chapter can be found in Appendix B.

3.1 Introduction

The advent of electric vehicles (EVs) and car-sharing services provides a sustainable option to move people and goods across dense urban areas. Car sharing services with EVs have the potential to increase the utilization of resources and offer a unique opportunity to the urban population in the form of free-floating EV sharing systems (FFEVSS). With the FFEVSS, examples of which include companies such as car2go car2go (2021b) and WeShare Volkswagen (2021), customers no longer need to own a vehicle and can conveniently pick up/drop off any EV, on-demand, from the parking lots of designated service areas. However, there are some critical operational challenges to bring this on-demand service into the mainstream.

Before the start of the day, an operating company needs to relocate EVs to the ideal demand locations to establish a supply-demand balance in the system. Furthermore, to provide a certain level of service, EVs need to be charged before they can be used by the customers. There are two major issues: (i) there exists a sparse demand in the service area network, and hence it is not trivial to find the ideal locations to relocate the EVs; and (ii) there needs to be an efficient routing plan to drop off the drivers for picking up the EVs and taking the EVs to the charging stations for charging, and then pick up the drivers from their respective locations car2go (2021a). It is evident that without efficient solutions for

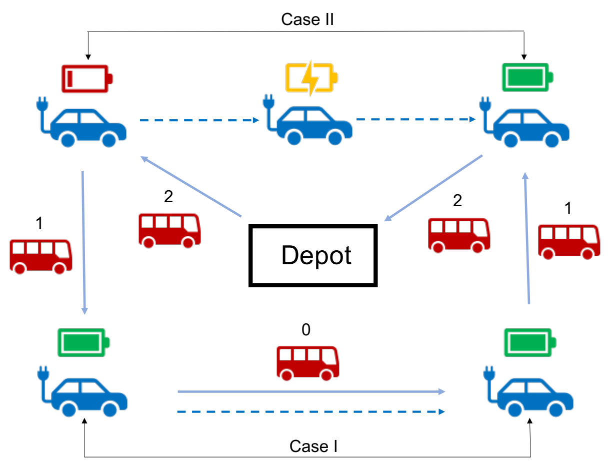


Figure 3.1: The overview of the rebalancing problem of FFEVSS, with a single shuttle and 2 drivers. The numbers indicate the number of drivers in the shuttle. Solid and dashed lines represent the routes of the shuttle and EVs, respectively. Cases I and II refer to relocation of EVs without and with charging, respectively.

the above complex and costly operational challenges Chianese et al. (2017), the sustainable existence of the FFEVSS is uncertain.

We consider a static, nightly rebalancing problem similar to Kypriadis et al. (2018); Santos et al. (2017); Folkestad et al. (2020); Haider et al. (2019), where a group of drivers is used to relocate and recharge the EVs based on the predicted demand for the next day, assuming the utilization level of FFEVSS is minimal. As shown in Figure 3.1, shuttles are used to support the movements of drivers. In this setting, rebalancing operations require two key decisions to be made: (i) how to route shuttles to pick up and drop off the drivers (shuttle routing decision) and (ii) where to charge and relocate each of the EVs (EV relocation decision). In this paper, focusing on solving the shuttle routing decision problem, we propose a reinforcement learning (RL) approach, in which the EV relocation decisions are made by a rule-based approach.

The proposed RL approach possesses several advantages compared to optimization-based approaches. First, unlike solutions coming from the static optimization techniques such as Folkestad et al. (2020); Haider et al. (2019), which need to be re-solved each time an input changes, the RL agent learns robust solutions that can be applied to any input coming from

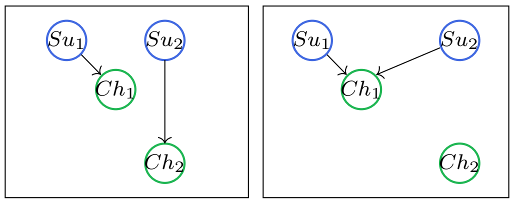


Figure 3.2: Assign supplier-charger pairs or reuse charger nodes? Ch and Su denotes charger and supplier nodes respectively.

the same distribution Bello et al. (2016). Second, while static optimization approaches can take significant time to solve a problem, a trained RL agent can be invoked to produce quality solutions instantaneously. Third, many practical considerations can be flexibly incorporated within the simulator in the training phase.

The shuttle routing to rebalance FFEVSS with its variety of trade-offs is not a trivial problem. For instance, as depicted in Figure 3.2, one may allow or disallow the reuse of charging stations in the derivation of solutions. The former choice offers more flexibility, but it also increases the complexity of exploring solutions. Therefore, the existing methods do not allow the reuse of the charging stations Haider et al. (2019). On the other hand, such a choice results in opportunity loss.

Another trade-off is depicted in Figure 3.3, where the first supplier node has an EV that needs to be recharged while the second supplier has an EV with a sufficient charging level. Then one needs to balance between traveling time and waiting time when routing a shuttle to supplier nodes. The complexity of such routing decisions increases with the network size, the network structure, and the number of shuttles and drivers deployed. Hence, it may not be possible to explore potential solutions with human-driven heuristics efficiently. With the proven ability of neural networks in recognizing patterns in graph-based representations, the utilization of neural network architecture with the proposed RL approach will provide better approximations and assist in obtaining efficient solutions that can be generalized.

In recent years, there has been a surge of studies that apply reinforcement learning to solve various traditional vehicle routing problems (VRPs) Nazari et al. (2018); Kalakanti et al. (2019); James et al. (2019) with capacity constraints, time windows, or stochastic demand.

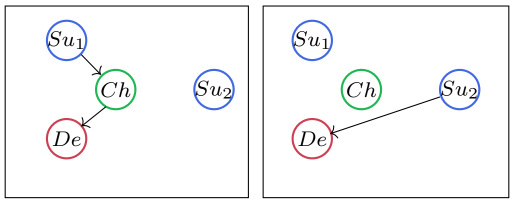


Figure 3.3: How to balance traveling time and waiting time trade-off? De, Ch and Su denotes demander, charger and supplier nodes respectively.

$$\begin{tikzcd}
S_0 \arrow[r, "a_0 : \mathrm{SR}_0"] & S_1 \arrow[r, "a_1 : \mathrm{SR}_1", "\mathrm{EVR}_0"'] & S_2 \arrow[r, "a_2 : \mathrm{SR}_2", "\mathrm{EVR}_1"'] & {}
\end{tikzcd}$$

Figure 3.4: State transitions: SR_i - shuttle routing decisions, EVR_i - EV relocation decisions, a_i - selected action

The shuttle routing problem, taken under this study, possesses significant differences with traditional VRPs. First, in a VRP setting, nodes to visit (demand) are typically independent of the routing decisions. However, in the shuttle routing problem, the locations of drivers to be picked up are determined by preceding routing decisions. This highlights a *strong interdependence* between demand and routing. Second, unlike VRP, the shuttle routing problem is characterized by *delayed rewards*. As shown in Figure 3.4, the actual relocations of EVs from a node happen after the execution of shuttle routing to the node. As a result, we observe delayed rewards with respect to the shuttle routing decision only after EVs reach their designated nodes. Such differences require a new approach to finding solutions for the shuttle routing problem.

We consider two settings of rebalancing FFEVSS. In the first setting, we focus on a single shuttle problem, where we train a single agent to learn routing policies. In the second setting, we aim to train a fleet of shuttles through single-agent reinforcement learning, where a central controller is responsible for routing multiple shuttles. In both cases, we deploy policy gradient methods along with recurrent neural networks for training. The shuttle routing problem under both of the above-mentioned settings possesses significant challenges that prohibit the direct use of the existing solution methods. For instance, in routing a single shuttle, we must train an agent not only to find efficient routes, but at the same time maintain the feasibility of the solutions related to the precedence of the visiting nodes. As for routing the fleet of shuttles, we must promote learning policies to route multiple shuttles that will contribute to a common goal.

The main contributions of this study are as follows. First, to the best of our knowledge, this study is the first to present an RL-based approach for handling *multiple* vehicles *explicitly* in the context of VRPs, while focusing on the shuttle routing problem for rebalancing the FFEVSS. Second, within the RL framework, we propose the utilization of deep neural network architecture to process the complex and high dimensional observations from an urban service area network to help train the RL agent in its decision-making. In particular, we adopt sequence-to-sequence models with an attention mechanism to fit the unique challenges of the rebalancing FFEVSS. Third, we present a novel training algorithm to route efficiently a fleet of shuttles to rebalance FFEVSS by utilizing policy gradient methods. Our training algorithm does not require splitting an urban network into sub-clusters for each shuttle, but instead allows developing policies that efficiently utilize shuttles and drivers in a whole network. Fourth, we develop a simulator to mimic real-world FFEVSS, which serves as the environment for training an RL-agent and allows efficient exploration of joint actions of multiple shuttles.

Unlike the solutions obtained using the methods from the literature, the empirical results obtained from this study show that the proposed method allows solving the general version of the problem with no restrictions on the urban network structure and charging levels of EVs. The learned policies offer a wide range of flexibility, resulting in a significant reduction in the time needed to rebalance the network.

The remainder of the paper will proceed as follows. In Section 3.2 we provide an overview of relevant literature and outline the unique challenges of the rebalancing FFEVSS. In Section 3.3 we present the problem formulation. In Section 3.4 we introduce the proposed reinforcement learning model. In Section 3.5 we demonstrate the results of our computational studies.

3.2 Related Work

Even though the problem of rebalancing FFEVSS has been recognized as essential for their sustainable existence in the literature Schulte and Voß (2015); Herrmann et al. (2014), most of the studies focus on high-level approaches to address the issue. One category of studies falls on incentive-based methods that aim to rebalance the system through influencing customer behavior Weikl and Bogenberger (2013). Another set of papers study the deployment of personnel and offer rule-based high-level decision-making frameworks Weikl and Bogenberger (2015); Zhao et al. (2018). There are only a few studies that specifically focus on the shuttle routing problem to rebalance FFEVSS, thus offering detailed solutions for day-to-day operational challenges.

One of such studies is Folkestad et al. (2020), which aims to solve both EV relocation and shuttle routing problems jointly. However, the proposed model does not enforce relocation of EVs directly to demander nodes, but indeed permits leaving EVs in charger nodes. As a result, charger stations will be blocked and cannot be reused, requiring the postponing of charging for the remaining set of EVs. Similarly, a recent study Haider et al. (2019) presents novel approaches in addressing EV relocation and shuttle routing problems simultaneously. Even though the study aims at relocating EVs directly to demander nodes, it assumes the abundance of charger stations in an urban network. Thus, again reusing charger stations is not considered, and the postponement of charging for EVs requiring it is allowed. Since charging infrastructure is often limited He et al. (2020), the reuse of charging stations must be an integral part of solutions to rebalance FFEVSS in real-world urban networks.

Recently reinforcement learning approaches gained popularity to solve various problems in transportation, including fleet management and rebalancing in ride-hailing services Shi et al. (2019); Lin et al. (2018); Wen et al. (2017); Sadeghianpourhamami et al. (2018). However, none of the existing studies focus on FFEVSS specifically and do not address the unique issue of charging and relocation together. For solving VRPs, deep reinforcement learning has been first applied in Nazari et al. (2018), which utilizes sequence-to-sequence methods Sutskever et al. (2014) and an attention mechanism Vinyals et al. (2015a). Later Kool et al. (2018) adopted the transformer model Vaswani et al. (2017) to solve VRPs without recurrent neural networks. James et al. (2019) proposes a novel model to solve online VRPs by utilizing neural combinatorial optimization and deep reinforcement learning. Similarly, Zhao et al. (2020) presents a hybrid model that combines local search with an attention mechanism. However, these studies focus on routing a single capacitated vehicle, where the main goal is to minimize the distance traveled. While multiple loops of a single capacitated vehicle can be interpreted as multiple vehicles, this paper is the first to present explicit modeling of multiple vehicles within an RL framework.

Although this study also adopts sequence-to-sequence models with an attention mechanism similar to Nazari et al. (2018), the significant differences in the nature of the rebalancing FFEVSS problem and VRP dictate the development of novel solution techniques. For instance, in the given problem, shuttles need to leave a depot, drop off, pick up drivers who relocate EVs, and return to the depot, highlighting two sets of constraints. First, the precedence of visited nodes needs to be maintained when charging stations are visited after nodes with EVs and nodes that require EVs are visited after either charging stations or nodes with EVs. Second, the capacity constraint must be satisfied when nodes with EVs are visited only when there is a driver in a shuttle and nodes with drivers are visited only if there is seating available for a driver in the shuttle. In addition to feasibility constraints, since both charging and relocations of EVs are involved in the shuttle routing problem, only considering factors that affect the total distance traveled is not sufficient. Moreover, the dynamics of an urban network due to routing a shuttle is more complex compared to the VRP due to the delayed movements of EVs relocation. Also, routing multiple shuttles requires a novel training algorithm. In particular, when several shuttles are present in an urban network and each of their movement influence the state of the network, we need a novel framework that enables the application of reinforcement learning tools based on Markov Decision Process (MDP).

3.3 Problem Statement and Formulations

3.3.1 Network

Let us consider a network $\mathcal{N}$ consisting of N number of nodes and a depot. We define a node as a supplier if it has an excess EV and a demander if it requires an EV. The network also has charger nodes. Each node in the network can store at most one EV. Depending on the charging levels of EVs there are two possibilities of the EVs relocation. In Case I, EVs are relocated from supplier nodes directly to demander nodes. In Case II, EVs first need to be taken to charger nodes, and after charging is complete, they need to be relocated to the demander nodes, as shown in Figure 3.1. We consider discrete charging levels of EVs, where a threshold-based rule is applied to decide whether to charge an EV or not. Also, a driver may wait at a charging station until an EV is fully charged or may head for the next activity. We consider two settings of the problem when a single shuttle or a fleet of shuttles is deployed for rebalancing the system. We formulate the routing problem for a single shuttle as MDP and utilize a central controller to route a fleet of shuttles.

3.3.2 Multi-shuttle Routing as MDP

Even though it is possible to formulate the routing of a fleet of shuttles using a multi-agent reinforcement learning framework, such an approach suffers from several drawbacks. Firstly, in the presence of several shuttles, each of which is treated as an autonomous agent, the stationary assumption of MDP is no longer valid Buşoniu et al. (2010). Therefore, a multi-agent reinforcement learning framework works under partially observable MDP Lowe et al. (2017); Gupta et al. (2017); Foerster et al. (2018), when each agent can observe only a local view of the network Oliehoek et al. (2016). Then each agent can only visit nodes visible from its local view, which imposes significant restrictions on developing an efficient routing.

Secondly, it is challenging to train autonomous agents without making strong assumptions about constant communication between agents. For instance, if at the current time step one agent selects a node to visit, then such information must be shared among other agents to avoid the presence of several agents at the same node. Lastly, under a static network, when the state of the network is constant and well-known, a centralized approach will help navigate a fleet of shuttles efficiently. Therefore, we formulate routing multiple shuttles to rebalance FFEVSS using a central controller responsible for making routing decisions of all shuttles. Then, we can formulate the problem using a single-agent reinforcement learning framework and MDP. We also note that the concept of multi-agent reinforcement learning and central controller is similar to decentralized control and centralized control in the transportation literature.

A fleet of shuttles with drivers leaves a depot and visits nodes in the network to relocate EVs from supplier nodes to demander nodes. Shuttles must return to a depot after fulfilling demand at all demander nodes and picking up all the drivers. These sequential decisions of a central controller for routing shuttles under uncertain demand (locations of drivers) can be formulated as a finite horizon MDP, where the future dynamics of the system depend only on the current state. We define the RL framework for the problem as tuple $\mathcal{M} = \langle X, \mathcal{A}, P, R, T \rangle$ representing states, actions, transition probabilities, reward function, and time horizon, respectively. The definitions are as follows:

- $\mathcal{I} = \{1, ..., I\}$ is the set of I shuttles that are controlled by a central controller;
- State set X represents the network, where for each node it shows its location, the relative distance, the number of EVs, the number of drivers, the charging levels of EVs' and indicators for the expected transitions. We utilize binary vectors to indicate if there is an expected EV coming to a node. We denote state as x_t at time t.
- $\mathcal{A}$ is the set of joint actions such that $\mathcal{A}_t = \mathcal{A}_t^1 \times \mathcal{A}_t^2 \times \cdots \times \mathcal{A}_t^I$, where $\mathcal{A}_t^i$ is the action set of shuttle i at time t and action a_t^i indicates a node number to be visited next by

Table 3.1: A summary of variables.

$\mathcal{I}$	the set of shuttles	$\mathcal{A}$	the set of joint actions
X	the state of network	t_c	current clock time
τ	traveling time	w	waiting time
T	the max number of time steps	r	immediate reward
R	total reward	θ_a	parameters of actor
θ_c	parameters of critic	π	routing policy
Y	the set of visited nodes	n	node in network

shuttle i. Then a central controller's action set consists of joint actions of all shuttles, $\mathcal{A}_t$, at time t.

- Transition Probabilities, P, determines state transitions probabilities $p(x_{t+1}|x_t, a_t)$ at time t with respect to taken action a_t. In the given problem, transitions are deterministic but often delayed. After an action is taken, the relocations of EVs are scheduled. However, the actual state transitions related to the movements of EVs occur later, as shown in Figure 3.4.

- All shuttles share a common reward R and immediate reward r_t, which are assigned based on the joint actions of all shuttles at time t denoted by a_t and state x_t;

- Instead of defining the specific time value of T, we define one episode rollout for the problem based on the experiment outcomes. One episode is terminated either if all demander nodes are fulfilled and all drives are picked up back to a depot or if the total number of time steps exceeds the predefined maximum time steps, the value of which is set based on the size of a network.

- Each time step t is determined by the earliest fulfilled action among all shuttles. Thus, each time step starts when a central controller takes an action and finishes whenever any action is fully executed.

The list of variables used can be found in Table 3.1.

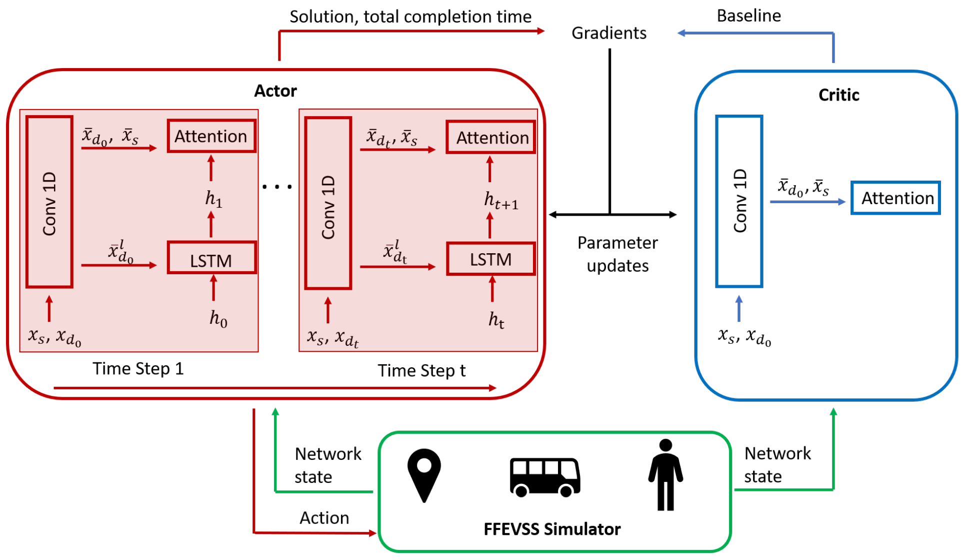


Figure 3.5: An overview of the reinforcement learning model.

3.4 Reinforcement Learning Model

We adopt a policy gradient method, similar to those popularly used in routing problems Nazari et al. (2018); Kool et al. (2018); James et al. (2019), to learn the complex routing policies of shuttles *directly*. In general, policy gradient methods consist of two separate networks: an actor and a critic. The critic estimates a value function given a state according to which the actor's parameters are set to generate policies in the direction of improvement. We train an agent and a central controller to route a single shuttle and multiple shuttles in an urban network by simulating the FFEVSS environment. The simulator is developed to handle EV relocations through rule-based decisions and utilizing sequence-to-sequence models to generate policies. The overview of the model is shown in Figure 3.5.

3.4.1 The FFEVSS Simulator

The main function of the FFEVSS simulator is to represent the dynamics in an urban network caused by movements of shuttles. There are immediate and delayed transitions related to routing shuttles. In an immediate update to the environment at each time step, we consider locations of shuttles, drivers, EVs, the number of drivers in a shuttle, and fulfillment of scheduled transitions either related to charging or relocation of EVs. Also, at each time step, we schedule transitions related to movements of EVs that have started but unfulfilled. In particular, starting at the current clock time $t_c = 0$, we update the environment according to movements of a shuttle:

$$t_c \leftarrow \begin{cases} t_c + \tau(n_{t-1}, n_t) & \text{if } n_{t-1} \neq n_t \\ t_c + w_t & \text{if } n_{t-1} = n_t \end{cases}$$

where τ represents traveling time between nodes visited by a shuttle at time $t-1$ and t and w_t denotes waiting time at node n. We define waiting time at node n as the difference between the time when a delayed transition at node n occurs and the time when a shuttle reaches node n. To account for delayed transitions, we introduce a time vector, which keeps track of remaining times until either EVs arrive at designated nodes or their charging completes. In the case of multiple shuttles, the environment is updated with the earliest movements of shuttles.

Another function of the FFEVSS simulator is to update a masking scheme according to the current state of the urban network. The masking scheme helps to maintain the feasibility of solutions related to the precedence of visited nodes and the number of drivers in a shuttle. Also, having an efficient masking scheme expedites the exploration of action space. We deploy the following masking scheme, where $\mathcal{A}_t = \emptyset$ stores the set of available nodes/actions to visit at time t and the rest of the nodes are masked. For each $n \in \mathcal{N}$, we update:

$$\mathcal{A}_t \leftarrow \begin{cases} \mathcal{A}_t \cup \{n\} & \text{if } l_t > 0 \quad \text{and } n \in \mathcal{D}_t \cup \mathcal{E}_t \\ \mathcal{A}_t \cup \{n\} & \text{if } l_t = 0 \quad \text{and } n \in \mathcal{D}_t \end{cases}$$

Here set $\mathcal{E}_t$ denotes nodes with an EV or nodes with the expected EV due to delayed transitions, set $\mathcal{D}_t$ denotes nodes with a driver or nodes with the expected drivers, and l_t denotes the number of drivers in a shuttle at time t.

3.4.2 EV Relocation Decisions

As described earlier, our focus in this study is to solve the shuttle routing problem. Hence, we are using a rule-based approach for EVs' relocation decisions. The rule-based approach is as follows: every time a supplier node with an EV has a driver, that EV is relocated to the nearest available either charger or demander node. The decision of whether to relocate an EV to a demander or charger node is predetermined in the settings of a simulator. We apply a threshold-based rule; that is, if the charging level of an EV exceeds the threshold, then it can be directly relocated to a demander node or must be charged, otherwise.

We maintain a binary vector in the simulator to indicate if a charger node is available or not. This representation helps in deciding the relocation of an EV from a supplier node to an available charger node. We determine the closest available charger node by multiplying the binary vector by a time matrix that indicates time to travel among any pair of nodes. To decide EVs' relocations from either supplier or charger nodes to demander nodes, we maintain a demand matrix that keeps track of demander nodes that still need an EV at time t. In particular, in the simulator, we store the time needed to move from all nodes to each demander node and increase those values to large numbers if a demander node is satisfied. Then, if an EV needs to be relocated to a demander node, we compute the minimum time from a node to the closest demander nodes.

3.4.3 A Sequence-to-sequence Model for the Shuttle Routing Problem

Motivated by Nazari et al. (2018), we propose using a sequence-to-sequence model for rebalancing FFEVSS, which typically consists of an encoder and a decoder. Given urban network $\mathcal{N}$, we aim to generate a sequence of nodes to be visited by either a shuttle or a fleet of shuttles. In other words, we are interested in learning the following conditional probability or parametrized policy π_θ:

$$\pi_\theta(Y_T|x_0) = \prod_{t=0}^{T-1} \phi(y_{t+1}|x_t, Y_t) \tag{3.1}$$

In (3.1), we let $x_t = \{x_t^1, \dots, x_t^N\}$, where x_t^n denotes static and dynamic states of node n at time t. Unlike in machine translation, the state of nodes in the network status changes dynamically with shuttles movement; thus, we need to consider both static and dynamic states for each node. Also, we let y_t denote a node to be visited at time t and $Y_t = \{y_1, \dots, y_t\}$ with $Y_0 = \emptyset$. Then to select a next node to visit y_{t+1}, we are interested in learning $\phi(y_{t+1}|x_t, Y_t)$.

However, a set of nodes in the network does not convey any sequential information. Therefore, it is common in literature Nazari et al. (2018), to omit recurrent neural network for encoding. Indeed, due to the sparse nature of networks, graph embedding is deployed in encoder to build their continuous vector representation as they suit better for statistical learning Perozzi et al. (2014). The following equation describes embedding for each $n \in \mathcal{N}$:

$$\overline{x}_s^n = b^s + W^s x_s^n \tag{3.2}$$

$$\overline{x}_{d_t}^n = b^d + W^d x_{d_t}^n \tag{3.3}$$

where, $\overline{x}_s^n$ and $\overline{x}_{d_t}^n$ are embedded static and dynamic states of node n at time t and b, W represent the trainable parameters of a neural network. We denote by $\overline{x}_t^n = (\overline{x}_s^n; \overline{x}_{d_t}^n)$ concatenation of embedded static and dynamic states of nodes.

For decoding we use recurrent neural networks (RNN), that takes static state of the last visited node and stores the sequence as follows:

$$h_t = W^h f(h_{t-1}) + W^x \overline{x}_s^n \tag{3.4}$$

where h_t is a memory state of RNN, f represents nonlinear transformation and x_s^n is a static state of node n visited at time t. Trainable weight matrices W^h and W^x represent connections between hidden state to hidden state and hidden state to an input respectively. Note in our implementations, we use a LSTM cell as RNN.

In addition to encoder and decoder, we also utilize content based attention mechanism as in Nazari et al. (2018). Content based attention tries to mimic associative memory and is designed to handle cases when an input to the sequence-to-sequence model is a set Vinyals et al. (2015a). In particular, the current state of an urban network is coupled with the memory state of RNNs about the sequence to calculate an alignment vector c_t that assigns the probabilities of nodes to visit next:

$$u_t^n = v \tanh(W(\overline{x}_t^n; h_t)) \qquad \forall n \in \mathcal{N} \tag{3.5}$$

$$c_t = \mathrm{softmax}(u_t) \tag{3.6}$$

where v and W are trainable weight matrices.

For the problem under study, we define the static state of nodes as their location coordinates and the initial charging levels of EVs at supplier nodes. Even though the charging levels of EVs will change as EVs are taken to charging stations, only their initial values determine charging times. Therefore, we consider them as a static state of nodes. For a dynamic representation of the states of nodes, we use the number of EVs, the number of drivers in a shuttle, and the distance from the current node to other nodes. Our experimental studies show that passing distance information as a dynamic state of nodes substantially reduces training time. Figures 3.5 summarizes the sequence-to-sequence model of the shuttle routing problem used in the actor network. In routing a fleet of shuttles, we also deploy a single actor network, where a sequence of visited nodes Y_t, includes nodes visited by all shuttles up to time t.

3.4.4 Reward Function

Reward function along with sets of available actions reflects our aim to maintain the feasibility and efficiency of routing decisions. Since the shuttle routing problem considers both charging and relocation of EVs, reward function must not only reflect traveling times between nodes, but also include waiting times. Therefore, we define reward function as the negative of total time spent in the system starting when a shuttle or a fleet of shuttles leaves a depot and ending when all shuttles are returned back to the depot with all drivers after fulfilling all demander nodes. Then our aim is to maximize the negative of total time spent in the system denoted by R. More formally we define reward function as follows, using immediate rewards r_t:

$$R = \sum_{t=1}^{T} r_t \tag{3.7}$$

where

$$r_t = \begin{cases} -\tau(n_{t-1}, n_t) & \text{if } n_{t-1} \neq n_t \\ -w_t & \text{if } n_{t-1} = n_t \end{cases}$$

and τ_t is traveling time and w_t is waiting time at time t.

3.4.5 Training Algorithm

In training, we are interested in finding policy parameters $\overline{\theta}$ that maximize the total expected reward:

$$\overline{\theta} = \arg\max_{\theta} \mathbb{E}_{\pi_\theta}[R]. \tag{3.8}$$

Given the state of network X, we can write:

$$J(\theta|x) = \mathbb{E}_{\pi \sim p_\theta(\cdot|x)}[R(\pi|x)] \tag{3.9}$$

and

$$\nabla_\theta J(\theta|x) = \mathbb{E}_\pi[A^\pi \nabla_\theta \log p_\theta(\pi|x)] \tag{3.10}$$

$$A^\pi = R(\pi|x) - V(x_0). \tag{3.11}$$

We use the REINFORCE algorithm with a baseline Williams (1992), which is the value of the initial state of an urban network estimated by a critic with trainable parameters θ_c. Algorithm 3 represents our training procedure, where the actor network with trainable parameters θ_a represents policy π. In a batch training setting, the batch of instances is generated by a data generator. Instances are passed through the simulator. Then, the actor network produces probabilities of nodes to be visited by shuttles at each time step, and the simulator is updated accordingly until the entire episode is finished. Then with the received total reward for the selected actions, the parameters of the actor and critic networks are updated. Unlike in the existing literature Haider et al. (2019), the algorithm does not require splitting an urban network into sub-clusters for each shuttle, but instead deploys all shuttles to serve the whole network. Also, utilizing a central controller that observes the entire urban network state along with the masking scheme in the simulator allows efficiently exploring joint action of all shuttles. For instance, if a node has been assigned to be visited by a shuttle, then that node is masked for other shuttles.

3.5 Computational Studies

3.5.1 Data Generation and Configurations

We consider a 1×1 square mile network consisting of demander, supplier, and charger nodes. We first specify the total number of nodes in the network and the number of demander and charger nodes. We sample x, y coordinate of each node from a uniform distribution with values ranging from 0 to 1. Similarly, we sample demander, charger, and supplier nodes from a uniform distribution. For each supplier node we set the initial charging levels of EVs

Algorithm 3: Training Algorithm

```
Input: Initialize network parameters $\theta^a$ and $\theta^c$ for actor and critic networks
       respectively. Set the maximum number of epochs, a batch size and the
       maximum number of steps denoted as $M_{\text{epochs}}$, $M_{\text{epis}}$ and $T$ respectively;
1  for epochs = 1 to $M_{epochs}$ do
2      Reset gradients $d\theta^a, d\theta^c$;
3      for $m$ = 1 to $M_{epis}$ do
4          data $\sim$ DataGenerator($\rho$);
5          $x_0^m$, $\mathcal{A}_0$ = simulator.reset(data);
6          Add $x_0^m$ to $X_0$, set $R^m = 0$, set L to $\mathcal{I}$;
7          for $t$=0 to $T$ do
8              foreach $i \in L$ do
9                  $a_t^i$, $p_t^i$ = actor network($x_t$, $\mathcal{A}_t^i$);
10                 Store $p_t^i$ in $p^m$, remove $a_t^i$ from $\mathcal{A}_t$;
11             $x_{t+1}$, $\mathcal{A}_{t+1}$, $r_t$, $t_c$ = simulator.step($a_t$);
12             Empty set $L$;
13             foreach $i \in \mathcal{I}$ do
14                 if $a_t^i$ is complete at $t_c$ then
15                     add $i$ to $L$
16                 else
17                     $a_{t+1}^i = a_t^i$ and remove $a_t^i$ from $\mathcal{A}_{t+1}$
18             $R^m = R^m + r_t$;
19         Calculate $V^m(x_0^m; \theta_c)$ using critic;
20     $d\theta_a = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} (R^m - V^m(x_0^m; \theta_c)) \nabla_{\theta^a} \log p^m$;
21     $d\theta_c = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} \nabla_{\theta_c} (R^m - V^m(x_0^m; \theta_c))^2$;
```

Table 3.2: Hyperparamter values

Conv1D, LSTM hidden dim	128	Conv1D kernel size	1
Critic, linear hidden dim	128	Learning rate actor, critic	10^{-4}

randomly between 1 and 5. We assume that EVs do not need charging and can be directly taken to demander nodes if their charging levels exceed 3. Otherwise, EVs first need to be taken to charger nodes, where all of them are charged until the charging level of 5 is reached. For each charging level, we assign the charging time equal to the average traveling time between all pairs of nodes in the network. We do not consider discharging rates in the movements of EVs, while we assume the constant velocity for EVs equal to 45 miles/hour.

Computational experiments are conducted with 2 Intel Xeon E5-2630 2.2 GHz 20-Core Processors, 32 GB RAM, and the Ubuntu 18.04.4 LTS operating system. All implementations are done in Python 3.7 using PyTorch 1.5. Our implementations of the critic network have similarities to the actor network structure except for the use of LSTM. We first embed the initial static state of the urban network using 1D convolution networks and then pass it to the attention mechanism. We pass the output of the attention mechanism through a sequence of feed-forward networks to obtain the final estimate for a value function. Table 3.2 represents the hyperparameters used for training, which are the same as in Nazari et al. (2018). We train RL agents on networks of various sizes and difficulty levels. For each problem class defined by the size of a network, we consider instances with three different levels of difficulty. Cases when there is an abundant presence of charging stations than the number of EVs requiring charging we call *easy* instances. Similarly, cases when there is an exact number of charging stations as the number of demander nodes we call *medium* difficulty instances. Finally, in cases when there is a less number of charging stations than the number of demander nodes, we call them *hard* instances. The descriptions of difficulty levels are found in Table 3.3.

Table 3.3: Difficulty levels description, where De, Ch, Su, and $Su^{'}$ denote the set of demanders, chargers, suppliers, and suppliers with EVs that require charging, respectively.

	Easy				Medium				Hard			
$\lvert\mathcal{N}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$
23	7	7	8	4	7	7	8	8	8	6	8	8
50	16	16	17	8	16	16	17	17	17	15	17	17
100	33	33	33	16	33	33	33	33	33	32	34	34

3.5.2 RL Agents and Benchmarks

We train three types of agents using the proposed RL models. The first agent denoted as *gen-RL* is trained on all three difficulty levels, but on a fixed network size. The second agent denoted as *net-RL* is trained on networks of various sizes, but it is tailored to a specific difficulty level. The last agent denoted as *RL* is trained on a fixed network size and on a specific difficulty level. For our computational studies, we consider a benchmark from Haider et al. (2019). The benchmark model denotes as *Sim* represents a joint model that solves the EVs relocation and the shuttle routing problems simultaneously. To solve multi-shuttle routing problems, the heuristic splits an urban network into some clusters and solves a single-shuttle routing problem for each cluster. However, there are some limitations to the method. One of them is related to the inflexibility of the solutions when drivers that have been dropped off from one shuttle cannot be picked up by other shuttles. Another disadvantage is related to charger nodes. The heuristic can only handle cases when the number of charger nodes is not less than the number of EVs that must be charged.

3.5.3 Results on Random Instances

Figure 3.6 shows training rewards for the multi-shuttle problems on the network with 23 nodes and 3 drivers. Overall, training time depends on the network size, its structure, and the features passed to the actor network. Using distance information from the current node

Table 3.4: Comparison of RL agents in terms of total time spent in the system, the average of 128 test instances are reported. In bold are the best results.

			Easy			Medium			Hard		
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	net-RL	gen-RL	RL	net-RL	gen-RL	RL	net-RL	gen-RL	RL
23	1	3	9.29	8.34	**7.70**	14.63	11.75	**10.27**	16.01	13.73	**12.32**
	2	3	6.01	5.79	**5.40**	7.93	8.45	**6.93**	8.89	9.00	**8.34**
	3	2	5.48	5.28	**5.21**	7.02	7.58	**6.38**	8.33	8.11	**7.79**
50	1	3	14.97	13.96	**13.77**	20.35	19.36	**17.93**	22.60	20.05	**18.92**
	2	3	8.54	**8.21**	8.41	11.81	**10.81**	11.23	12.15	11.76	**11.96**
	3	2	7.22	6.90	**6.89**	9.58	9.41	**9.23**	10.33	9.89	**9.77**
100	1	3	23.16	22.98	**22.18**	**30.62**	32.33	30.67	**31.53**	32.30	36.67
	2	3	**12.91**	14.25	12.92	17.55	18.42	**17.54**	**17.21**	18.79	17.90
	3	2	10.23	**10.21**	**10.21**	13.39	13.73	**13.33**	13.69	**13.67**	14.94

to other nodes in the actor network results in better rewards compared to when not passing such information.

To compare different RL agents' performances, we conduct experiments on various network sizes and the degree of difficulty of instances and measure the mean of the total time spent in the system out of 128 instances. Table 3.4 shows the experiments' results. In most instances, an RL agent trained on a specific size and a specific instance difficulty level tends to perform the best. We observe that net-RL agents, trained on various network sizes, tend to perform better on larger network sizes, while gen-RL agents, trained on various difficulty levels, can be competitive on medium-sized networks. As the network size increases, the results show that using net-RL and gen-RL agents can be beneficial. For the rest of the experiments, we use RL agents.

Table 3.5 illustrates the performance of the RL solutions with those of the heuristic optimization method, labeled Sim. The reinforcement learning approach can solve all instances of the problem, while the optimization method can handle only easy and medium cases. Moreover, for easy and medium cases measured in the mean of total time spent in the system, the RL solutions perform better than the heuristic optimization solutions. We also note that the derived RL solutions do not solve for optimal relocation of EVs and are only based

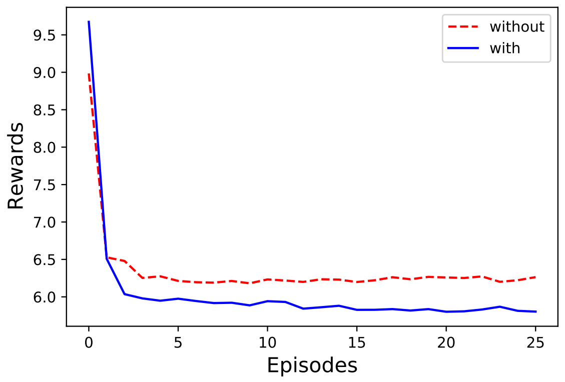


Figure 3.6: Training rewards with and without distance as an input

Table 3.5: RL model vs. the heuristic optimization in terms of total time spent in the system, the percentages of winning instances and computational time in seconds, the average of 128 test instances are reported. In bold are the best results.

			Easy					Med					Hard			
			Mean		Win %	Time, s		Mean		Win %	Time, s		Mean		Time, s	
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	Sim	RL	RL-Sim	Sim	RL	Sim	RL	RL-Sim	Sim	RL	Sim	RL	Sim	RL
23	1	3	8.81	**7.70**	85.94%	7.07	0.01	12.39	**10.27**	94.53%	13.98	0.02	–	12.32	–	0.02
	2	3	5.72	**5.40**	65.63%	1.61	0.04	7.43	**6.93**	73.44%	3.31	0.04	–	8.34	–	0.09
	3	2	5.27	**5.21**	51.56%	0.91	0.06	6.39	**6.38**	48.44%	1.71	0.05	–	7.79	–	0.11
50	1	3	17.34	**13.77**	96.09%	48.43	0.05	24.59	**17.93**	100.00%	98.61	0.05	–	18.92	–	0.06
	2	3	9.19	**8.41**	74.22%	11.62	0.21	12.25	**11.23**	73.44%	23.47	0.16	–	11.96	–	0.25
	3	2	6.96	**6.89**	53.13%	5.35	0.20	9.25	**9.23**	50.00%	10.75	0.21	–	9.77	–	0.35
100	1	3	34.20	**22.18**	100.00%	152.15	0.16	45.97	**30.67**	100.00%	599.36	0.17	–	36.67	–	0.26
	2	3	16.11	**12.92**	100.00%	66.13	0.44	21.63	**17.54**	96.09%	118.68	0.44	–	17.90	–	0.55
	3	2	11.71	**10.21**	86.72%	28.23	0.92	15.63	**13.33**	92.97%	53.86	1.03	–	14.94	–	1.02

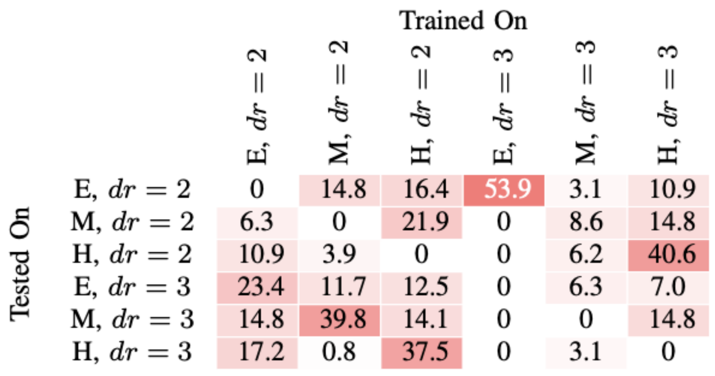
on predefined rules, while the optimization heuristic solves for both the shuttle routing and EV relocation problems.

Table 3.5 shows the performance comparison of Sim and RL models in terms of percentages of winning instances. For instance, in an RL-Sim pair comparison, the value of cells under the column indicates the percentages of instances when the RL model performed at least equally to Sim model out of 128 test instances. As shown in Table 3.5 the RL model performs better than the heuristic method in at least 50% of all instances, except one instance.

To show the generation of the instantaneous solutions using RL models, we measured computation time. Table 3.5 demonstrates the computation time it takes to derive a solution under Sim and RL models. We report an average time to solve an instance out of 128 instances in total. The difference in deriving solutions between Sim and RL models increases up to 585 times in the case of a single shuttle routing in a network with 100 nodes for Easy instances.

We also compare the effects of the number of drivers and difficulty levels on the trained models. In particular, we train models with a specific number of drivers on easy, medium, and hard instances on a fixed network size and check these models' performances against the models with varying a number of drivers and difficulty levels. For example, in Figure 3.7 rows indicate the problems' configurations in testing and columns indicate the problems' configurations in training datasets. The cells corresponding to a row and column show the percentages of instances when a trained model outperformed the model specifically trained for a test dataset. As we observe, models trained on specific difficulty levels tend to perform better on similar instances with a different number of drivers compared to on test models with the same number of drivers, but different difficulty levels.

The sample solution for a single-shuttle case, where 4 EVs in an urban network require charging, is shown in Figure 3.8. A shuttle with 3 drivers leaves the depot and visits supplier nodes first, followed by a charger node. By interchangeably visiting nodes thorough the



Tested On \ Trained On	E, $dr = 2$	M, $dr = 2$	H, $dr = 2$	E, $dr = 3$	M, $dr = 3$	H, $dr = 3$
E, $dr = 2$	0	14.8	16.4	53.9	3.1	10.9
M, $dr = 2$	6.3	0	21.9	0	8.6	14.8
H, $dr = 2$	10.9	3.9	0	0	6.2	40.6
E, $dr = 3$	23.4	11.7	12.5	0	6.3	7.0
M, $dr = 3$	14.8	39.8	14.1	0	0	14.8
H, $dr = 3$	17.2	0.8	37.5	0	3.1	0

Figure 3.7: The number of drivers vs. difficulty levels.

Table 3.6: The Amsterdam dataset structure and RL model vs. Sim on the dataset in terms of total time spent in the system.

			Weekdays		Weekends	
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	Sim	RL	Sim	RL
170	1	5	420.08	**416.23**	**411.59**	433.63
	2	3	207.63	**182.39**	224.88	**185.06**
	3	2	142.90	**136.52**	153.38	**142.20**

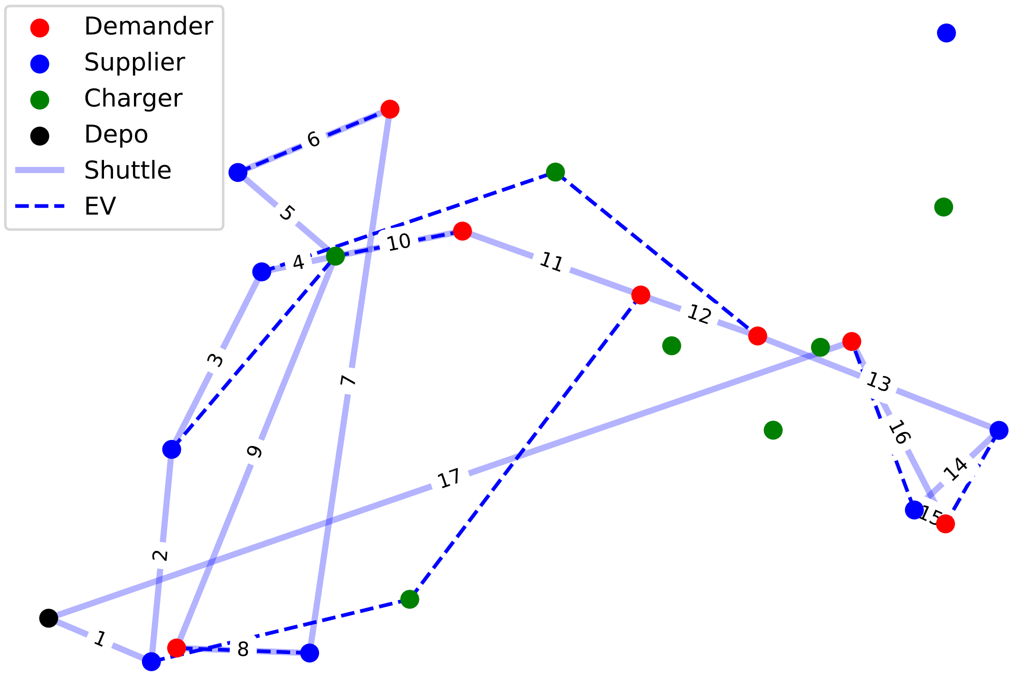


Figure 3.8: Example solution for a single-shuttle case, $|\mathcal{N}| = 23$, $|Dr| = 3$ and $|\mathcal{I}| = 1$.

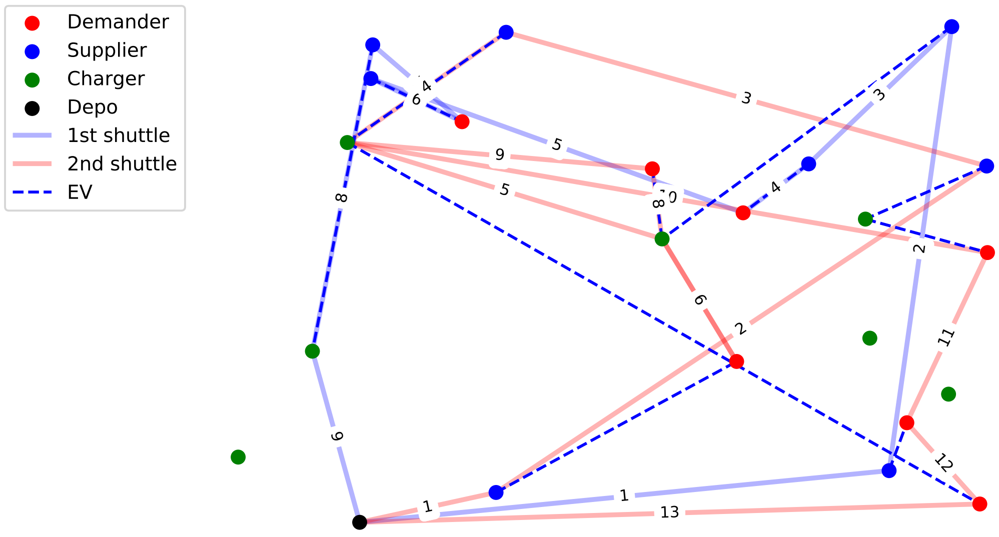


Figure 3.9: Example solution for a multiple-shuttle cases, $|\mathcal{N}| = 23$, $|Dr| = 3$ and $|\mathcal{I}| = 2$.

network, the shuttle returns to the depot after picking up drivers from demander nodes. We can observe the versatility of the produced solutions by looking at the charging stations. For instance, a driver dropped off at the first visited supplier node relocates the EV to a charging station, waits there until the EV is charged, and then relocates it to a demander node. Only then the driver is picked up by a shuttle. In another example, the driver dropped off at the second visited supplier node is picked up immediately at a designated charging station by a shuttle. Similarly, Figure 3.9 represents the sample solution for the case with 2 shuttles. Each shuttle visits supplier nodes first until it runs out of drivers. Then each of them interchangeably visits charger, supplier, and demander nodes and returns to the depot. The flexibility of the produced solutions can be observed when a driver originally dropped at the second visited supplier node by the first shuttle is picked up at a charging station by the second shuttle.

3.5.4 Results on the Amsterdam Cases

We also use real data of FFEVSS representing car2go operations in Amsterdam, the Netherlands, which was collected between May 5th and October 29th, 2016. From the actual data, we collect locations of supplier, demander and charger nodes and reduce the network by removing EVs that do not need relocation/charging. We also group the data into weekdays and weekends, which results in 14 and 12 instances for the respective groups that we use as test data for the experiments. To train an RL agent, we generate on the fly training data by sampling locations of nodes using weekdays data by extracting the CDF of the distribution for nodes' coordinates. We also assign node types randomly by following the similar structure observed in weekdays data. In particular, the training and testing data have 170 nodes with 71 supplier, 71 demander, and 27 charger nodes. Table 3.6 presents the results of the RL agent performance on the Amsterdam dataset. In all instances with except one, the RL agent performs better as compared to the heuristic; overall, RL performs 6.17% better. The poor performance of the RL agent in the instance with a single shuttle and 5 drivers on weekends may be improved by training with more weekend data.

Chapter 4: Hybrid Model for Solving Travelling Salesman Problem with Drone

4.1 Introduction

Last mile delivery, which refers to the transportation of products from a distribution center to the doorstep of a customer, is an integral part of the supply chain. However, last mile delivery is often not cost-effective due to transportation costs associated with individualized shipments, complex routes, and various destinations. Therefore, emerging technologies such as unarmed aerial vehicles (drones) are viewed as a potential breakthrough in reducing inefficiencies in last mile delivery. For instance, Amazon launched the Prime Air program which aims to deliver parcels using drones Amazon (2021). Similarly, Wing owned by Alphabet delivers books, meals, and medicine using drones Wing (2021). The steady development of drone technology including its capacity and flying range enables transporting of various products. For instance, HorseFly drone, used by UPS can carry packages up to 10 pounds and has a flight time of 30 minutes UPS (2021).

While drones certainly have high speed and do not require any infrastructure such as roads, bridges, etc., they also possess some limitations. Drones have a limited flying range, which depends on battery life. Also, drones cannot carry parcels of many customers, thus they must return every time to a distribution center to pick up customer orders. On the other hand, trucks, which are traditionally used for last mile delivery, have a large capacity to store all customer orders and can transport goods to long distances. However, trucks usually have slow speeds due to congestion and require built infrastructure to reach their destinations. Therefore, combining drone and truck is a promising tandem to improve the efficiency of last mile delivery. Indeed, UPS has started testing combining drone and truck to deliver goods UPS (2021). In this setting, a driver of the truck loads a package to drone and sends drone to an autonomous route to an address. Meanwhile truck can serve other customers. When drone returns to truck, the driver swaps the battery of drone and launches the next delivery.

To effectively deploy trucks and drones together for last mile delivery, we must answer several challenging questions such as which customers should be served by drone, which customers should be served by truck, where to recharge drone, how to route both drone and truck, and etc. In the literature, the problem of routing of truck and drone is known as Travelling Salesman Problem with Drone (TSPD) Agatz et al. (2018). The objective of TSPD is to serve customers in a minimal time, while truck and drone are subjected to routing constraints. TSPD is NP-hard problem Agatz et al. (2018), highlighting the need to develop heuristic methods that can scale to large urban networks. Indeed, several exact and heuristic methods have been proposed to solve TSPD in the operations research literature Poikonen et al. (2019); Agatz et al. (2018). However, the proposed solutions fail to generalize and they solve each instance of TSPD from scratch while suffering from either long computational time or low solution quality. Therefore, in this study, we propose using deep reinforcement learning (RL) to solve TSPD, where deep neural networks are utilized to process complex urban network information to help an RL agent to route both truck and drone. Unlike methods coming from the operations research literature, the proposed model generalizes to all instances coming from the same distribution and the trained RL model produces good quality solutions in a matter of seconds.

From a reinforcement learning modeling perspective, TSPD possesses some unique challenges. For instance, the RL agent must learn how to route drone and truck that operate in a shared urban environment. In this case, we must promote cooperation in drone and truck routing decisions while constrained by capacity and flying time limits. In particular, the learned routing policies must consider the timely recharging of drone. Also, unlike regular TSP, where after visiting each customer, that customer no longer needs to be revisited, TSPD allows revisiting some customers, presenting a wider variety in routing decisions. Lastly, due to the distinct characteristics of truck and drone, the RL agent must efficiently explore all resulting routing opportunities.

In this study, we present a novel deep reinforcement learning model called the hybrid model to solve TSPD. The hybrid model consists of an attention encoder and LSTM decoder, which allows to learning policies to promote cooperation in routing truck and drone. As our computational studies show, the proposed hybrid model works well in solving regular TSP and outperforms the existing methods in solving TSPD.

4.2 Related Work

Before presenting the proposed model, we briefly introduce some related studies to the problem and the method.

4.2.1 TSPD

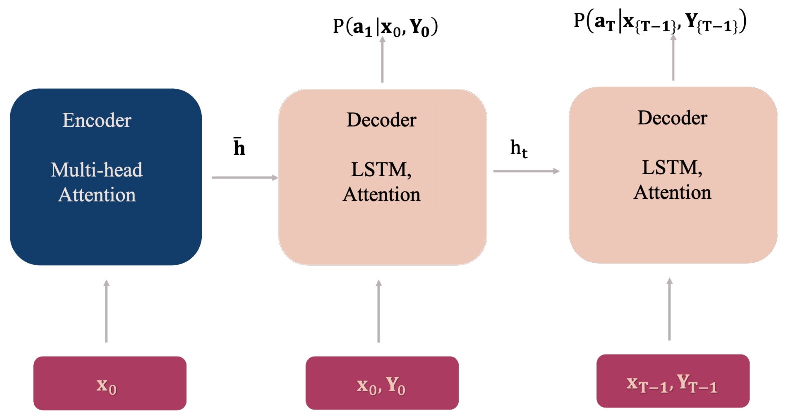
TSPD, in general, has many variants depending on the number of drones and trucks used and routing operations involved in the literature Chung et al. (2020). Most studies focus on a single truck and single drone variation of the problem and consider only customer locations as possible points for recharging drone. Also, it is common to assume that drone can carry only a single parcel at a time. Typically, TSPD is formulated as a mixed-integer program and is known to be NP-hard. Therefore, the existing exact methods solve TSPD with less than 20 customers, while relying on heuristic methods to handle large instances. Some heuristics Agatz et al. (2018); Poikonen et al. (2019); Ha et al. (2018) first solve regular TSP to generate a feasible solution and then assign customers to be served either by drone or truck, while another set of heuristics first assign customers to either drone or truck and then develop routes Wang and Sheu (2019). Nevertheless, the presented heuristics fail to generalize and each instance of the problem must be resolved from scratch.

4.2.2 RL for Combinatorial Optimization Problems

Recently reinforcement learning has received increased attention to solve combinatorial optimization problems. For instance, Bello et al. (2016) proposed using Pointer Networks Vinyals et al. (2015b) and policy gradient methods to solve TSP. Dai et al. (2017) proposed graph embedding trained with DQN algorithm to solve a group of combinatorial problems. Later, Nazari et al. (2018) showed inefficiencies of Pointer Networks and proposed adopting Sequence to Sequence models with an attention mechanism. In particular, Nazari et al. (2018) removed LSTM cells in the encoder and instead used element-wise projections, while a LSTM cell served as a decoder to store the sequence of visited nodes. Dai et al. (2017); Deudon et al. (2018) presented attention models to solve combinatorial problems. In particular, Kool et al. (2018) uses several layers of multi-head attention to encode a graph and proposes the idea of context node used by a decoder to output probabilities of visiting the next node. However, all the studies above focus on well-known problems such as TSP, Vehicle Routing Problem (VRP) and route only a single-vehicle. Another set of studies focus on routing a fleet of vehicles Sykora et al. (2020); Bogyrbayeva et al. (2020) consisting of homogeneous vehicles. However, TSPD considers routing multiple heterogeneous vehicles including truck and drone, which have different properties including different routing constraints.

4.3 The Hybrid Model

We consider a fully connected graph $\mathcal{G}$ with a set of nodes $\mathcal{N}$ representing customer locations, where the distances between nodes are computed using the Euclidean distance. Given the state of a graph $\mathbf{x}$, we aim to learn the routing policy of drone and truck,π, and generate the sequence of nodes to be visited Y. Then using the chain rule, we aim to learn the following conditional probability parametrized by θ:



$P(\mathbf{a_1}|\mathbf{x_0}, \mathbf{Y_0})$
$P(\mathbf{a_T}|\mathbf{x_{\{T-1\}}}, \mathbf{Y_{\{T-1\}}})$
Encoder
Multi-head Attention
$\bar{\mathbf{h}}$
Decoder
LSTM, Attention
$\mathrm{h_t}$
Decoder
LSTM, Attention
$\mathbf{x_0}$
$\mathbf{x_0}, \mathbf{Y_0}$
$\mathbf{x_{T-1}}, \mathbf{Y_{T-1}}$

Figure 4.1: An overview of the hybrid model.

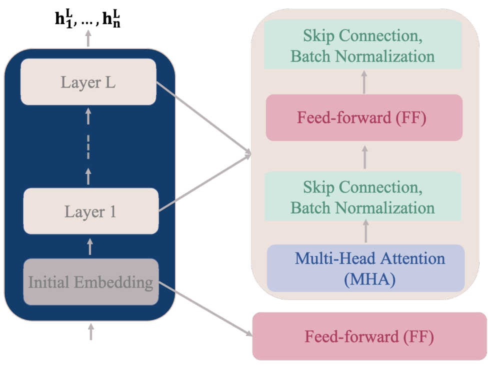
$$\pi_\theta(Y_T|\mathcal{G}) = \prod_{t=0}^{T} p(y_{t+1}|\mathbf{x_t}, Y_t) \tag{4.1}$$

The state of a graph $\mathbf{x_t}$ at time t consists of the static and dynamic features of nodes such as coordinates and distances from other nodes. We start with an empty set Y at time $t = 0$. We develop the hybrid model consisting of an attention-based encoder and a LSTM cell as a decoder to learn π_θ as shown in Figure 4.1. Encoder embeddes the static elements of a graph including the locations of customers. Decoder takes as an input the coordinates of the last visited node and stores the sequence of visited nodes. The outputs of encoder and decoder are passed to the attention which outputs the probabilities of visiting the next node.

4.3.1 Encoder

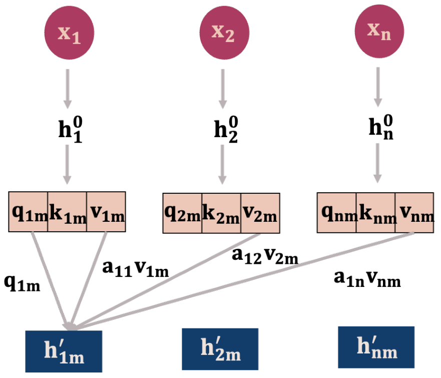
Given a graph with a set of nodes $\mathcal{N}$, we have coordinates of each node in 2-dimensional Euclidean space. Let $\mathbf{x}_n$ represent x and y coordinates for node n. To embed such a fully connected graph we start from the initial embeddings of the nodes using linear projection:

$$\mathbf{h}_n^0 = W^x\mathbf{x} + \mathbf{b}^x \quad \forall n \in \mathcal{N} \tag{4.2}$$



$\mathbf{h}_1^L, \ldots, \mathbf{h}_n^L$
Layer L
Layer 1
Initial Embedding
Skip Connection, Batch Normalization
Feed-forward (FF)
Skip Connection, Batch Normalization
Multi-Head Attention (MHA)
Feed-forward (FF)

(a) The structure of Encoder.



$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_n$
$\mathbf{h}_1^0$
$\mathbf{h}_2^0$
$\mathbf{h}_n^0$
$\mathbf{q}_{1m}$ $\mathbf{k}_{1m}$ $\mathbf{v}_{1m}$
$\mathbf{q}_{2m}$ $\mathbf{k}_{2m}$ $\mathbf{v}_{2m}$
$\mathbf{q}_{nm}$ $\mathbf{k}_{nm}$ $\mathbf{v}_{nm}$
$\mathbf{q}_{1m}$
$\mathbf{a}_{11}\mathbf{v}_{1m}$
$\mathbf{a}_{12}\mathbf{v}_{2m}$
$\mathbf{a}_{1n}\mathbf{v}_{nm}$
$\mathbf{h}'_{1m}$
$\mathbf{h}'_{2m}$
$\mathbf{h}'_{nm}$

(b) m-th head of MHA in the first layer.

Figure 4.2: Encoder overview.

where W^x and b^x are learnable parameters. These initial embeddings of nodes $\mathbf{h}^0$ are then passed through L number of attention layers. Each attention layer, l, consists of two sublayers: a multi-head attention layer and a fully connected feed-forward layer as shown in Figure 4.2a.

A multi-head attention (MHA) layer takes as an input the output of the previous layer (either output from the initial embedding or the output of the previous attention layer) and passes messages between nodes. In particular, for each input to MHA we compute the values of $\mathbf{q} \in \mathbb{R}^{d_q}, \mathbf{k} \in \mathbb{R}^{d_q}, \mathbf{v} \in \mathbb{R}^{d_v}$ or queries, keys and values respectively by projecting the input $\mathbf{h_n}$:

$$\mathbf{q_n} = W^Q \mathbf{h_n}, \quad \mathbf{k_n} = W^K \mathbf{k_n}, \quad \mathbf{v_n} = W^V \mathbf{h_n} \quad \forall n \in N \tag{4.3}$$

where, W^Q, W^K, W^V are trainable parameters with sizes $(d_k \times d_h)$, $(d_k \times d_h)$ and $(d_v \times d_h)$ respectively. From keys and queries we compute compatibility between nodes i and j as follows since we consider the fully connected graph:

$$u_{i,j} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}} \quad \forall i, j \in N \tag{4.4}$$

We use compatibility $u_{i,j}$ to compute the attention weights, $a_{i,j} \in [0,1]$ using softmax:

$$a_{i,j} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{i,j'}}} \tag{4.5}$$

Then a message received by node n is a convex combination of messages received from all nodes as shwon in Figure 4.2b:

$$\mathbf{h}_n^{'} = \sum_j a_{n,j}\mathbf{v}_j \tag{4.6}$$

Instead of using a single head, we use a multi head attention with head size M, which allows to pass different messages from other nodes. Then after computing messages from each head using Equation 4.5, we can combine all massages coming to node n as follows:

$$\text{MHA}_n(\mathbf{h}_1, \ldots, \mathbf{h}_N) = \sum_1^M W_m^O \mathbf{h}_{nm}^{'} \tag{4.7}$$

The output of the MHA sublayer along with skip connection is passed through Batch Normalization:

$$\widehat{\mathbf{h}}_n^l = \mathbf{BN}^l(\mathbf{MHA}(\mathbf{h}_1^{l-1}, \ldots, \mathbf{h}_N^{l-1})) \tag{4.8}$$

The output of Batch Normalization then passed through fully connected feed forward (FF) network with ReLu activation function. We again apply skip connecion and batch normalization to the output of FF.

$$\mathbf{h}_n^l = \mathbf{BN}^l(\widehat{\mathbf{h}}_n^l + \mathbf{FF}^l(\widehat{\mathbf{h}}_n^l)) \tag{4.9}$$

where

$$\mathbf{FF}^l(\widehat{\mathbf{h}}_n^l)) = W^{\text{ff},1} \cdot \text{ReLU}(W^{\text{ff},0}\widehat{\mathbf{h}}_n^l + \mathbf{b}^{\text{ff},0}) + \mathbf{b}^{\text{ff},1} \tag{4.10}$$

The overview of encoder is shown in Figure 4.2.

4.3.2 Decoder

Given the encoder outputs as embeddings of each node in the graph, we pass to the LSTM the embedding of the last node selected $\mathbf{h}_{l_t}^L$.

$$\widetilde{\mathbf{h}}_{t+1}, \widetilde{\mathbf{c}}_{t+1} = \text{rnn}(\mathbf{h}_{l_t}^L, (\widetilde{\mathbf{h}}_t, \widetilde{\mathbf{c}}_t)) \tag{4.11}$$

where $\widetilde{\mathbf{h}}_t$ and $\widetilde{\mathbf{c}}_t$ correspond to the hidden and cell states of the LSTM at time t.

After applying dropout to the output of RNN, we pass its hidden state to attention along with embeddings of all nodes and the distance information from the current location of a decision taker :

$$a_{i,j} = v_a^\top \tanh(W_a[\overline{\mathbf{h}}; \widetilde{\mathbf{h}}_{t+1}; W^d d_{i,j}]) \tag{4.12}$$

where i represents the current location of a decision taker, $\overline{\mathbf{h}}$ is the embeddings of all nodes in the graph, $\widetilde{\mathbf{h}}_{t+1}$ is a hidden state of LSTM, $d_{i,j}$ is the distance from the current node to all other nodes, v_a, W_a, W^d are trainable parameters. Then the computed attention is passed through softmax to produce the probabilities of visiting next node.

$$p_i = \frac{e^{a_{ij}}}{\sum_{j'} e^{a_{i,j'}}} \tag{4.13}$$

Figure 4.3 gives the overview of decoder.

4.4 The Central Controller for Routing

To route both drone and truck in a shared urban environment, we deploy the idea of a central controller. Then a central controller observes the entire graph and cooperatively routes drone and truck to finish serving customers in a minimal time. We can formalize routing drone and truck using a central controller as Markov Decision Process (MDP), where only the current state of the graph defines the next node to be visited. Then we aim to learn directly the policy to route drone and truck using the policy gradient methods. In particular,

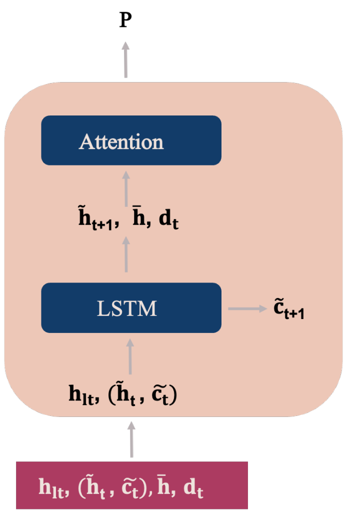


Figure 4.3: Decoder overview

actor network in a form of the hybrid model learns the routing policy, while critic estimates the total time needed to serve all customers given the initial state of the urban network.

Algorithm 4: Distributed training for TSPD

```
Input: Generate a set of $K$ multiple parallel agents, $A = \{a_1, \dots, a_K\}$. Initialize
       actor and critic networks parameters $\{\theta_1^a, \dots, \theta_K^a\}$, $\{\theta_1^c, \dots, \theta_K^c\}$ of $A$. Set
       the maximum number of epochs, $M_{\text{epochs}}$;
1  for $epochs = 1$ to $M_{epochs}$ do
2      Initialize a set of test rewards, $R = \{\}$;
3      for $k = 1$ to $K$ do
4          Reset gradients $d\theta_k^a, d\theta_k^c$;
5          $B_k \sim \text{DataGenerator}(\rho)$;
6          $\text{RolloutUpdate}(a_k, B_k)$;
7          $r_k = \text{Test}(a_k)$;
8          $R = R \cup r_k$;
9      $j \leftarrow \arg\min(R)$;
10     for $k = 1$ to $K$ do
11         $\theta_k^a \leftarrow \theta_j^a$;
12         $\theta_k^c \leftarrow \theta_j^c$
```

For an efficient exploration of an RL agent, we trained the model in a distributed setting as shown in Algorithm 4 . First, we generate multiple RL agents in a parallelized manner with the same initial weights. Then each agent rollouts a batch of episodes with different

Algorithm 5: RolloutUpdate

```
Input: Batch of data $B$ with a number of episodes denoted $M_{epis}$, an agent id $k$.
       Set the maximum number of steps denoted, $T$;
1  Create an empty list logs, set $R = 0$;
2  Initialize $(\widetilde{h}_0, \widetilde{c}_0)$;
3  $x_0$, $mask_0$ = simulator.reset($B$);
4  for $t$=0 to $T$ do
5      $a_t^{tr}$, log, $(\widetilde{h}_{t'}, \widetilde{c}_{t'})$ = actor-network($x_t^{tr}$, $mask_t^{tr}$, $(\widetilde{h}_t, \widetilde{c}_t)$);
6      add log to logs;
7      $a_t^{d}$, log, $(\widetilde{h}_t, \widetilde{c}_t)$ = actor-network($x_t^{d}$, $mask_t^{d}$, $(\widetilde{h}_{t'}, \widetilde{c}_{t'})$);
8      add log to logs;
9      $x_{t+1}$, $mask$, $r_t$ = simulator.step($a_t^{tr}$, $a_t^{d}$);
10     $R = R + r_t$;
11 logs = $\sum_{t=0}^{2T} \text{logs}_t$;
12 Calculate $V(x_0; \theta_k^c)$ using critic;
13 $d\theta_k^a = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} (R^m - V^m(x_0^m; \theta_k^c)) \nabla_{\theta_k^a} \text{logs}^m$;
14 $d\theta_k^c = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} \nabla_{\theta_k^c} (R^m - V^m(x_0^m; \theta_k^c))^2$;
```

data instances generated by a distrubution denoted as ρ and updates its weights by the REINFORCE algorithm (Williams, 1992) using the advantage function as a baseline as shown in Algorithm 5. Each agent tests the updated weights with a shared test instance. Finally, we pick the agent with the best testing reward, and copy the weight of the best agent to the weight of the other agents. We repeat this procedure iteratively each epoch during training.

4.5 Computational Experiments

4.5.1 Data Generation and Configurations

We consider a graph on 100×100 square miles, where x and y coordinates of each node are sampled from a uniform distribution with values ranging from 1 to 100. Similarly, we sample x and y coordinates of depot from a uniform distribution with values ranging from 0, 1. For the general setting of TSPD, we consider an infinite time range for drone. However, drone can visit a single customer per launch and after serving a customer it must be recharged.

Each node in a graph can be visited by either drone, truck or by both, but only once. Then we develop a masking scheme to restrict available nodes to be visited by drone or truck according to the above assumptions.

Our implementations are done in Python 3.8 using PyTorch 1.5. The architecture of the critic network has similarities to the actor network, except we do not use recurrent neural networks. The goal of the critic network is to estimate the total time needed to serve all the customers and return back to depot, which is achieved through embedding the initial state of the graph. In particular, we embed x and y coordinates of the nodes through element-wise projections with 1D convolution networks whose outputs are passed through attention followed by feed-forward networks.

4.5.2 Benchmarks and RL Methods

There are exisist two groups of methods to solved TSPD. The first group solves the problem exactly and allows revisiting nodes visited both by drone and truck (Agatz et al., 2018). The second group solves the problem using the heuristic methods by splitting the graph into several subgraphs Poikonen et al. (2019). For instance, divide-and-conquer heuristic (DCH) of Poikonen et al. (2019) splits the network into n-groups and solves each subgroup using the TSP-ep-all heuristic of Agatz et al. (2018). The TSP-ep-all heuristic aims to partition exactly a graph into nodes to be served by drone and truck, by considering different combinations. Therefore, TSP-ep-all provides high-quality solutions on small-sized graphs and fails to generalize into large-sized graphs.

There are exist two methods by which we sample solutions from the trained hybrid model. In the first method called *greedy*, we always select nodes with the highest probabilities to visit at each time step. In the second method, called *sampling*, we sample several solutions from the trained model, by selecting randomly nodes to visit from output probabilities. Then we select the sample solution with the lowest reward as the final solution.

Table 4.1: The optimal solutions vs. Reinforcement Learning solutions on a grpah with 11 nodes.

Instance	Optimal	RL Greedy	Gap Greedy, %	RL Sampling	Gap Sampling, %
1	221.19	223.41	1.00	223.41	1.00
2	205.76	205.76	0.00	205.76	0.00
3	192.96	203.93	5.68	200.01	3.65
4	241.26	241.26	0.00	241.26	0.00
5	248.14	248.82	0.27	248.82	0.27
6	217.69	217.69	0.00	217.69	0.00
7	237.34	244.95	3.21	237.34	0.00
8	214.77	226.64	5.53	225.36	4.93
9	256.34	269.80	5.25	256.83	0.19
10	227.90	227.90	0.00	227.90	0.00
mean	**226.33**	**231.01**	**2.09**	**228.44**	**1.01**

4.5.3 Results

First, we measure the performance of the hybrid model against the optimal solutions (Agatz et al., 2018). In a graph with 11 nodes with 10 instances shown in Table 4.1, the greedy RL solutions result in 2.09% optimality gap, while the sampling method reduced the optimality gap by half. Figure 4.4, represents the optimal solution of 6-th instance found by the greedy RL method. After leaving the depot, drone and truck visit customer locations and drone returns back to the customers served by truck to recharge. We also, note that drone mostly serves customers located far away from depot and in total travels longer distances compared to truck.

In the second set of experiments we compare the performance of the proposed model against the heuristics from the operations research literature in graphs of various sizes. As shown in Table 4.2, the sampling RL method outperforms the existing heuristic in a majority of instances. Table 4.3 shows the computational time of heuristics and reinforcement learning agents, where a greedy solution can be invoked instantaneously.

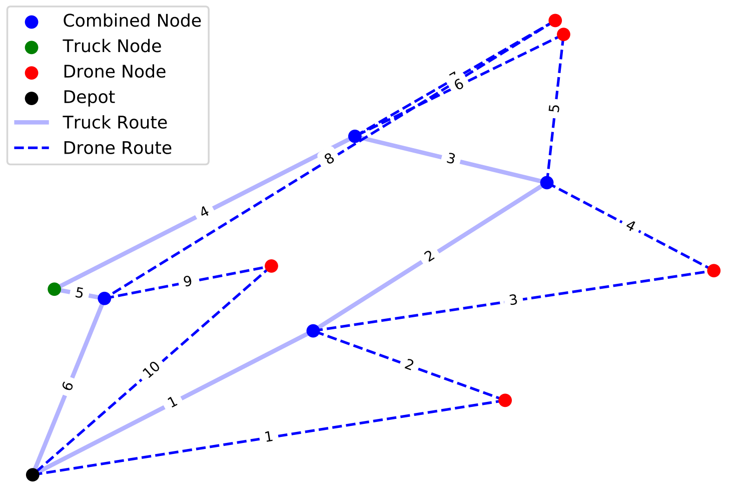


Figure 4.4: An example optimal solution produced by RL.

Table 4.2: The heuristic solutions vs. reinforcement learning solutions on a grpah with a various number of nodes. The average of 100 instances are reported.

$\mathcal{N}$	TSP-ep-all	DCH	RL Greedy	Gap Greedy, %	RL Sampling	Gap Sampling %
11	230.57	-	234.48	1.89	229.15	-0.41
15	255.53	-	262.63	2.80	256.03	0.23
20	281.71	292.11	288.55	2.46	282.10	0.16
50	-	423.89	439.20	3.82	412.29	-2.53
100	-	570.90	568.60	-0.48	548.92	-3.85

Table 4.3: The computational time of the heuristics and reinforcement learning agents on a grpah with a various number of nodes in terms of CPU time, sec. The average of 100 instances are reported.

$\mathcal{N}$	TSP-ep-all	DCH	RL Greedy	RL Sampling
11	0.12	-	0.00	1.13
15	0.50	-	0.00	1.61
20	0.57	0.04	0.00	1.72
50	-	0.10	0.00	5.27
100	-	0.34	0.02	20.15

Chapter 5: Conclusions and Future Research Directions

In this dissertation, we use optimization and machine learning approaches to solve problems in urban transportation.

In the first problem, we propose the design of an iterative combinatorial auction for the fractional ownership of AVs. The auction allows bidding for combinations of time slots for customers with no restrictions on their lengths. Also, the iterative nature of the auction enables customers to learn the bidding prices needed to win interested time slots. We also devise practical tools such as algorithms and user agents to help bidders and an auctioneer apply the auction in the real life.

As a future direction, the auction can include machine learning methods to learn the bidding behavior of customers to help them to generate new time slots to bid and learn the bidding prices for the packages of time slots. Also, the Winner Determination Problem solution methods can be advanced to improve the solution quality for auctions with multiple homogenous vehicles. We also need to investigate how the properties of the auction will change if a fleet of heterogeneous vehicles is offered.

In the second problem, we proposed a reinforcement learning approach to route a fleet of homogenous shuttles to rebalance FFEVSS. The proposed methods allow solving the challenging task of relocating and efficiently recharging EVs in a static network in the absence of customer demand. Also, the flexibility of the reinforcement method enables to solve the hard instances of the problem, when charging stations must be reused.

In the future, the problem can be extended to the dynamic version of the problem, when we need to rebalance the system in the presence of customer demand. Also, the rule-based approach to decide where to relocate EVs can be replaced with a more sophisticated method that will take into account the long-term rewards associated with relocating EVs.

In the last problem, we develop a novel hybrid model to route drone and truck to serve customers for last mile delivery. The hybrid model allows to cooperatively route drone and truck through encoding the graph using multi-head attention and decoding with the combination of a LSTM cell and attention. The proposed method produces good quality solutions and scales on large instances of the problem.

The hybrid model can be later applied to route multiple drones and trucks. Also, it will be interesting to see other applications involving routing heterogenous vehicles being solved by the hybrid model and the comparison of its results with other optimization and reinforcement learning methods.

References

Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. *Transportation Science* 52, 4, 965–981.

Akbar, A., Aasen, A.K.A., Msakni, M.K., Fagerholt, K., Lindstad, E., Meisel, F., . An economic analysis of introducing autonomous ships in a short-sea liner shipping network. *International Transactions in Operational Research* n/a, n/a. https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12788.

Amazon, 2021. Amazon Prime Air. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011. Accessed: 2021-03-10.

An, N., Elmaghraby, W., Keskinocak, P., 2005. Bidding strategies and their impact on revenues in combinatorial auctions. *Journal of Revenue and Pricing Management* 3, 4, 337–357.

Ausubel, L.M., Cramton, P., Milgrom, P., 2006. The clock-proxy auction: A practical combinatorial auction design. *Handbook of Spectrum Auction Design* pp. 120–140.

Ausubel, L.M., Cramton, P., Pycia, M., Rostek, M., Weretka, M., 2014. Demand reduction and inefficiency in multi-unit auctions. *The Review of Economic Studies* 81, 4, 1366–1400.

Ausubel, L.M., Cramton, P., et al., 2011. Activity rules for the combinatorial clock auction. *Department of Economics, University of Maryland, Discussion Paper*

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*

Bezanson, J., Karpinski, S., Shah, V.B., Edelman, A., 2012. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*

Bogyrbayeva, A., Jang, S., Shah, A., Jang, Y.J., Kwon, C., 2020. A reinforcement learning approach for rebalancing electric vehicle sharing systems. *arXiv preprint arXiv:2010.02369*

Buşoniu, L., Babuška, R., De Schutter, B., 2010. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*. Springer, pp. 183–221.

Cantillon, E., Pesendorfer, M., 2006. Auctioning bus routes: The London experience. *Combinatorial auctions*

car2go, 2021a. Do you know: What a "relocation" is and which car2gos are relocated? https://blog.car2go.com/2017/06/26/know-relocation-car2gos-relocated/. Accessed: 2021-02-24.

car2go, 2021b. The electric drive in Montreal. https://www.car2go.com/NA/en/nextgen/. Accessed: 2021-02-24.

Center Microeconomic Data , 2018. Household Debt and Credit. Technical report.

Chianese, Y., Avenali, A., Gambuti, R., Palagi, L., 2017. One-way free floating car-sharing: applying vehicle-generated data to assess the market demand potential of urban zones. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2, IEEE, pp. 777–782.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research* p. 105004.

Clarke, E.H., 1971. Multipart pricing of public goods. *Public Choice* 11, 1, 17–33.

Cramton, P., 2013. Spectrum auction design. *Review of Industrial Organization* 42, 2, 161–190.

Dai, H., Khalil, E.B., Zhang, Y., Dilkina, B., Song, L., 2017. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*

Day, R.W., Cramton, P., 2012. Quadratic core-selecting payment rules for combinatorial auctions. *Operations Research* 60, 3, 588–603.

De Vries, S., Vohra, R.V., 2003. Combinatorial auctions: A survey. *INFORMS Journal on Computing* 15, 3, 284–309.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., Rousseau, L.M., 2018. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, Springer, pp. 170–181.

Dunning, I., Huchette, J., Lubin, M., 2017. JuMP: A modeling language for mathematical optimization. *SIAM Review* 59, 2, 295–320.

Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S., 2018. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*.

Folkestad, C.A., Hansen, N., Fagerholt, K., Andersson, H., Pantuso, G., 2020. Optimal charging and repositioning of electric vehicles in a free-floating carsharing system. *Computers & Operations Research* 113, 104771.

Gansterer, M., Hartl, R.F., 2017. Collaborative vehicle routing: a survey. *European Journal of Operational Research* 268, 1–12.

Groves, T., 1973. Incentives in teams. *Econometrica: Journal of the Econometric Society* 41, 4, 617–631.

Gu, Y., Goez, J.C., Guajardo, M., Wallace, S.W., . Autonomous vessels: state of the art and potential opportunities in logistics. *International Transactions in Operational Research* n/a, n/a. https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12785.

Gupta, J.K., Egorov, M., Kochenderfer, M., 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, Springer, pp. 66–83.

Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2018. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies* 86, 597–621.

Haider, Z., Charkhgard, H., Kim, S.W., Kwon, C., 2019. Optimizing the relocation operations of free-floating electric vehicle sharing systems. Available at SSRN: http://dx.doi.org/10.2139/ssrn.3480725.

Hara, Y., Hato, E., 2018. A car sharing auction with temporal-spatial OD connection conditions. *Transportation Research Part B: Methodological* 117, 723–739.

He, L., Ma, G., Qi, W., Wang, X., 2020. Charging an electric vehicle-sharing fleet. *Manufacturing & Service Operations Management*

Herrmann, S., Schulte, F., Voß, S., 2014. Increasing acceptance of free-floating car sharing systems using smart relocation strategies: a survey based study of car2go hamburg. In *International conference on computational logistics*, Springer, pp. 151–162.

Hershberger, J., Suri, S., 2001. Vickrey prices and shortest paths: What is an edge worth? In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, IEEE, pp. 252–259.

Hoffman, K., Menon, D., van den Heever, S., Wilson, T., 2006. Observations and near-direct implementations of the ascending proxy auction. *Combinatorial Auctions* pp. 415–450.

James, J., Lam, A.Y., Lu, Z., 2018. Double auction-based pricing mechanism for autonomous vehicle public transportation system. *IEEE Transactions on Intelligent Vehicles* 3, 2, 151–162.

James, J., Yu, W., Gu, J., 2019. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* 20, 10, 3806–3817.

Kalakanti, A.K., Verma, S., Paul, T., Yoshida, T., 2019. Rl solver pro: Reinforcement learning for solving vehicle routing problem. In *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, IEEE, pp. 94–99.

Kool, W., Van Hoof, H., Welling, M., 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*

Kuo, A., Miller-Hooks, E., 2015. Combinatorial auctions of railway track capacity in vertically separated freight transport markets. *Journal of Rail Transport Planning & Management* 5, 1, 1–11.

Kurz, C.J., Li, G., Vine, D.J., 2016. The Young and the Carless? The Demographics of New Vehicle Purchases. Technical report, Board of Governors of the Federal Reserve System (US).

Kypriadis, D., Pantziou, G., Konstantopoulos, C., Gavalas, D., 2018. Minimum walking static repositioning in free-floating electric car-sharing systems. In *2018 21st international conference on intelligent transportation systems (ITSC)*, IEEE, pp. 1540–1545.

Lin, K., Zhao, R., Xu, Z., Zhou, J., 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390.

Milgrom, P., 2019. Auction market design: Recent innovations. *Annual Review of Economics* 11, 1, 383–405. https://doi.org/10.1146/annurev-economics-080218-025818.

Nazari, M., Oroojlooy, A., Snyder, L., Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pp. 9839–9849.

Ofcom, 2011. Consultation on Assessment of Future Mobile Competition and Proposals for the Award of 800 MHz and 2.6 GHz Spectrum and Related Issues. Technical report, Annexes.

Oliehoek, F.A., Amato, C., et al., 2016. *A concise introduction to decentralized POMDPs*, Vol. 1. Springer.

Olivares, M., Weintraub, G.Y., Epstein, R., Yung, D., 2012. Combinatorial auctions for procurement: An empirical study of the Chilean school meals auction. *Management Science* 58, 8, 1458–1481.

Parkes, D.C., 2006. *Iterative combinatorial auctions*. MIT press.

Pekeč, A., Rothkopf, M.H., 2003. Combinatorial auction design. *Management Science* 49, 11, 1485–1503.

Perozzi, B., Al-Rfou, R., Skiena, S., 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

Poikonen, S., Golden, B., Wasil, E.A., 2019. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing* 31, 2, 335–346.

Rassenti, S.J., Smith, V.L., Bulfin, R.L., 1982. A Combinatorial Auction Mechanism for Airport Time Slot Allocation. *The Bell Journal of Economics* 13, 2, 402–417.

Rothkopf, M.H., 2007. Thirteen reasons why the Vickrey-Clarke-Groves process is not practical. *Operations Research* 55, 2, 191–197.

Sadeghianpourhamami, N., Deleu, J., Develder, C., 2018. Achieving scalable model-free demand response in charging an electric vehicle fleet with reinforcement learning. In *Proceedings of the Ninth International Conference on Future Energy Systems*, pp. 411–413.

Santos, A.G., Cândido, P.G., Balardino, A.F., Herbawi, W., 2017. Vehicle relocation problem in free floating carsharing using multiple shuttles. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, pp. 2544–2551.

Schulte, F., Voß, S., 2015. Decision support for environmental-friendly vehicle relocations in free-floating car sharing systems: The case of car2go. *Procedia CIRP* 30, 275–280.

Shi, J., Gao, Y., Wang, W., Yu, N., Ioannou, P.A., 2019. Operating electric vehicle fleet for ride-hailing services with reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*

Shoup, D.C., 2005. *The High Cost of Free Parking*, Vol. 206. Planners Press Chicago.

Smet, P., 2021. Ride sharing with flexible participants: a metaheuristic approach for large-scale problems. *International Transactions in Operational Research* 28, 1, 91–118. https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12737.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112.

Sykora, Q., Ren, M., Urtasun, R., 2020. Multi-agent routing value iteration network. In *International Conference on Machine Learning*, PMLR, pp. 9300–9310.

Takalloo, M., Bogyrbayeva, A., Charkhgard, H., Kwon, C., . Solving the winner determination problem in combinatorial auctions for fractional ownership of autonomous vehicles. *International Transactions in Operational Research* n/a, n/a. https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12868.

UPS, 2021. UPS Flight Forward Drone Delivery. https://www.ups.com/us/en/services/shipping-services/flight-forward-drones.page. Accessed: 2021-03-10.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008.

Vickrey, W., 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance* 16, 1, 8–37.

Vinyals, O., Bengio, S., Kudlur, M., 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*

Vinyals, O., Fortunato, M., Jaitly, N., 2015b. Pointer networks. *arXiv preprint arXiv:1506.03134*

Volkswagen, 2021. I have got 1,500 cars here in Berlin. https://www.volkswagenag.com/en/news/stories/2019/06/ive-got-1500-cars-here-in-berlin.html. Accessed: 2021-02-24.

Wang, X., Agatz, N., Erera, A., 2018. Stable matching for dynamic ride-sharing systems. *Transportation Science* 52, 4, 850–867.

Wang, Z., Sheu, J.B., 2019. Vehicle routing problem with drones. *Transportation research part B: methodological* 122, 350–364.

Weikl, S., Bogenberger, K., 2013. Relocation strategies and algorithms for free-floating car sharing systems. *IEEE Intelligent Transportation Systems Magazine* 5, 4, 100–111.

Weikl, S., Bogenberger, K., 2015. A practice-ready relocation model for free-floating carsharing systems with electric vehicles–mesoscopic approach and field trial results. *Transportation Research Part C: Emerging Technologies* 57, 206–223.

Wen, J., Zhao, J., Jaillet, P., 2017. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, pp. 220–225.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4, 229–256.

Wing, 2021. Wing delivers library books to students in Virginia. https://blog.wing.com/2020/06/wing-delivers-library-books-to-students.html. Accessed: 2021-03-10.

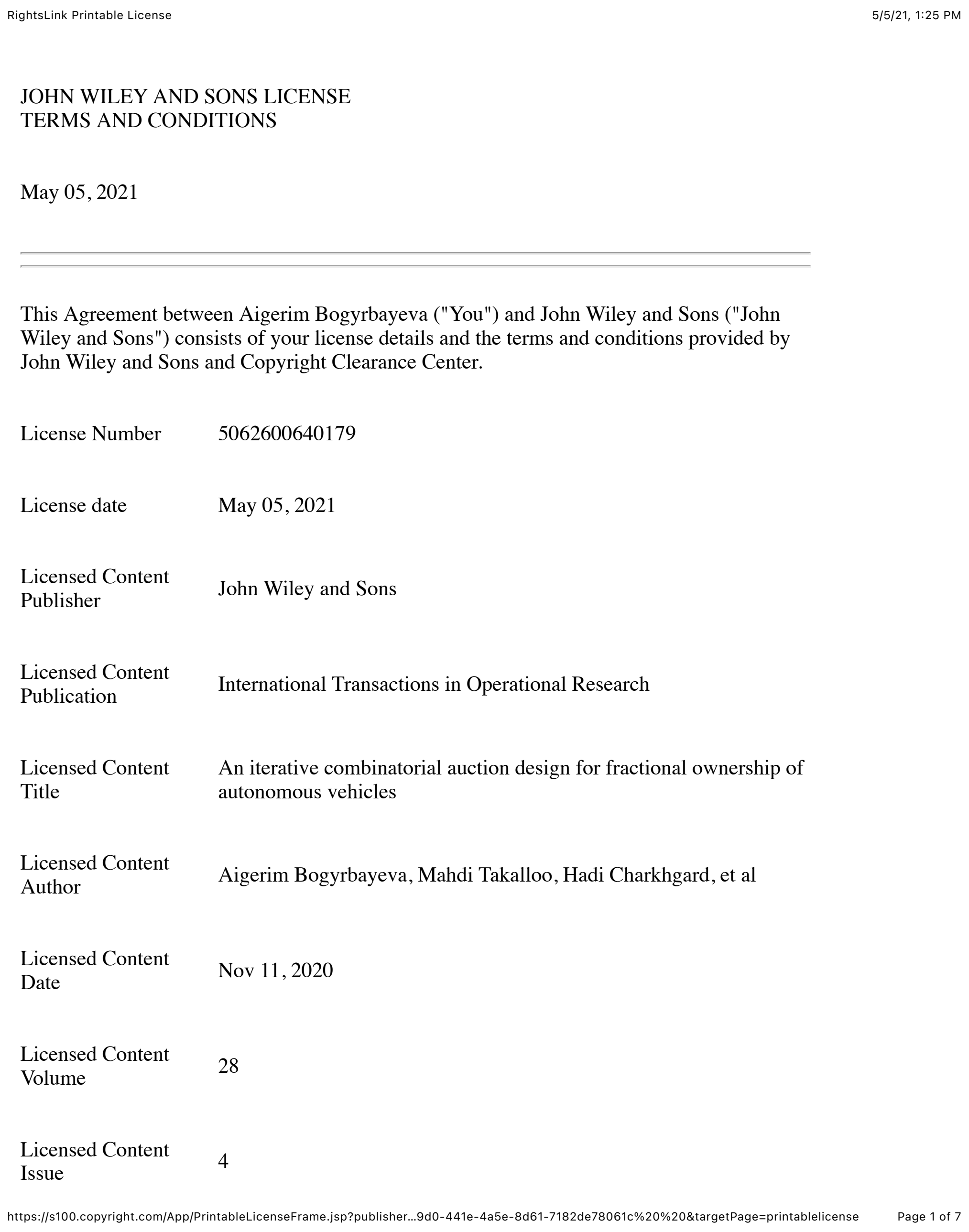
Zhang, A., Kang, J.E., Kwon, C., 2020. Generalized stable user matching for autonomous vehicle co-ownership programs. *Service Science* 12, 2-3, 61–79. https://doi.org/10.1287/serv.2020.0257.

Zhang, B., Yao, T., Friesz, T.L., Sun, Y., 2015. A tractable two-stage robust winner determination model for truckload service procurement via combinatorial auctions. *Transportation Research Part B: Methodological* 78, 16–31.

Zhao, J., Mao, M., Zhao, X., Zou, J., 2020. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems*

Zhao, M., Li, X., Yin, J., Cui, J., Yang, L., An, S., 2018. An integrated framework for electric vehicle rebalancing and staff relocation in one-way carsharing systems: Model formulation and lagrangian relaxation-based solution approach. *Transportation Research Part B: Methodological* 117, 542–572.

Appendix A: Copyright Permission for Chapter 2



RightsLink Printable License 5/5/21, 1:25 PM

License Number	5062600640179
License date	May 05, 2021
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	International Transactions in Operational Research
Licensed Content Title	An iterative combinatorial auction design for fractional ownership of autonomous vehicles
Licensed Content Author	Aigerim Bogyrbayeva, Mahdi Takalloo, Hadi Charkhgard, et al
Licensed Content Date	Nov 11, 2020
Licensed Content Volume	28
Licensed Content Issue	4

https://s100.copyright.com/App/PrintableLicenseFrame.jsp?publisher…9d0-441e-4a5e-8d61-7182de78061c%20%20&targetPage=printablelicense Page 1 of 7

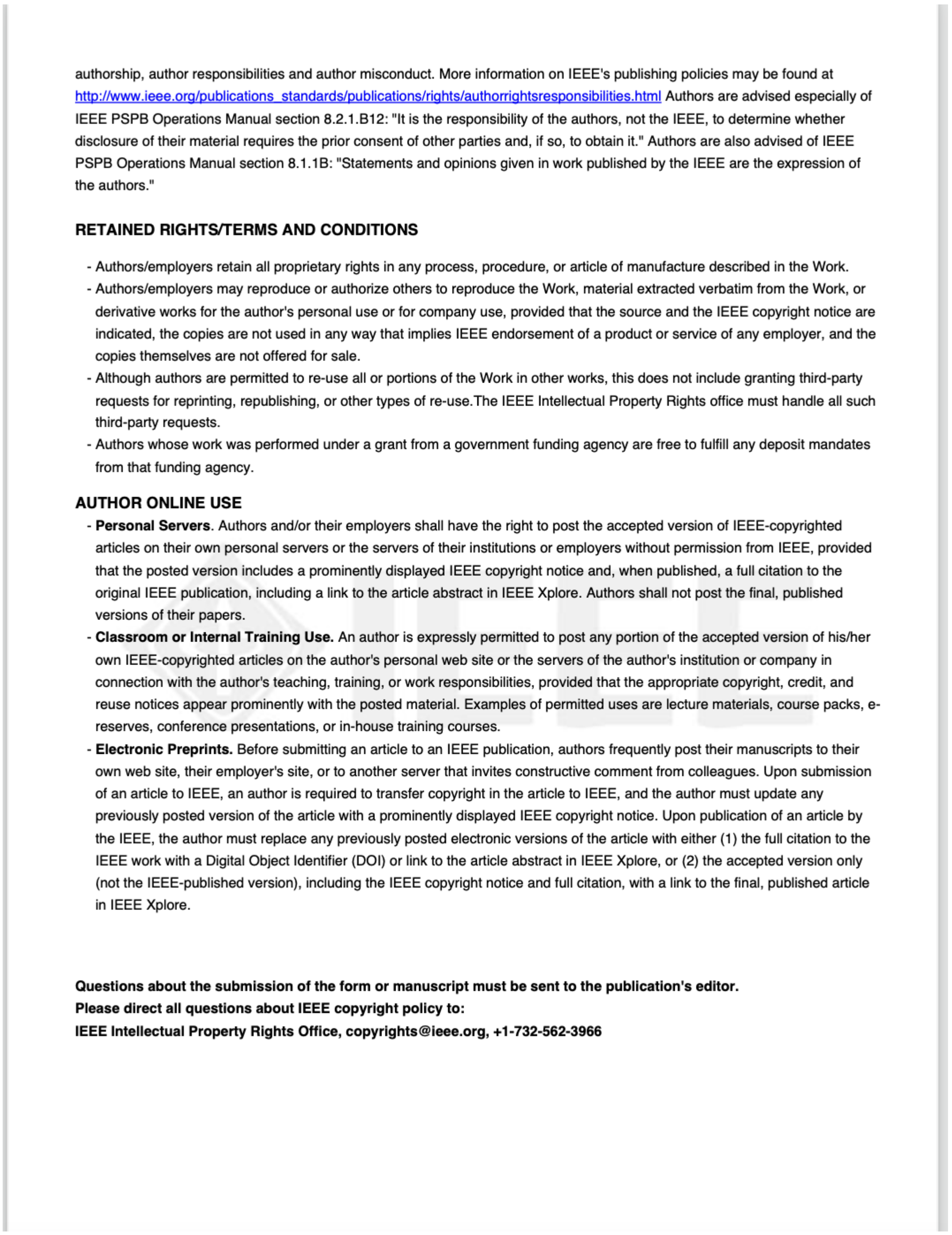
Licensed Content Pages	25
Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley article
Format	Electronic
Portion	Full article
Will you be translating?	No
Title	An iterative combinatorial auction design for fractional ownership of autonomous vehicles
Institution name	University of South Florida
Expected presentation date	May 2021
Requestor Location	Aigerim Bogyrbayeva 14618 Grenadine Dr apt 6 TAMPA, FL 33613 United States Attn: University of South Florida
Publisher Tax ID	EU826007151
Total	0.00 USD

Terms and Conditions

TERMS AND CONDITIONS

Licensed Content Pages	25
Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley article
Format	Electronic
Portion	Full article
Will you be translating?	No
Title	An iterative combinatorial auction design for fractional ownership of autonomous vehicles
Institution name	University of South Florida
Expected presentation date	May 2021
Requestor Location	Aigerim Bogyrbayeva 14618 Grenadine Dr apt 6 TAMPA, FL 33613 United States Attn: University of South Florida
Publisher Tax ID	EU826007151
Total	0.00 USD

Appendix B: Copyright Permission for Chapter 3



authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

Questions about the submission of the form or manuscript must be sent to the publication's editor.
Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966